aws

Guida per l'utente per Aurora

Amazon Aurora

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Aurora: Guida per l'utente per Aurora

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è Aurora? 1
Modello di responsabilità RDS condivisa di Amazon 2
Come funziona Amazon Aurora con Amazon RDS 2
Cluster di database Aurora 4
Versione di Aurora 6
Motori di database supportati 6
Specificare una versione di Aurora 7
Versioni Aurora 7
Aggiornamenti del cluster Aurora DB 13
Supporto per la versione Aurora 16
Regioni e zone di disponibilità 18
AWS Regioni 19
Zone di disponibilità 29
Fuso orario locale per i cluster DB 30
Funzionalità supportate da Aurora per regione e motore 36
Convenzioni tabella 37
Distribuzioni blu/verdi 37
Configurazioni dei cluster Aurora 38
Flussi di attività di database 38
Esportazione dei dati del cluster in Amazon S3 47
Esportazione dei dati snapshot in Simple Storage Service (Amazon S3) 49
Database globali di Aurora 50
Autenticazione del database IAM 61
Autenticazione Kerberos 62
Aurora machine learning 69
Approfondimenti sulle prestazioni 78
Integrazioni Zero-ETL 89
Server proxy per RDS 93
Integrazione di Secrets Manager 106
Aurora Serverless v2 106
Aurora Serverless v1 112
API dati RDS 117
Applicazione ZDP di patch senza tempi di inattività () 127
Funzionalità native del motore 128

Connessioni agli endpoint 129
Tipi di endpoint di Aurora 129
Visualizzazione degli endpoint 131
Endpoint ed elevata disponibilità 132
Endpoint del cluster 133
Endpoint di lettura 134
Endpoint dell'istanza 135
Endpoint personalizzati 136
Classi di istanze database 156
Tipi di classi di istanza database 156
Motori di database supportati 159
Determinazione del supporto delle classi di istanze DB in Regioni AWS 172
Specifiche dell'hardware 177
Storage 188
Panoramica dello storage di Aurora 189
Contenuto del volume del cluster 189
Configurazione dell'archiviazione dei cluster Aurora 189
Ridimensionamento dello storage 190
Fatturazione dei dati 192
Affidabilità 192
Riparazione automatica dello storage 193
Cache delle pagine superstite 193
Ripristino in seguito a riavvii non pianificati 194
Sicurezza di Aurora 195
Utilizzo SSL con i cluster Aurora DB 196
Elevata disponibilità 196
Elevata disponibilità di dati Aurora 197
Architetture ad alta disponibilità per istanze database Aurora 197
Elevata disponibilità nelle regioni AWS con database globali Aurora 198
Tolleranza ai guasti 198
Elevata disponibilità con Amazon RDS Proxy 201
Replica 201
Repliche di Aurora 201
Aurora MySQL 204
Aurora PostgreSQL 204
Fatturazione delle istanze database per Aurora 205

Istanze di database on demand 207
Istanze database riservate 209
Configurazione dell'ambiente 225
Registrati per un Account AWS 225
Crea un utente con accesso amministrativo 226
Concessione dell'accesso programmatico 227
Determinazione dei requisiti 228
Fornire l'accesso al cluster DB 230
Nozioni di base 233
Creazione e connessione a un cluster di database Aurora MySQL 233
Prerequisiti 235
Fase 1: creazione di un'istanza EC2 235
Fase 2: creazione di un cluster di database Aurora MySQL 241
(Facoltativo) Crea VPC, istanza EC2 e cluster Aurora MySQL utilizzando AWS CloudFormation 246
Fase 3: connessione a un cluster di database Aurora MySQL 248
Fase 4: eliminazione dell'istanza EC2 e del cluster di database 251
(Facoltativo) Elimina l'istanza EC2 e il cluster DB creati con CloudFormation 252
(Facoltativo) Connessione del cluster database a una funzione Lambda 253
Creazione e connessione di un cluster di database Aurora PostgreSQL 253
Prerequisiti 255
Fase 1: creazione di un'istanza EC2 255
Fase 2: creazione di un cluster di database Aurora PostgreSQL 261
(Facoltativo) Crea VPC, istanza EC2 e cluster Aurora PostgreSQL utilizzando AWS CloudFormation 266
Fase 3: connessione a un cluster di database Aurora PostgreSQL 268
Fase 4: eliminazione dell'istanza EC2 e del cluster di database 271
(Facoltativo) Elimina l'istanza EC2 e il cluster DB creati con CloudFormation 272
(Facoltativo) Connessione del cluster database a una funzione Lambda 272
Tutorial: creazione di un server Web e di un cluster database Amazon Aurora 274
Avvia un'EC2istanza per connetterti con l' 275
Creazione di un cluster DB 281
Installazione di un server Web 292
Tutorial e codice di esempio 305
Tutorial in questa guida 305
Tutorial in altre guide AWS 306

AWS portale di contenuti per workshop e laboratori per PostgreSQL 307
AWS portale di contenuti per workshop e laboratori per MySQL 309
Tutorial ed esempi di codice in GitHub 310
Lavorare con AWS SDKs 310
Accesso programmatico ad 312
Console-to-Code 313
Configurazione di un cluster Aurora DB 314
Creazione di un cluster di database 315
Prerequisiti 316
Creazione di un cluster di database 322
Impostazioni disponibili 332
Impostazioni che non si applicano ad Aurora per i cluster di database 354
Impostazioni che non si applicano alle istanze database Aurora 355
Creazione di risorse con AWS CloudFormation 358
Aurora e modelli AWS CloudFormation 358
Scopri di più su AWS CloudFormation 358
Connessione a un cluster database 359
Connessione ai cluster Aurora DB con i driver AWS 360
Connessione ad Aurora MySQL 362
Connessione ad Aurora PostgreSQL 368
Risoluzione dei problemi relativi alle connessioni 371
Gruppi di parametri 372
Panoramica dei gruppi di parametri 372
Gruppi di parametri di cluster database 376
Gruppi di parametri database 395
Confronto di gruppi di parametri database 412
Specifica dei parametri del database 413
Migrazione di dati a un cluster database 419
Aurora Mia SQL 419
Aurora Postgre SQL 419
Creazione di una ElastiCache cache da Amazon RDS 420
Panoramica della creazione di ElastiCache cache con le impostazioni dell'istanza DB del cluster Aurora RDS 420
..... 421
Migrazione automatica dei database EC2 424
Panoramica 424

Prerequisiti 425
Limitazioni 426
Creazione di risorse IAM 427
Configurare la migrazione dei dati 434
Gestione delle migrazioni 436
Monitoraggio 439
Tutorial: creazione di un cluster con un personalizzato e un nuovo gruppo di opzioni 441
Prerequisiti 442
Crea un gruppo di parametri personalizzato 442
Modifica il valore dei parametri 442
Crea il tuo cluster DB di 443
Gestione di un cluster DB Aurora 445
Avvio e arresto di un cluster 446
Panoramica dell'avvio e dell'arresto di un cluster. 446
Restrizioni 447
Arresto di un cluster di database 448
Mentre il cluster di database è in fase di arresto 449
Avvio di un cluster di database 449
Connessione di un'istanza EC2 451
Panoramica 452
Connessione di un'istanza EC2 457
Visualizzazione delle risorse di calcolo connesse 460
Connessione a un'istanza database che esegue un motore DB specifico 461
Connessione di una funzione Lambda 462
Panoramica 464
Connessione di una funzione Lambda 475
Visualizzazione delle risorse di calcolo connesse 477
Modifica di un cluster database Aurora 479
Modifica del cluster di database tramite la console, la CLI e l'API 479
Modifica di un'istanza database in un cluster database 482
Modifica della password dell'utente principale 484
Impostazioni disponibili 487
Impostazioni che non si applicano ai cluster di database Aurora 524
Impostazioni che non si applicano alle istanze database Aurora 525
Aggiunta di repliche di Aurora 527
Auto Scaling con repliche Aurora 534

Aggiungere una politica di scalabilità automatica ... 540
Modifica di una politica di scalabilità automatica ... 553
Eliminazione di una politica di scalabilità automatica ... 556
Gestione delle prestazioni e del dimensionamento ... 559
Dimensionamento dello storage ... 559
Dimensionamento delle istanze ... 566
Dimensionamento della lettura ... 567
Gestione delle connessioni ... 567
Gestione dei piani di esecuzione delle query ... 568
Clonazione di un volume per un cluster database Aurora ... 569
Panoramica sulla clonazione Aurora ... 569
Limitazioni della clonazione Aurora ... 570
Come funziona la clonazione Aurora ... 571
Creazione di un clone Aurora ... 575
Clonazione incrociata VPC ... 586
Clonazione tra più account ... 604
Integrazione con i servizi AWS ... 621
Aurora Mia SQL ... 621
Aurora Postgre SQL ... 621
Manutenzione di un cluster database Aurora ... 623
Panoramica degli aggiornamenti di manutenzione dei cluster di database delle ... 623
Visualizzazione della manutenzione in sospeso ... 625
Scelta della frequenza degli aggiornamenti di manutenzione di Aurora MySQL ... 627
Maintenance window (Finestra di manutenzione) ... 628
Applicazione di aggiornamenti ... 634
Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora ... 636
Aggiornamenti del sistema operativo ... 641
Riavvio di un cluster o di un'istanza Aurora DB ... 650
Riavvio di un'istanza database in un cluster Aurora ... 651
Riavvio di un cluster Aurora con disponibilità di lettura ... 652
Riavvio di un cluster Aurora senza disponibilità di lettura ... 654
Verifica del tempo di attività per i cluster e le istanze Aurora ... 655
Esempi di operazioni di riavvio Aurora ... 658
Esecuzione del failover di un cluster Aurora DB ... 675
Eliminazione di cluster e istanze di Aurora ... 677
Eliminazione di un cluster database Aurora ... 677

Protezione dall'eliminazione per i cluster Aurora ... 685
Eliminazione di un cluster Aurora arrestato ... 686
Eliminazione di Aurora I SQL miei cluster che sono repliche lette ... 686
Lo snapshot finale durante l'eliminazione di un cluster ... 686
Eliminazione di un'istanza database da un cluster database Aurora ... 687
Taggare RDSAurora e risorse ... 690
Perché usare i RDS tag? ... 691
Come funzionano i RDS tag ... 692
Best practice ... 695
Copia di tag in snapshot di cluster database ... 696
Aggiungere ed eliminare tag in Amazon RDS ... 696
Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare ... 701
ARNsin Amazon RDS ... 705
Costruzione di un ARN ... 705
Ottenere un esistente ARN ... 713
Aggiornamenti di Aurora ... 717
Identificare la versione Amazon Aurora ... 717
Estensione del supporto di RDS ... 719
Panoramica del supporto RDS Extended ... 720
RDSCosti di Extended Support ... 721
Evitare gli addebiti per RDS Extended Support ... 721
Versioni con RDS Extended Support ... 721
Responsabilità con RDS Extended Support ... 722
Responsabilità di Amazon Aurora ... 722
Le tue responsabilità ... 722
Creazione Aurora DB o di un cluster globale ... 723
RDSComportamento dell'Extended Support ... 724
Considerazioni per l'RDSExtended Support ... 724
Crea Aurora DB o un cluster RDS globale con Extended Support ... 725
Visualizzazione della registrazione a RDS Extended Support ... 726
Visualizzazione delle date di assistenza ... 729
Ripristino di Aurora DB o di un cluster globale ... 731
RDSComportamento dell'Extended Support ... 732
Considerazioni per l'RDSExtended Support ... 732
Ripristina del cluster Aurora DB o un cluster RDS globale con Extended Support ... 733
Utilizzo delle implementazioni blu/verde per gli aggiornamenti del database ... 735

Panoramica delle implementazioni Blue/Green ... 736
Disponibilità di regioni e versioni ... 737
Vantaggi ... 737
Flusso di lavoro ... 737
Autorizzazione di accesso ... 743
Considerazioni e limitazioni ... 744
Best practice ... 751
Creazione di un'implementazione blu/verde ... 755
Preparazione di una implementazione blu/verde ... 756
Specifica delle modifiche ... 757
Creazione di un'implementazione blu/verde ... 758
Impostazioni disponibili ... 760
Visualizzazione di un'implementazione blu/verde ... 762
Switchover di un'implementazione blu/verde ... 766
Timeout dello switchover ... 767
Guardrail dello switchover ... 767
Azioni dello switchover ... 768
Best practice per lo switchover ... 769
Verifica CloudWatch delle metriche prima del passaggio al digitale ... 771
Monitoraggio del ritardo nella replica prima del passaggio al digitale ... 771
Switchover di un'implementazione blu/verde ... 772
Dopo lo switchover ... 774
Eliminazione di un'implementazione blu/verde ... 776
Backup e ripristino di un cluster DB Aurora ... 781
Panoramica di backup e ripristino ... 782
Backup ... 782
Finestra di backup ... 783
Ripristino dei dati ... 786
Clonazione dei database ... 787
Backtrack ... 787
Mantenimento dei backup automatici ... 788
Periodo di conservazione ... 788
Costi di retention ... 789
Impedire l'eliminazione automatica dei backup ... 789
Limitazioni ... 789
Visualizzazione dei backup automatici conservati ... 790

Eliminazione dei backup automatici mantenuti ... 790
Storage di backup ... 793
Storage di backup automatici ... 793
Archiviazione delle istantanee ... 794
CloudWatch parametri per lo storage di backup ... 794
Calcolo dell'utilizzo dello storage dei backup ... 795
FAQs ... 796
Creazione di uno snapshot del cluster database ... 799
Determinare se lo snapshot è disponibile ... 801
Ripristino da uno snapshot cluster database ... 802
Gruppi di parametri ... 803
Gruppi di sicurezza ... 803
Considerazioni Aurora ... 803
Ripristino da uno snapshot ... 804
Copia di snapshot del cluster DB ... 807
Copiare uno snapshot del cluster DB con AWS Management Console ... 807
Limitazioni ... 809
Considerazioni ... 810
Copia di uno snapshot del cluster DB non crittografato ... 812
Copia di uno snapshot crittografato del cluster DB ... 815
Copia di una snapshot cluster database tra account ... 819
Condivisione di uno snapshot cluster database ... 823
Condivisione di uno snapshot ... 824
Condivisione di snapshot pubblici ... 827
Condivisione di snapshot crittografati ... 829
Interruzione della condivisione delle istantanee ... 833
Esportazione dei dati del cluster database in Amazon S3 ... 835
Considerazioni ... 837
Configurazione dell'accesso a un bucket S3 ... 839
Creazione di attività di esportazione di cluster DB ... 843
Monitoraggio delle esportazioni del cluster database ... 846
Annullamento di un'esportazione del cluster database ... 849
Risoluzione dei problemi ... 850
Esportazione dei dati dello snapshot del cluster di database in Amazon S3 ... 853
Considerazioni ... 855
Configurazione dell'accesso a un bucket S3 ... 868

Creazione di attività di esportazione di istantanee 873
Monitoraggio delle esportazioni di snapshot 877
Annullamento di un'esportazione di snapshot 879
Andamento delle esportazioni in Aurora My SQL 881
Risoluzione dei problemi 881
Point-in-time recupero 884
Ripristino di un cluster DB in un punto temporale 885
oint-in-time Ripristino P da un backup automatizzato mantenuto 888
oint-in-time Ripristino P utilizzando AWS Backup 891
Eliminazione di una snapshot del cluster di database 897
Eliminazione di una snapshot del cluster di database 897
Tutorial: Ripristino di cluster database da uno snapshot 899
Ripristino di un cluster database mediante la console 899
Ripristino di un cluster DB utilizzando il AWS CLI 904
Monitoraggio dei parametri in un cluster di database Aurora 911
Piano di monitoraggio 911
Baseline delle prestazioni 912
Linee guida per le prestazioni 912
Strumenti di monitoraggio 914
Strumenti di monitoraggio automatici 914
Strumenti di monitoraggio manuali 916
Visualizzazione dello stato dell' del cluster 918
Visualizzazione di un cluster di database 919
Visualizzazione dello stato del cluster del DB 925
Visualizzazione dello stato dell'istanza database di in un cluster Aurora 929
..... 936
Visualizzazione dei suggerimenti 938
Applicazione dei consigli 946
Ignorare i consigli 952
Modificare i consigli ignorati in attivi 954
Riferimento ai consigli 955
Visualizzazione delle metriche nella console Amazon RDS 976
Visualizzazione del pannello di controllo Performance Insights 977
Scelta della nuova visualizzazione di monitoraggio dalla scheda Monitoraggio 978
Scelta della nuova visualizzazione di monitoraggio dalla pagina Performance Insights 980
Creazione di una dashboard personalizzata 981

Scelta della dashboard preconfigurata 985
Monitoraggio di Aurora con CloudWatch 987
Panoramica di Amazon Aurora e Amazon CloudWatch 988
Visualizzazione delle metriche CloudWatch 990
Esportazione delle metriche di Performance Insights in CloudWatch 995
Creazione di allarmi CloudWatch 1001
Monitoraggio con Database Insights 1003
Prezzi 1003
Supporto per motori, regioni e classi di istanze 1003
Attivazione della modalità Avanzata 1006
Attivazione della modalità Standard 1010
Monitora le query lente 1015
Considerazioni 1016
Monitoraggio del carico DB con Performance Insights 1018
Panoramica su Performance Insights 1019
Attivazione e disattivazione di Performance Insights 1031
Schema delle prestazioni per Aurora MySQL 1038
Policy di Performance Insights 1044
Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights 1057
Visualizzazione dei consigli proattivi di Performance Insights 1091
Recupero delle metriche con Performance Insights API 1094
Registrazione delle chiamate Performance Insights utilizzando AWS CloudTrail 1119
VPCendpoint ()AWS PrivateLink 1121
Analisi delle prestazioni con Guru per DevOps RDS 1125
Vantaggi di DevOps Guru per RDS 1126
Come funziona DevOps Guru for RDS 1127
Configurare Guru per DevOps RDS 1128
Monitoraggio del sistema operativo con il monitoraggio avanzato 1137
Panoramica sul monitoraggio avanzato 1137
Configurare e abilitare il monitoraggio avanzato 1139
Visualizzazione delle metriche del sistema operativo nella console RDS 1146
Visualizzazione delle metriche del sistema operativo tramite i registri CloudWatch 1148
Riferimento alle metriche Aurora 1150
CloudWatch metriche per Aurora 1150
CloudWatch dimensioni per Aurora 1191
Disponibilità delle metriche Aurora nella console Amazon RDS 1192

CloudWatch metriche per Performance Insights 1196
Parametri contatore per Performance Insights 1199
Statistiche SQL per Performance Insights 1232
Metriche OS nel monitoraggio avanzato 1241
Monitoraggio di eventi, registri e flussi di attività di database 1250
Visualizzazione di log, eventi e stream nella console Amazon RDS 1251
Monitoraggio degli eventi Aurora 1255
..... 1255
Visualizzazione degli RDS eventi Amazon 1257
Utilizzo delle notifiche di RDS eventi di Amazon 1261
Creazione di una regola che si attiva su un evento Amazon Aurora 1287
Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora 1291
Monitoraggio dei log di Aurora 1326
Visualizzazione ed elenco dei file di log del database 1326
Download di un file di log di database 1328
Controllo di un file di log di database 1329
Pubblicazione su Logs CloudWatch 1331
Lettura del contenuto del file di registro utilizzando REST 1333
File di log del database MySQL 1335
File di log del database PostgreSQL 1345
Monitoraggio delle chiamate Aurora APIin ingresso CloudTrail 1356
CloudTrail 1356
Amazon Aurora Voci dei file di RDS log di Aurora 1357
Monitoraggio di Aurora tramite i flussi di attività del database 1361
Panoramica 1361
Prerequisiti di rete Aurora MySQL 1365
Avvio di un flusso di attività di database 1367
Recupero dello stato del flusso di attività 1371
Arresto di un flusso di attività di database 1372
Monitoraggio dei flussi di attività 1374
IAMesempi di policy per i flussi di attività 1411
Monitoraggio delle minacce con GuardDuty RDS Protection 1414
Utilizzo di Aurora MySQL 1416
Panoramica di Aurora MySQL 1417
Miglioramenti alle prestazioni di Amazon Aurora MySQL 1417
Aurora MySQL e dati spaziali 1418

Aurora My SQL versione 3 compatibile con My 8.0 SQL 1419
Aurora My SQL versione 2 compatibile con My 5.7 SQL 1453
Sicurezza con Aurora MySQL 1456
Privilegi dell'utente master con Aurora MySQL. 1457
Connessioni TLS 1458
Aggiornamento delle applicazioni per i nuovi certificati TLS 1467
Determinazione se un'applicazione si connette al cluster databse Aurora MySQL utilizzando SSL 1468
Determinare se un client richiede la verifica del certificato per la connessione 1468
Aggiornare l'archivio di trust delle applicazioni 1470
Codice Java di esempio per stabilire connessioni TLS 1470
Utilizzo dell'autenticazione Kerberos per Aurora My SQL 1472
Panoramica dell'autenticazione Kerberos per Aurora My SQL 1473
Limitazioni 1474
Configurazione dell'autenticazione Kerberos per Aurora My SQL 1475
Connessione ad Aurora MySQL con l'autenticazione Kerberos 1486
Gestione di un cluster di database in un dominio 1490
Migrazione dei dati su Aurora My SQL 1492
Migrazione da un database MySQL esterno a Aurora MySQL 1497
Migrazione da un'istanza database MySQL a Aurora MySQL 1525
Gestione di Aurora MySQL 1551
Gestione delle prestazioni e del dimensionamento per Amazon Aurora MySQL 1551
Backtrack di un cluster database 1563
Test di Amazon Aurora My SQL using Fault Injection 1584
Alterazione delle tabelle in Amazon Aurora mediante Fast DDL 1589
Visualizzazione dello stato del volume per un cluster di database Aurora 1596
Ottimizzazione di Aurora MySQL 1598
Concetti essenziali per la regolazione di Aurora MySQL 1598
Regolazione di Aurora MySQL con eventi di attesa 1602
Regolazione di Aurora MySQL con stati del thread 1655
Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru .. 1663
Interrogazione parallela per Aurora My SQL 1669
Panoramica delle query in parallelo 1669
Creazione di un cluster abilitato per le query in parallelo 1673
Attivazione e disattivazione della query parallela 1678
Ottimizzazione delle interrogazioni parallele 1681

Verifica dell'utilizzo della funzione di query in parallelo 1686
Monitoraggio della funzionalità di query in parallelo 1690
SQLcostrutti per interrogazioni parallele 1698
Audit avanzato con Aurora My SQL 1720
Abilitazione dell'audit avanzato 1720
Visualizzazione dei log di audit 1724
Dettagli dei log di audit 1724
Replica con Aurora MySQL 1726
Repliche di Aurora 1726
Opzioni di replica 1728
Prestazioni di replica 1729
Riavvio senza tempi di inattività () ZDR 1729
Filtri di replica 1732
Monitoraggio della replica 1739
Replica in più regioni 1740
Replica di log binari (binlog) 1756
GTIDreplica basata 1799
Inoltro di scrittura locale 1806
Abilitazione dell'inoltro di scrittura locale 1807
Verifica se l'inoltro di scrittura è abilitato un cluster database 1809
Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura 1810
Livelli di isolamento per l'inoltro di scrittura 1812
Consistenza di lettura 1812
Esecuzione di istruzioni a più parti con inoltro scrittura 1816
Transazioni con inoltro di scrittura 1817
Parametri di configurazione per l'inoltro di scrittura 1817
Parametri per l'inoltro di scrittura 1819
Identificazione delle transazioni e delle query inoltrate 1823
Integrazione di Aurora SQL My con i servizi AWS 1826
Autorizzazione di Aurora MySQL per accedere ai servizi AWS 1826
Caricamento di dati da file di testo in Amazon S3 1845
Salvataggio dei dati dei file di testo in Amazon S3 1860
Richiamo di una funzione Lambda da Aurora My SQL 1871
Pubblicazione dei log di Aurora MySQL su Logs CloudWatch 1883
Modalità di laboratorio per Aurora MySQL 1889
Caratteristiche della modalità di laboratorio per Aurora 1889

Le migliori pratiche con Aurora My SQL 1891
Determinazione dell'istanza database a cui si è connessi 1892
Le migliori pratiche per prestazioni e scalabilità 1892
Le migliori pratiche per l'alta disponibilità 1902
Consigli per le mie funzionalità SQL 1904
Valutazione dell'utilizzo delle istanze DB per Aurora My SQL con i parametri di Amazon CloudWatch 1912
Risoluzione dei problemi relativi a Aurora My performance SQL 1914
AWS opzioni di monitoraggio 1914
Le cause più comuni dei problemi di prestazioni del DB 1915
Risoluzione dei problemi relativi al carico di lavoro 1916
Registrazione per Aurora MySQL 1946
Risoluzione dei problemi di connessione al database 1949
Risoluzione dei problemi delle prestazioni delle query 1965
Aurora La mia referenza SQL 1970
Parametri di configurazione 1970
Variabili di stato globali 2044
Eventi di attesa 2061
Stati del thread 2067
Livelli di isolamento 2072
Suggerimenti 2079
Riferimento alla procedura memorizzata 2083
Tabelle information_schema 2131
Aurora I miei aggiornamenti SQL 2140
Controllo dei numeri di versione 2140
Supporto a lungo termine e versioni beta 2143
Preparazione per la fine del ciclo di vita di Aurora My SQL version 2 2145
Preparazione per Aurora My SQL version 1 end of life 2150
Aggiornamento dei cluster Amazon Aurora My DB SQL 2154
Aggiornamenti e correzioni del motore di database per Amazon Aurora MySQL 2283
Utilizzo di Aurora PostgreSQL 2284
L'ambiente di anteprima del database 2286
Tipi di classi di istanze DB supportati 2286
Funzionalità non supportate nell'ambiente di anteprima 2287
Creazione di un nuovo cluster DB nell'ambiente di anteprima 2288
Postgre SQL versione 17 nell'ambiente Database Preview 2289

Sicurezza con Aurora PostgreSQL 2290
Informazioni su ruoli e autorizzazioni di PostgreSQL 2291
Sicurezza dei dati Aurora PostgreSQL con SSL/TLS 2308
Aggiornamento delle applicazioni per i nuovi certificati SSL/TLS 2321
Determinare se un'applicazione si connette al cluster DB Aurora PostgreSQL utilizzando SSL 2322
Determinare se un client richiede la verifica del certificato per la connessione 2322
Aggiornare l'archivio di trust delle applicazioni 2323
Utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni 2324
Utilizzo di Autenticazione Kerberos 2325
Disponibilità di regioni e versioni 2326
Panoramica dell'autenticazione Kerberos 2326
Configurazione 2327
Gestione di un cluster Aurora PostgreSQL DB (RDS per PostgreSQL) per un'istanza DB PostgreSQL in un Directory 2342
Connessione con Autenticazione Kerberos 2343
Utilizzo dei gruppi di sicurezza AD per il controllo degli accessi di Aurora SQL Postgree ... 2347
Migrazione dei dati su Aurora Postgre SQL 2359
Migrazione di un'istanza DB per Postgre tramite un'istantanea RDS SQL 2360
Migrazione di un'istanza database RDS for PostgreSQL tramite una replica di lettura Aurora 2367
Ottimizzazione delle prestazioni delle query in Aurora PostgreSQL 2381
Prestazioni delle query migliorate con Letture ottimizzate per Aurora 2382
Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL 2388
Miglioramento delle prestazioni delle query utilizzando adaptive join 2395
Utilizzo di tabelle non registrate in Aurora PostgreSQL 2397
Creazione di tabelle non registrate 2398
Conversione di tabelle non registrate in tabelle registrate 2398
Tabelle non registrate e replica logica 2398
Funzionamento di PostgreSQL Autovacuum 2399
Allocazione di memoria per il vacuum 2400
Riduzione della probabilità che si verifichi il wraparound dell’ID delle transazioni 2401
Determinare se le tabelle nel database devono essere sottoposte a vacuum 2403
Determinare quali tabelle sono attualmente idonee per l'Autovacuum 2404
Determinare se l'Auotvacuum è attualmente in esecuzione e per quanto tempo 2405
Esecuzione di un congelamento manuale del vacuum 2407

Indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione 2409
Gestione di autovacuum con indici di grandi dimensioni 2410
Altri parametri che influenzano l'autovacuum 2414
Impostazione dei parametri di autovacuum a livello tabella 2415
Registrazione delle attività di autovacuum e vacuum 2415
Comprensione del comportamento di autovacuum con database non validi 2417
Identificazione degli aspirapolveri 2419
Utilizzo di Babelfish per Aurora PostgreSQL 2444
Limitazioni di Babelfish 2446
Comprendere l'architettura e la configurazione di Babelfish 2447
Creazione di un cluster Babelfish per Aurora Postgre DB SQL 2494
Migrazione di un database Server su Babelfish SQL 2505
Autenticazione del database con Babelfish per Aurora Postgre SQL 2515
Connessione a un cluster database Babelfish 2533
Utilizzo di Babelfish 2546
Risoluzione dei problemi relativi a Babelfish 2625
Disattivazione di Babelfish 2627
Gestione degli aggiornamenti delle versioni di Babelfish 2628
Informazioni di riferimento su Babelfish 2650
Prestazioni e scalabilità per Aurora PostgreSQL 2706
Dimensionamento delle istanze database Aurora PostgreSQL 2707
Numero massimo connessioni 2708
Limiti di storage temporaneo 2709
Huge Pages per Aurora PostgreSQL 2713
Test di Amazon Aurora PostgreSQL mediante query Fault Injection 2714
Visualizzazione dello stato del volume per un cluster di database Aurora 2719
Specificare il disco RAM per stats_temp_directory 2721
Gestione dei file temporanei con PostgreSQL 2722
Sintonizzazione degli eventi di attesa per Aurora PostgreSQL 2728
Concetti essenziali per l'ottimizzazione di Aurora PostgreSQL 2729
Eventi di attesa Aurora PostgreSQL 2734
Cliente: ClientRead 2737
Cliente: ClientWrite 2741
CPU 2743
IO:buffileRead e IO:buffileWrite 2749
IO: DataFileRead 2758

IO: XactSync ... 2769
IPC:DamRecordTxAck ... 2771
IPC: eventi di attesa paralleli ... 2772
Lock:advisory ... 2779
Lock:extend ... 2782
Lock:Relation ... 2785
Lock:transactionid ... 2792
Lock:tuple ... 2795
LWLock:buffer_content (BufferContent) ... 2800
LWLock:buffer_mapping ... 2802
LWLock:bufferIO (IPC: bufferio) ... 2805
LWLock:lock_manager ... 2807
LWLock:MultiXact ... 2812
Timeout: PG Sleep ... 2816
Ottimizzazione con informazioni proattive su Amazon SQL Guru DevOps ... 2817
Il database ha una connessione di transazione inattiva da molto tempo ... 2817
Best practice con Aurora PostgreSQL ... 2821
Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per le istanze database Aurora PostgreSQL ... 2821
Diagnosi delle dimensioni della tabella e dell'indice ... 2822
Migliore gestione della memoria in Aurora PostgreSQL ... 2826
Failover rapido ... 2828
Ripristino rapido dopo il failover ... 2840
Gestione dell’abbandono della connessione ... 2847
... 2856
Analizza l'utilizzo delle risorse con i parametri CloudWatch ... 2865
Utilizzo della replica logica per un aggiornamento a una versione principale ... 2869
Risoluzione dei problemi di storage in Aurora PostgreSQL ... 2878
Replica con Aurora PostgreSQL ... 2879
Repliche di Aurora ... 2880
Miglioramento della disponibilità di lettura delle repliche Aurora ... 2881
Monitoraggio della replica 2883
Panoramica della replica logica ... 2883
Configurazione della replica logica ... 2885
Disattivazione della replica logica ... 2887
Monitoraggio della cache di scrittura e degli slot logici ... 2888

Esempio: Utilizzo della replica logica ... 2890
Esempio: replica logica con SQL Aurora Postgre e AWS DMS ... 2892
Inoltro di scrittura locale in Aurora PostgreSQL ... 2895
Limitazioni e considerazioni sull'inoltro di scrittura locale in Aurora PostgreSQL ... 2896
Configurazione di Aurora SQL Postgre per l'inoltro di scrittura locale ... 2897
Utilizzo dell'inoltro di scrittura locale per Aurora PostgreSQL ... 2900
Monitoraggio dell'inoltro di scrittura locale in Aurora Postgre SQL ... 2904
Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock ... 2910
Prerequisiti ... 2911
Preparazione di Aurora PostgreSQL come Knowledge Base ... 2912
Creazione di una Knowledge Base nella console Bedrock ... 2915
Crea rapidamente una SQL Knowledge Base Aurora Postgre per Amazon Bedrock ... 2916
Integrazione di Aurora SQL Postgre con i servizi AWS ... 2919
Importazione di dati da Amazon S3 in Aurora PostgreSQL ... 2920
Esportazione di dati PostgreSQL in Amazon S3 ... 2940
Richiamare una funzione Lambda da Aurora PostgreSQL ... 2957
Pubblicazione dei log di Aurora PostgreSQL in Logs CloudWatch ... 2973
Monitoraggio dei piani di esecuzione delle query e della memoria di picco per Aurora Postgre SQL ... 2985
Accesso ai piani di esecuzione delle query e alla memoria di picco utilizzando le funzioni Aurora ... 2985
Riferimento ai parametri per i piani di esecuzione delle query di Aurora SQL Postgre ... 2987
Gestione dei piani di esecuzione delle query per Aurora PostgreSQL ... 2991
Panoramica della gestione del piano di query per Aurora PostgreSQL ... 2992
Best practice per la gestione del piano di query Aurora PostgreSQL ... 3000
Gestione del piano di query ... 3003
Acquisizione dei piani di esecuzione Aurora PostgreSQL ... 3005
Utilizzo dei piani gestiti per Aurora PostgreSQL ... 3008
Esame dei piani di query Aurora PostgreSQL nella vista dba_plans ... 3013
Miglioramento dei piani ... 3014
Eliminazione dei piani ... 3018
Esportazione e importazione di piani gestiti ... 3020
Riferimento ai parametri ... 3021
Informazioni di riferimento sulle funzioni ... 3029
Riferimento per la visualizzazione apg_plan_mgmt.dba_plans ... 3040
Funzionalità avanzate della gestione del piano di query ... 3045

Utilizzo di estensioni e wrapper di dati esterni ... 3060
Utilizzo del supporto delegato delle estensioni di Amazon Aurora per Postgre SQL ... 3061
Gestione di oggetti di grandi dimensioni in maniera più efficiente con il modulo lo ... 3075
Gestione dei dati spaziali con Post GIS ... 3079
Gestione delle partizioni con l'estensione pg_partman ... 3089
Pianificazione della manutenzione con l'estensione pg_cron ... 3095
Utilizzo pgAudit per registrare l'attività del database ... 3105
Utilizzo di pglogical per sincronizzare i dati ... 3118
Wrapper di dati esterni supportati in Amazon Aurora Postgre Amazon for SQL ... 3132
Utilizzo di Trusted Language Extensions per PostgreSQL ... 3148
Terminologia ... 3149
Requisiti per l'utilizzo di Trusted Language Extensions ... 3150
Impostazione di Trusted Language Extensions ... 3153
Panoramica di Trusted Language Extensions ... 3157
Creazione di TLE estensioni ... 3158
Eliminare le TLE estensioni da un database ... 3163
Disinstallazione di Trusted Language Extensions ... 3164
Utilizzo di hook PostgreSQL con le estensioni TLE ... 3165
Riferimento funzionale per Trusted Language Extensions ... 3171
Riferimento sugli hook per Trusted Language Extensions ... 3185
Riferimento Aurora PostgreSQL ... 3189
Fascicolazioni Aurora PostgreSQL per EBCDIC e altre migrazioni del mainframe ... 3189
Collazioni supportate in Aurora Postgre SQL ... 3191
Informazioni di riferimento sulle funzioni Aurora PostgreSQL ... 3192
Aurora PostgreSQL parametri ... 3249
Eventi di attesa Aurora PostgreSQL ... 3311
Aggiornamenti Aurora Postger SQL ... 3341
Identificazione delle versioni di Amazon Aurora Postgre SQL ... 3341
Pubblicazioni di Aurora Postgre SQL ... 3343
Versioni di estensione per Aurora Postgre SQL ... 3344
Aggiornamento dei cluster database Amazon Aurora PostgreSQL ... 3344
Uso di versioni con supporto a lungo termine (Long-Term Support, LTS) ... 3432
Utilizzo del database Aurora SQL Postgre Limitless ... 3435
Architettura di database senza limiti ... 3436
Termini chiave ... 3438
Tipi di tabelle ... 3440

Fatturazione 3440
Guida introduttiva a Limitless Database 3441
Requisiti e considerazioni illimitati per i database 3442
Requisiti 3442
Considerazioni 3443
Funzionalità non supportate 3446
Prerequisiti per l'utilizzo di Limitless Database 3447
Abilitazione delle operazioni dei gruppi condivisi DB 3447
Creazione di un cluster DB che utilizza Limitless Database 3448
Correlazione della capacità massima con router e shard 3448
Creazione del cluster DB 3451
Lavorare con gruppi di shard DB 3461
Connessione al cluster Aurora PostgreSQL Limitless Database DB 3462
Individuazione del numero di router e shard in un gruppo di shard DB 3463
Descrizione dei gruppi di shard DB 3463
Riavvio di un gruppo di shard DB 3464
Modifica della capacità di un gruppo di shard DB 3465
Dividere uno shard 3467
Aggiungere un router 3473
Eliminazione di un gruppo di shard DB 3477
Aggiungere un gruppo di shard DB a un cluster Limitless Database DB esistente 3478
Creazione di tabelle di database illimitate 3483
Creazione di tabelle illimitate utilizzando variabili 3484
Conversione di tabelle standard in tabelle illimitate 3489
Schemi di esempio 3493
Caricamento dei dati in Limitless Database 3495
Utilizzo del comando con Limitless Database COPY 3497
Utilizzo dell'utilità di caricamento dati Limitless Database 3499
Interrogazione di Limitless Database 3528
Query a frammento singolo 3529
Interrogazioni distribuite 3539
Tracciamento distribuito delle interrogazioni 3540
Deadlock distribuiti 3548
Gestione di un database illimitato 3551
Considerazioni sulle dimensioni di database e tabelle 3551
Recupero dello spazio di archiviazione mediante aspirazione 3552

Monitoraggio di un database illimitato 3556
Monitoraggio di un database illimitato con CloudWatch 3557
Monitoraggio di database illimitati con Database Insights 3562
Monitoraggio di database illimitati con log CloudWatch 3574
Monitoraggio di database illimitati con monitoraggio avanzato 3575
Monitoraggio di database illimitati con Performance Insights 3576
Monitoraggio di database illimitati con protezione GuardDuty RDS 3584
Funzioni e visualizzazioni per Limitless Database 3585
Attendi gli eventi per Limitless Database 3611
Backup e ripristino di Limitless Database 3624
Backup di un cluster DB che utilizza SQL Aurora Postgre Limitless Database 3624
Ripristino di un cluster DB che utilizza SQL Aurora Postgre Limitless Database 3626
Le utilità di SQL backup e ripristino di Postgre non sono supportate 3629
Aggiornamento di Limitless Database 3630
Aggiornamento dei cluster DB che utilizzano Aurora PostgreSQL Limitless Database 3630
Riferimento illimitato al database 3632
Comandi DDL 3632
Limitazioni e informazioni DDL 3641
Comandi DML 3679
DMLlimitazioni e informazioni 3682
Variables 3686
Parametri del cluster DB 3687
Utilizzo di Aurora Global Database 3688
Panoramica di Aurora Global Database 3688
Vantaggi del database globale di Amazon Aurora 3690
Disponibilità di regioni e versioni 3691
Limitazioni di Aurora Global Database 3691
Guida introduttiva ad Aurora Global Database 3693
Requisiti di configurazione 3694
Creazione di un database globale 3696
Aggiungere un cluster secondario 3710
Creazione di un cluster secondario headless 3715
Creazione di un database globale da uno snapshot 3718
Gestione di un database globale Aurora 3719
Modifica di un database globale Aurora 3720
Modifica dei parametri del database globale 3722

Rimozione di un cluster da un database globale Aurora ... 3722
Eliminazione di un database globale Aurora ... 3726
Etichettatura per Aurora Global Database ... 3728
Connessione a Aurora Global Database ... 3733
Scelta dell'endpoint che soddisfa le esigenze dell'applicazione ... 3733
Visualizzazione degli endpoint di un database globale ... 3734
Considerazioni sull'utilizzo degli endpoint Global writer ... 3738
Utilizzo dell'inoltro di scrittura in un database globale Aurora ... 3739
Utilizzo dell'inoltro di scrittura in Aurora MySQL ... 3739
Utilizzo dell'inoltro di scrittura in Aurora PostgreSQL ... 3760
Switchover o failover per Aurora Global Database ... 3775
Pianificazione per BCDR ... 3776
Esecuzione degli switchover ... 3777
Ripristino dopo un'interruzione non pianificata ... 3784
Gestione RPOs (Aurora PostgreSQL) ... 3795
Resilienza interregionale per i cluster secondari ... 3801
Monitoraggio di un database globale Aurora ... 3801
Monitoraggio di un database globale Aurora con Performance Insights ... 3803
Monitoraggio dei database globali Aurora con i flussi di attività di database ... 3804
Monitoraggio dei database globali SQL basati su Aurora My ... 3804
Monitoraggio dei database globali basati su Aurora SQL Postgre ... 3808
Utilizzo dei database globali Aurora con altri servizi AWS ... 3811
Aggiornamento di un database globale Amazon Aurora ... 3812
Aggiornamenti di una versione principale ... 3813
Aggiornamenti della versione secondaria ... 3814
Server proxy per Amazon RDS ... 3817
Disponibilità di regioni e versioni ... 3818
Quote e limiti ... 3818
Limitazioni di MySQL ... 3820
Limitazioni di PostgreSQL ... 3821
Pianificazione sull'utilizzo di RDS Proxy ... 3822
RDSConcetti e terminologia relativi ai proxy ... 3823
Panoramica dei concetti relativi al proxy RDS ... 3824
Pooling di connessioni ... 3825
Sicurezza ... 3826
Failover ... 3828

Transazioni ... 3829
Guida introduttiva a RDS Proxy ... 3830
Configura una rete proxy ... 3830
Impostazione delle credenziali del database ... 3834
Configurazione dell'autenticazione IAM ... 3838
Creazione di un proxy ... 3841
Visualizzazione di un proxy ... 3851
Collegamento tramite RDS Proxy ... 3853
Gestione di un RDS proxy ... 3856
Modifica di un RDS Proxy ... 3857
Aggiunta di un utente di database ... 3864
Considerazioni sulla connessione del proxy RDS ... 3865
Evita di bloccare RDS Proxy ... 3870
Eliminazione di un proxy RDS ... 3875
Utilizzo degli endpoint RDS Proxy ... 3876
Panoramica degli endpoint proxy ... 3876
Limitazioni per gli endpoint proxy ... 3877
Utilizzo degli endpoint di lettura con cluster Aurora ... 3878
Accesso ai database Aurora RDS VPCs ... 3883
Creazione di un endpoint proxy ... 3884
Visualizzazione degli endpoint proxy ... 3887
Modifica di un endpoint proxy ... 3888
Eliminazione di un endpoint proxy ... 3889
Monitoraggio del proxy RDS con CloudWatch ... 3891
Lavorare con gli eventi RDS Proxy ... 3900
RDSEventi proxy ... 3900
RDSEsempi di proxy ... 3903
Risoluzione dei problemi RDS Proxy ... 3906
Verifica della connettività a un proxy ... 3906
Problemi e soluzioni comuni ... 3908
Risoluzione dei problemi per RDS for MySQL ... 3909
Risoluzione dei problemi per RDS per PostgreSQL ... 3911
Utilizzo di RDS Proxy con AWS CloudFormation ... 3917
Utilizzo del Server proxy per RDS con i database globali Aurora ... 3918
Limitazioni di Server proxy per RDS con i database globali ... 3919
Come funzionano gli endpoint Server proxy per RDS con i database globali ... 3919

Integrazioni Zero-ETL 3921
Vantaggi 3922
Concetti chiave 3923
Limitazioni 3923
Limitazioni generali 3924
Aurora MySQL 3925
Limitazioni di Aurora PostgreSQL 3925
Limitazioni di Amazon Redshift 3926
Quote 3926
Regioni supportate 3926
Guida introduttiva alle integrazioni Zero-ETL 3927
Fase 1: creazione di un gruppo di parametri del cluster DB personalizzato 3927
Passaggio 2: selezionare o creare un cluster DB del di origine 3928
Fase 3: creazione di un data warehouse Amazon Redshift di destinazione 3929
Configurare un'integrazione utilizzando AWS SDKs 3930
Passaggi successivi 3940
Creazione di integrazioni Zero-ETL 3940
Prerequisiti 3941
Autorizzazioni richieste 3941
Creazione di integrazioni Zero-ETL 3944
Integrazioni crittografate 3948
Passaggi successivi 3950
Filtraggio dei dati per integrazioni zero-ETL 3950
Formato di un filtro dati 3951
Logica dei filtri 3954
Precedenza dei filtri 3954
..... 3955
Esempi di Aurora PostgreSQL 3956
Aggiungere filtri di dati 3957
Rimozione dei filtri di dati 3959
Aggiunta di dati ed esecuzione di query 3959
Creazione di un database di destinazione in Amazon Redshift 3960
..... 3960
Interrogazione dei dati di in Amazon Redshift 3961
Differenze dei tipi di dati 3962
Operazioni DDL per Aurora PostgreSQL 3975

Visualizzazione e monitoraggio delle integrazioni zero ETL ... 3978
Visualizzazione delle integrazioni ... 3978
Monitoraggio tramite tabelle di sistema ... 3980
Monitoraggio con EventBridge ... 3981
Modifica delle integrazioni zero ETL ... 3981
Eliminazione delle integrazioni zero ETL ... 3982
Risoluzione dei problemi relativi alle integrazioni zero ETL ... 3984
Non riesco a creare un'integrazione zero ETL ... 3984
La mia integrazione è bloccata in uno stato di `Syncing` ... 3985
Le mie tabelle non si replicano su Amazon Redshift ... 3985
Una o più tabelle Amazon Redshift richiedono una risincronizzazione ... 3986
DDLle operazioni non complete vengono visualizzate in Amazon Redshift ... 3989
Utilizzo Aurora Serverless v2 ... 3990
Aurora Serverless v2 casi d'uso ... 3990
Conversione dei carichi di lavoro con provisioning ... 3992
Vantaggi di Aurora Serverless v2 ... 3993
In che modo Aurora Serverless v2 funzionamento ... 3994
Panoramica ... 3995
Configurazioni cluster ... 3996
Capacità ... 3997
Dimensionamento ... 3999
Elevata disponibilità ... 4002
Storage ... 4003
Parametri di configurazione ... 4004
Requisiti e limitazioni per Aurora Serverless v2 ... 4004
Disponibilità di regioni e versioni ... 4004
Cluster che utilizzano Aurora Serverless v2 deve avere un intervallo di capacità specificato ... 4005
Alcune funzionalità fornite non sono supportate in Aurora Serverless v2 ... 4006
Medio Aurora Serverless v2 gli aspetti sono diversi da Aurora Serverless v1 ... 4006
Creare un Aurora Serverless v2 cluster di database ... 4007
Impostazioni ... 4007
Creare un Aurora Serverless v2 cluster di database ... 4008
Creare un Aurora Serverless v2 scrittore ... 4013
Gestione di Aurora Serverless v2 ... 4014
Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster ... 4015

Verifica del Aurora Serverless v2 Intervallo di capacità ... 4022
Aggiungere un Aurora Serverless v2 reader ... 4023
Conversione da fornito a Aurora Serverless v2 ... 4024
Conversione da Aurora Serverless v2 a fornito ... 4025
Pausa Aurora Serverless v2 Istanze DB ... 4026
Scelta del livello di promozione per un Aurora Serverless v2 reader ... 4026
Utilizzo di TLS/SSL con Aurora Serverless v2 ... 4028
Visualizzazione Aurora Serverless v2 scrittori e lettori ... 4029
Registrazione per Aurora Serverless v2 ... 4030
Prestazioni e dimensionamento per Aurora Serverless v2 ... 4037
Scelta dell'intervallo di capacità ... 4038
Uso di gruppi di parametri per Aurora Serverless v2 ... 4052
Evitare gli errori out-of-memory ... 4058
Metriche importanti CloudWatch ... 4059
Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni ... 4064
Risoluzione dei problemi di capacità di Aurora Serverless v2 ... 4065
Ridimensionamento a 0 ACUs con pausa e ripresa automatiche ... 4067
Panoramica della pausa automatica ... 4067
Prerequisiti e limitazioni ... 4069
Attivazione e disattivazione della pausa automatica ... 4070
Come funziona la pausa automatica ... 4074
Pausa automatica e configurazioni del cluster Aurora ... 4078
Monitoraggio della pausa automatica ... 4082
Risoluzione dei problemi relativi alla pausa automatica ... 4085
Progettazione dell'applicazione per la pausa automatica ... 4087
Migrazione verso Aurora Serverless v2 ... 4088
Utilizzo Aurora Serverless v2 con un cluster esistente ... 4090
Passaggio di un cluster con provisioning ... 4093
Confronto di Aurora Serverless v2 e Aurora Serverless v1 ... 4099
Aggiornamento da Aurora Serverless v1 in Aurora Serverless v2 ... 4110
Migrazione da un database locale a Aurora Serverless v2 ... 4113
Utilizzo Aurora Serverless v1 ... 4114
Disponibilità della regione e della versione per Aurora Serverless v1 ... 4115
Vantaggi di Aurora Serverless v1 ... 4115
Casi d'uso per Aurora Serverless v1 ... 4115

Limitazioni di Aurora Serverless v1 4116
Requisiti di configurazione per Aurora Serverless v1 4119
Utilizzo di TLS/SSL con Aurora Serverless v1 4119
Suite di crittografia supportate per connessioni a Aurora Serverless v1 Cluster database .. 4123
In che modo Aurora Serverless v1 funzionamento 4123
Aurora Serverless v1 architecture 4124
Dimensionamento automatico 4125
Operazione di timeout 4126
Sospendi e riprendi 4128
Determinazione di max_connections 4129
Gruppi di parametri 4132
Registrazione 4135
Manutenzione 4138
Failover 4140
Snapshot 4140
Creare un Aurora Serverless v1 cluster di database 4140
Ripristino di un Aurora Serverless v1 cluster di database 4144
Modificare un Aurora Serverless v1 cluster di database 4148
Modifica della configurazione di dimensionamento 4148
Aggiornamento della versione principale 4151
Conversione da Aurora Serverless v1 a fornito 4153
Considerazioni sulla conversione da un Aurora Serverless v1 Un cluster DB a un cluster predisposto 4155
Dimensionamento Aurora Serverless v1 Capacità del cluster DB manualmente 4156
Visualizzazione Aurora Serverless v1 Cluster database 4159
Monitoraggio Aurora Serverless v1 Cluster DB con CloudWatch 4163
Eliminazione di un Aurora Serverless v1 cluster di database 4163
Aurora Serverless v1 e versioni del motore di database Aurora 4167
Aurora MySQL Serverless 4167
Aurora PostgreSQL Serverless 4168
Aggiornamenti a versioni secondarie automatiche 4168
Utilizzo dell'API dati Amazon RDS 4169
Disponibilità di regioni e versioni per l'Amazon RDS Data API 4170
Limitazioni 4170
Data API rispetto a Aurora Serverless v2 Aurora Serverless v1 4171
Numero massimo di richieste al secondo 4172

Abilitazione o disabilitazione dell'API Amazon RDS Data su un database esistente 4172
CloudTrail eventi 4172
Supporto per più istruzioni 4173
Richieste simultanee per lo stesso ID di transazione 4174
BatchExecuteStatement comportamento 4175
Comportamento ExecuteSQL 4175
ExecuteStatement comportamento 4175
Comportamento dei parametri dello schema 4176
Autenticazione e autorizzazione 4177
Autorizzazione basata su tag 4178
Memorizzazione delle credenziali in un segreto di Secrets Manager 4180
Abilitazione dell'API Data 4181
Abilitazione dell'API RDS Data quando crei un database 4182
Abilitazione dell'API RDS Data su un database esistente 4183
Creazione di un endpoint Amazon VPC () PrivateLink 4186
Chiamata dell'API Data 4189
Riferimento operativo 4190
Chiamata da AWS CLI 4194
Chiamate dalle app Python 4205
Chiamate da app Java 4209
Controllo del comportamento di timeout 4214
Libreria client Java 4216
Download della libreria client Java per l'API dati 4216
Esempi di librerie client Java 4216
Elaborazione dei risultati delle query in JSON 4218
Recupero dei risultati delle query in formato JSON 4219
Mappatura dei tipi di dati 4219
Risoluzione dei problemi 4220
Esempi 4221
Risoluzione dei problemi relativi all'API Data 4226
<transaction_ID>La transazione non è stata trovata 4226
Il pacchetto per la query è troppo grande 4227
La risposta del database è andata oltre il limite delle dimensioni 4227
HttpEndpointnon è abilitato per il cluster <cluster_ID> 4227
DatabaseErrorException: Transaction sta ancora eseguendo una query 4228
Eccezione di risultato non supportata 4228

Le istruzioni multiple non sono supportate ... 4228
Il parametro Schema non è supportato ... 4229
Registrazione delle chiamate Data API ... 4229
Utilizzo delle informazioni dell'API dati in CloudTrail ... 4229
Inclusione ed esclusione di eventi Data API da un CloudTrail percorso ... 4230
Informazioni sulle voci dei file di registro dell'API dei dati ... 4233
Monitoring Data API ... 4235
Come vengono rappresentate le query RDS Data API in Performance Insights ... 4235
Utilizzo dell'editor della query ... 4236
Disponibilità dell'editor di query ... 4236
Autorizzazione di accesso ... 4236
Esecuzione di query ... 4238
DBQMSAPIriferimento ... 4242
CreateFavoriteQuery ... 4243
CreateQueryHistory ... 4243
CreateTab ... 4243
DeleteFavoriteQueries ... 4243
DeleteQueryHistory ... 4243
DeleteTab ... 4243
DescribeFavoriteQueries ... 4243
DescribeQueryHistory ... 4244
DescribeTabs ... 4244
GetQueryString ... 4244
UpdateFavoriteQuery ... 4244
UpdateQueryHistory ... 4244
UpdateTab ... 4244
Utilizzo dell'apprendimento automatico Aurora ... 4245
Utilizzo dell'apprendimento automatico di Aurora con Aurora My SQL ... 4246
Requisiti per l'utilizzo del machine learning di Aurora ... 4247
Disponibilità di regioni e versioni ... 4248
Funzionalità e limitazioni supportate ... 4249
Configurazione del cluster Aurora per machine learning di Aurora ... 4250
Utilizzo di Amazon Bedrock con il cluster Aurora My DB SQL ... 4263
Utilizzo di Amazon Comprehend con il cluster Aurora My DB SQL ... 4266
Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora My DB SQL ... 4268
Considerazioni sulle prestazioni ... 4272

Monitoraggio 4274
Utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL 4275
Requisiti per l'utilizzo del machine learning di Aurora 4276
Funzionalità e limitazioni supportate 4277
Configurazione del cluster database Aurora per utilizzare il machine learning di Aurora 4278
Utilizzo di Amazon Bedrock con il cluster Aurora SQL Postgre DB 4291
Utilizzo di Amazon Comprehend con il cluster Aurora Postgre DB SQL 4294
Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora SQL Postgre DB 4296
Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced) 4300
Considerazioni sulle prestazioni 4301
Monitoraggio 4307
Esempi di codice 4309
Nozioni di base 4320
Hello Aurora 4321
Informazioni di base 4331
Azioni 4501
Scenari 4705
Creazione di una REST API per la libreria di prestiti 4705
Creazione di un tracciatore di elementi di lavoro di Aurora Serverless 4706
Best practice con Aurora 4711
Linee guida operative di base per Amazon Aurora 4711
Suggerimenti relativi alla RAM per un'istanza di database 4712
AWS driver di database 4713
Monitoraggio di Amazon Aurora 4713
Uso di gruppi di parametri database e gruppi di parametri cluster database 4714
Video sulle best practice Amazon Aurora 4714
Esecuzione di un proof of concept di Aurora 4715
Panoramica di un proof of concept di Aurora 4715
1. Individuare gli obiettivi 4716
2. Comprendere le caratteristiche del carico di lavoro 4717
3. Esercitati con la console o CLI 4718
Pratica con la console 4718
Esercitati con AWS CLI 4719
4. Creare il cluster Aurora 4719
5. Configurare lo schema 4721

6. Importare i dati ... 4722
7. Trasferisci il tuo SQL codice ... 4722
8. Specificare le impostazioni di configurazione ... 4723
9. Connettere ad Aurora ... 4724
10. Eseguire il carico di lavoro ... 4726
11. Misurare le prestazioni ... 4726
12. Fare pratica con la disponibilità elevata di Aurora ... 4730
13. Cosa fare in seguito ... 4732
Sicurezza ... 4734
Database authentication (Autenticazione del database) ... 4736
Autenticazione password ... 4737
Autenticazione del database IAM ... 4738
Autenticazione Kerberos ... 4738
Gestione delle password con Aurora e Secrets Manager ... 4739
Disponibilità di regioni e versioni ... 4739
Limitazioni ... 4739
Panoramica ... 4740
Vantaggi ... 4740
Autorizzazioni necessarie per l'integrazione di Secrets Manager ... 4741
Implementazione della gestione da parte di Aurora ... 4742
Gestione della password dell'utente master per un cluster database ... 4743
Rotazione del segreto della password dell'utente master per un cluster database ... 4747
Visualizzazione dei dettagli di un segreto per un cluster database ... 4749
Protezione dei dati ... 4752
Crittografia dei dati ... 4753
Riservatezza del traffico Internet ... 4784
Gestione dell'identità e degli accessi ... 4786
Destinatari ... 4786
Autenticazione con identità ... 4787
Gestione dell'accesso con policy ... 4791
Come funziona Amazon Aurora con IAM ... 4793
Esempi di policy basate su identità ... 4801
AWS politiche gestite ... 4819
Aggiornamenti alle policy ... 4825
Prevenzione del problema "confused deputy" tra servizi ... 4836
Autenticazione del database IAM ... 4838

Risoluzione dei problemi ... 4888
Registrazione di log e monitoraggio ... 4890
Convalida della conformità ... 4893
Resilienza ... 4894
Backup e ripristino ... 4894
Replica ... 4895
Failover ... 4895
Sicurezza dell'infrastruttura ... 4896
Gruppi di sicurezza ... 4896
Public accessibility (Accesso pubblico) ... 4896
Endpoint VPC (AWS PrivateLink) ... 4898
Considerazioni ... 1122
Disponibilità ... 1122
Creazione di un endpoint VPC dell'interfaccia ... 1122
Creazione di una policy di endpoint VPC ... 1122
Best practice relative alla sicurezza ... 4901
Controllo dell'accesso con i gruppi di sicurezza ... 4903
Panoramica dei gruppi di sicurezza VPC ... 4903
Scenario del gruppo di sicurezza ... 4904
Creazione di un gruppo di sicurezza VPC ... 4905
Associazione a un cluster database ... 4906
Privilegi dell'account utente master ... 4906
Ruoli collegati ai servizi ... 4909
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora ... 4909
Autorizzazioni di ruolo collegate ai servizi per Amazon RDS Beta ... 4912
Ruolo collegato ai servizi per Amazon RDS Preview ... 4913
Utilizzo di Amazon Aurora con Amazon VPC ... 4915
Uso di un cluster database in un VPC ... 4915
Scenari per l'accesso a un cluster di DB in un VPC ... 4932
Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 ... 4938
Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack) . 4946
Quote e vincoli ... 4957
Quote in Amazon Aurora ... 4957
Vincoli per la denominazione in Amazon Aurora ... 4963
Limiti di dimensione Amazon Aurora ... 4964
Risoluzione dei problemi ... 4966

Impossibile connettersi all'istanza database di ... 4966
Test della connessione a un'istanza di database ... 4969
Risoluzione di problemi di autenticazione della connessione ... 4970
Problemi relativi alla sicurezza ... 4970
Messaggio di errore "Impossibile recuperare gli attributi dell'account, alcune funzioni della console potrebbero non essere attive." ... 4970
Reimpostazione della password del ruolo di proprietario dell'istanza di database ... 4970
Errore o riavvio di un'istanza di database ... 4971
Modifiche ai parametri che non hanno effetto ... 4972
Problemi di memoria liberabile Aurora ... 4972
Problemi di replica di Aurora MySQL ... 4973
Diagnosi e risoluzione del ritardo tra repliche di lettura ... 4974
Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL ... 4976
Errore di replica interrotta ... 4978
La replica in lettura non riesce a inizializzare la struttura dei metadati ... 4978
Documentazione di riferimento dell'API Amazon RDS ... 4980
Uso dell'API query ... 4980
Parametri di query ... 4980
Autenticazione delle richieste di query ... 4981
Risoluzione dei problemi delle applicazioni ... 4981
Errore durante il recupero ... 4981
Suggerimenti per la risoluzione dei problemi ... 4982
Cronologia dei documenti ... 4983
AWS Glossario ... 5088
... 5089

Che cos'è Amazon Aurora?

Amazon Aurora (Aurora) è un motore di database relazionale completamente gestito compatibile con My e Postgre. SQL SQL Sai già come My SQL e Postgre SQL combinano la velocità e l'affidabilità dei database commerciali di fascia alta con la semplicità e l'economicità dei database open source. Il codice, gli strumenti e le applicazioni che usi oggi con i tuoi SQL database My SQL e Postgre esistenti possono essere usati con Aurora. Con alcuni carichi di lavoro, Aurora è in grado di offrire un throughput fino a cinque volte superiore a SQL My e fino a tre volte quello di SQL Postgre senza richiedere modifiche alla maggior parte delle applicazioni esistenti.

Aurora include un sottosistema di storage ad alte prestazioni. I suoi motori di database SQL compatibili con My SQL e Postgre sono personalizzati per sfruttare questo storage distribuito veloce. Lo storage sottostante cresce automaticamente in funzione delle necessità. Un volume del cluster Aurora può raggiungere una dimensione massima di 128 tebibytes (TiB). Aurora inoltre automatizza ed esegue la standardizzazione del clustering e della replica di database, ovvero due aspetti in genere tra i più impegnativi nell'ambito della configurazione e dell'amministrazione dei database.

Aurora fa parte del servizio di database gestito Amazon Relational Database Service (Amazon). RDS Amazon RDS semplifica la configurazione, il funzionamento e la scalabilità di un database relazionale nel cloud. Se non conosci già AmazonRDS, consulta la Amazon Relational Database Service User Guide. Per ulteriori informazioni sulle varie opzioni di database disponibili su Amazon Web Services, consulta Choosing the right database for your organization on AWS (Scelta del database appropriato per l'organizzazione su AWS).

Argomenti

- Modello di responsabilità RDS condivisa di Amazon
- Come funziona Amazon Aurora con Amazon RDS
- Cluster database Amazon Aurora
- Versioni di Amazon Aurora
- Regioni e zone di disponibilità
- Funzionalità supportate in Amazon Aurora by e Regione AWS Aurora DB engine
- Connessioni endpoint Amazon Aurora
- Classi di istanze DB Amazon Aurora
- Archiviazione Amazon Aurora
- Affidabilità Amazon Aurora

- Sicurezza di Amazon Aurora
- Elevata disponibilità di Amazon Aurora
- Replica con Amazon Aurora
- DB per Aurora

Modello di responsabilità RDS condivisa di Amazon

Amazon RDS è responsabile dell'hosting dei componenti software e dell'infrastruttura delle istanze DB e dei cluster DB. Sei responsabile dell'ottimizzazione delle query, ovvero il processo di regolazione delle SQL query per migliorare le prestazioni. Le prestazioni delle query dipendono fortemente dalla progettazione del database, dalla dimensione dei dati, dalla distribuzione dei dati, dal carico di lavoro dell'applicazione e dai modelli di query, che possono variare notevolmente. Il monitoraggio e l'ottimizzazione sono processi altamente personalizzati di tua proprietà per i tuoi database. RDS Puoi utilizzare Amazon RDS Performance Insights e altri strumenti per identificare le query problematiche.

Come funziona Amazon Aurora con Amazon RDS

I seguenti punti illustrano in che modo Amazon Aurora si relaziona ai motori SQL My e SQL Postgre standard disponibili in Amazon: RDS

- Scegli Aurora My SQL o Aurora Postgre SQL come opzione del motore DB quando configuri nuovi server di database tramite Amazon. RDS
- Aurora sfrutta le familiari funzionalità di Amazon Relational Database Service (AmazonRDS) per la gestione e l'amministrazione. Aurora utilizza l'RDS AWS Management Console interfaccia, AWS CLI i comandi e API le operazioni di Amazon per gestire le attività di routine del database come il provisioning, l'applicazione di patch, il backup, il ripristino, il rilevamento degli errori e la riparazione.
- Le operazioni di gestione Aurora in genere implicano cluster interi di server di database sincronizzati mediante operazioni di replica anziché singole istanze database. Il clustering automatico, la replica e l'allocazione dello storage rendono semplice ed economica la configurazione, il funzionamento e la scalabilità delle più grandi implementazioni My e Postgre. SQL SQL
- Puoi trasferire dati da Amazon RDS for My SQL e Amazon RDS for Postgre in SQL Aurora creando e ripristinando istantanee o configurando la replica unidirezionale. Puoi utilizzare strumenti di

migrazione premendo pulsanti per convertire le tue applicazioni esistenti RDS per My SQL e SQL Postgre RDS in Aurora.

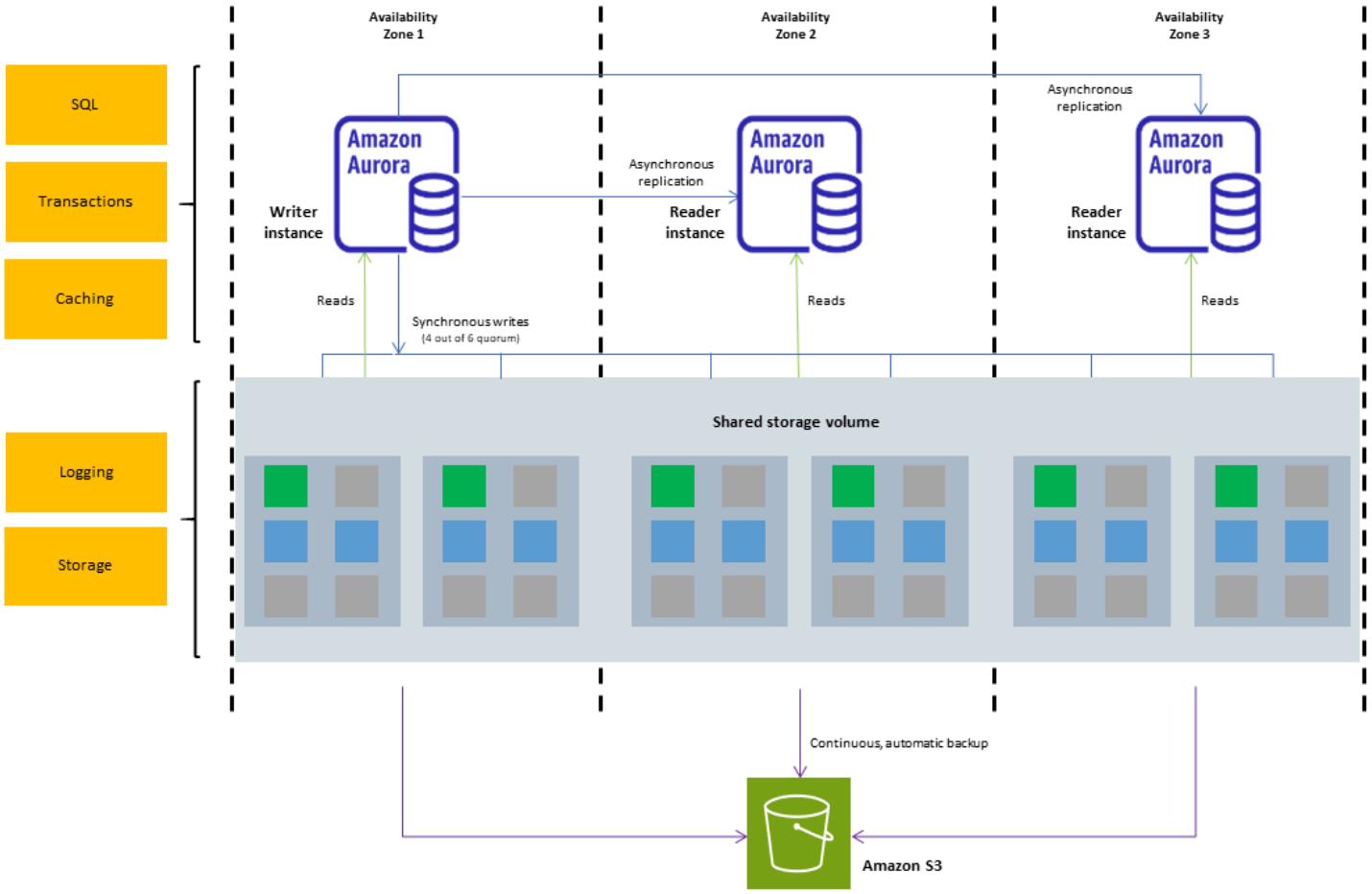
Prima di utilizzare Amazon Aurora, completa le fasi indicate in Configurazione dell'ambiente per Amazon Aurora e quindi esamina i concetti e le caratteristiche di Aurora in Cluster database Amazon Aurora.

Cluster database Amazon Aurora

Un cluster database Amazon Aurora è composto da una o più istanze database e da un volume del cluster che gestisce i dati per tali istanze. Un volume del cluster Aurora è uno storage di database virtuale che si estende su più zone di disponibilità, ciascuna delle quali include una copia dei dati del cluster database. Un cluster database Aurora è formato da due tipi di istanze:

- Istanza DB primaria (writer): supporta le operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati sul volume del cluster. Ciascun cluster database Aurora ha un'istanza database primaria.
- Aurora Replica (istanza Reader DB): si connette allo stesso volume di archiviazione dell'istanza DB principale ma supporta solo operazioni di lettura. Oltre all'istanza database primaria, ciascun cluster database Aurora può avere fino a 15 repliche di Aurora. È possibile mantenere l'alta disponibilità posizionando le repliche Aurora in zone di disponibilità separate. Aurora esegue automaticamente il failover in una replica di Aurora nel caso in cui l'istanza database principale non sia più disponibile. È possibile specificare la priorità di failover per le repliche di Aurora. Le repliche di Aurora possono anche effettuare l'offload dei carichi di lavoro in lettura dall'istanza database primaria.

Il diagramma seguente illustra la relazione tra il volume del cluster, l'istanza Writer DB e le istanze DB Reader in un cluster Aurora DB.



> **Note**
>
> Le informazioni precedenti si applicano a tutti i cluster Aurora DB: provisioned, parallel query, Aurora Global Database, Aurora MySQL compatibile e Aurora Aurora Serverless PostgreSQL.

Il cluster Aurora DB illustra la separazione tra capacità di elaborazione e archiviazione. Ad esempio, una configurazione Aurora con una sola istanza DB è ancora un cluster, poiché il volume di storage sottostante include più nodi di storage distribuiti su più zone di disponibilità ()AZs.

Input/output (I/O) le operazioni nei cluster Aurora DB vengono conteggiate allo stesso modo, indipendentemente dal fatto che si trovino su un'istanza DB writer o reader. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

Versioni di Amazon Aurora

Con Amazon Aurora, puoi scegliere il motore di database relazionale supportato che meglio si adatta ai requisiti delle tue applicazioni mantenendo la compatibilità con i motori sottostanti. Aurora riutilizza il codice del motore di database dei motori supportati, in modo da poter sfruttare le competenze, gli strumenti e le librerie esistenti per tali motori. Quando crei un cluster, specifichi la versione del motore di database Amazon Aurora che desideri utilizzare. La versione scelta determina la compatibilità e le funzionalità disponibili.

Questa documentazione spiega i punti comuni e le differenze tra Amazon Aurora e i motori di database corrispondenti. Con queste informazioni, puoi determinare la versione del software da selezionare e come verificare le funzionalità disponibili e le correzioni di bug in ciascuna versione. È inoltre possibile utilizzare questo riferimento per determinare la cadenza di aggiornamento appropriata e pianificare l'aggiornamento.

Motori di database supportati per i cluster di database Amazon Aurora

I seguenti database relazionali sono disponibili su Amazon Aurora. Aurora riutilizza il codice e mantiene la compatibilità con i motori DB sottostanti. Tuttavia, Aurora ha numeri di versione, cicli di rilascio e tempistiche univoci per l'obsolescenza delle versioni. Ogni nuova versione di Aurora include note di rilascio che elencano le nuove funzionalità, le correzioni e altre modifiche e miglioramenti che si applicano a ciascuna versione.

Banca dati Aurora	Guida per l'utente	Versioni disponibili	Note di rilascio
Amazon Aurora edizione compatibile con MySQL	Utilizzo di Amazon Aurora MySQL	Aggiornamenti del motore di database per Amazon Aurora My SQL	Note di rilascio di Aurora MySQL
Amazon Aurora PostgreSQL-Compatible Edition	Utilizzo di Amazon Aurora PostgreSQL	Aggiornamenti del motore di database per Amazon Aurora Postgre SQL	Note di rilascio per Aurora PostgreSQL

Specifica della versione del database Amazon Aurora per il cluster database

Quando si crea un nuovo cluster DB con l'operazione Crea database in o AWS CLI, AWS Management Console o con l'operazione `CreateDBCluster` API, è possibile specificare qualsiasi versione del database Aurora attualmente disponibile (principale o secondaria).

Per ulteriori informazioni su come creare cluster Aurora, consulta Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni su come modificare la versione di un cluster Aurora esistente, consulta Modifica di un cluster database Amazon Aurora.

> **Note**
>
> Non tutte le versioni del database Aurora sono disponibili in tutte. Regione AWS Per ulteriori informazioni sulle regioni e sulle versioni disponibili in ciascuna di esse Regione AWS, consulta Regioni e zone di disponibilità eRegioni e motori DB supportati per i database globali Aurora.

Controllo delle versioni di Amazon Aurora

Le versioni di Amazon Aurora sono diverse dai database della community upstream con cui sono compatibili. Per aiutarti a mantenere la compatibilità delle applicazioni e a sfruttare le funzionalità più recenti del motore DB, le sezioni seguenti spiegano le convenzioni di controllo delle versioni di Aurora e come le versioni di Aurora vengono mappate ai rispettivi database della comunità.

Per un elenco dei database relazionali disponibili su Amazon Aurora, consulta. Motori di database supportati per i cluster di database Amazon Aurora

Differenze nei numeri di versione tra database della community e Aurora

Ogni versione di Amazon Aurora è compatibile con una versione specifica del database della community corrispondente. Puoi trovare la versione community del tuo database con la `version` funzione e la versione Aurora con la `aurora_version` funzione.

Gli esempi seguenti mostrano come trovare la versione comunitaria del database per Aurora MySQL e Aurora PostgreSQL.

Aurora MySQL

La `version` funzione restituisce la versione comunitaria del database per Aurora MySQL.

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

Esempio di output:

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

E la `aurora_version` funzione restituisce la versione Aurora:

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

Esempio di output:

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

Aurora PostgreSQL

La `version` funzione restituisce la versione comunitaria del database per Aurora PostgreSQL.

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

Esempio di output:

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

E la `aurora_version` funzione restituisce la versione Aurora:

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

Esempio di output:

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

Per ulteriori informazioni, consultare Verifica delle versioni di Aurora My utilizzando SQL SQL e Identificazione delle versioni di Amazon Aurora Postgre SQL.

Versioni predefinite di Amazon Aurora

La versione predefinita è la versione che Aurora sceglie automaticamente per la creazione o l'aggiornamento del database quando non si specifica manualmente una versione del motore di destinazione. Ad esempio, il comando seguente mostra la versione predefinita del motore per Aurora PostgreSQL (output di esempio incluso).

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

16.4
```

A ogni versione principale corrisponde una versione secondaria predefinita. Pertanto, la versione secondaria predefinita è 16. n per Aurora PostgreSQL 16, con il numero di versione n che cambia quando Aurora rilascia nuove versioni secondarie predefinite. In genere, Aurora rilascia due versioni secondarie predefinite per ogni versione principale all'anno. Il seguente script della shell bash mostra le versioni secondarie predefinite per un set di versioni principali di Aurora PostgreSQL (output di esempio incluso).

```
for major in 16 15 14 13 12 11; do
  echo -n "Default for Aurora PostgreSQL major version $major: "
  aws rds describe-db-engine-versions \
    --engine aurora-postgresql \
    --engine-version "$major" \
    --default-only \
    --query 'DBEngineVersions[0].EngineVersion' \
    --output text
```

```
done

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

Se abiliti gli aggiornamenti automatici delle versioni secondarie per il tuo cluster Aurora DB, Aurora utilizza la versione secondaria predefinita o una versione secondaria più recente per una determinata versione principale. Ad esempio, se la versione secondaria predefinita per Aurora PostgreSQL 15 è la 15.8 ed è disponibile anche la versione più recente 15.10, Aurora può eseguire automaticamente l'aggiornamento alla 15.8 o alla 15.10.

Versioni principali di Amazon Aurora

Le versioni di Aurora usano lo schema *major*.*minor*.*patch*. Versione principale di Aurora si riferisce alla versione principale della community MySQL o PostgreSQL con cui Aurora è compatibile. Il supporto standard delle versioni principali di Aurora MySQL e Aurora PostgreSQL è disponibile almeno fino alla fine del ciclo di vita della community per la versione della community corrispondente. Puoi continuare a eseguire una versione principale dopo la data di fine del supporto standard di Aurora a pagamento. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con and Prezzi di Amazon Aurora.

Per ulteriori informazioni sulle versioni principali e sul calendario di rilascio di Aurora MySQL e Aurora PostgreSQL, consulta le pagine seguenti nelle rispettive Note di rilascio:

- Calendario di rilascio per le versioni principali di Aurora MySQL
- Calendario di rilascio per le versioni principali di Aurora PostgreSQL

È inoltre possibile visualizzare le informazioni sulle date di supporto per le principali versioni del motore eseguendo il AWS CLI comando describe-db-major-engine-versions o utilizzando l'operazione Descrivi RDS API. DBMajor EngineVersions

> **Note**
>
> Amazon RDS Extended Support per Aurora MySQL versione 2 inizia il 1° novembre 2024, ma non ti verrà addebitato alcun costo fino al 1° dicembre 2024. Tra il 1° novembre e il 30

novembre 2024, tutti i cluster DB Aurora MySQL versione 2 sono coperti da Amazon RDS Extended Support. Per ulteriori informazioni, consulta Amazon RDS Extended Support per versioni Aurora selezionate.

Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili

Le versioni principali di Amazon Aurora restano disponibili almeno fino alla fine del ciclo di vita della comunità per la versione della community corrispondente. Puoi utilizzare le date di fine del supporto standard per Aurora per pianificare i cicli di test e aggiornamento. Queste date rappresentano la prima data in cui potrebbe essere richiesto un aggiornamento a una versione più recente. Per ulteriori informazioni sulle date, consulta Versioni principali di Amazon Aurora.

Prima che Aurora ti chieda di eseguire l'aggiornamento a una versione principale più recente e di aiutarti a pianificare, riceverai un promemoria con almeno 12 mesi di anticipo. I promemoria comunicano quanto segue sul processo di aggiornamento.

- La tempistica di determinate tappe
- L'impatto sui cluster DB
- Azioni consigliate

Ti consigliamo di testare a fondo le tue applicazioni con nuove versioni del database prima di aggiornare il cluster a una nuova versione principale.

Dopo che la versione principale raggiunge la fine del supporto standard di Aurora, qualsiasi cluster DB che esegue ancora la versione precedente viene automaticamente aggiornato a una versione Extended Support durante una finestra di manutenzione pianificata. Potrebbero essere applicati costi di Extended Support. Per ulteriori informazioni su Amazon RDS Extended Support, consulta Using Amazon RDS Extended Support.

Versioni secondarie di Amazon Aurora

Le versioni di Aurora usano lo schema *major*.*minor*.*patch*. Una versione secondaria di Aurora fornisce al servizio miglioramenti incrementali specifici per la community e Aurora, ad esempio nuove funzionalità e correzioni.

Per ulteriori informazioni sulle versioni secondarie e sul calendario di rilascio di Aurora MySQL e Aurora PostgreSQL, consulta le pagine seguenti nelle rispettive Note di rilascio:

- Calendario di rilascio per le versioni minori di Aurora MySQL
- Calendario di rilascio per le versioni secondarie di Aurora PostgreSQL

Le sezioni seguenti descrivono i dettagli sulla cadenza e la durata che ci si può aspettare dalle versioni secondarie di Aurora.

Argomenti

- Quanto spesso vengono rilasciate le versioni secondarie di Amazon Aurora
- Per quanto tempo le versioni secondarie di Amazon Aurora rimangono disponibili

Quanto spesso vengono rilasciate le versioni secondarie di Amazon Aurora

In generale, le versioni secondarie di Amazon Aurora vengono rilasciate ogni trimestre. La pianificazione del rilascio potrebbe variare in base a funzionalità aggiuntive o correzioni.

Per quanto tempo le versioni secondarie di Amazon Aurora rimangono disponibili

In genere, Amazon Aurora rende disponibile ogni versione secondaria di una particolare versione principale per almeno 12 mesi. Al termine di questo periodo, Aurora potrebbe aggiornare automaticamente il database alla versione secondaria predefinita o a una versione successiva. Aurora avvia l'aggiornamento durante la finestra di manutenzione pianificata per qualsiasi cluster DB che esegue la versione secondaria precedente.

In alcuni casi, Aurora potrebbe sostituire una versione secondaria di una particolare versione principale prima del normale periodo di 12 mesi. I motivi possono includere problemi di sicurezza critici o la end-of-support data di una versione principale.

Prima di iniziare gli aggiornamenti automatici delle versioni secondarie che si avvicinano alla fine del ciclo di vita, Aurora in genere fornisce un promemoria con tre mesi di anticipo. Aurora descrive in dettaglio quanto segue sul processo di aggiornamento.

- La tempistica di determinate tappe
- L'impatto sui cluster DB
- Azioni consigliate

Le notifiche con un preavviso inferiore a tre mesi descrivono questioni critiche, come i problemi di sicurezza, che richiedono un'azione più rapida.

Se l'impostazione Aggiornamento automatico della versione secondaria è abilitata, viene visualizzato un promemoria ma nessuna notifica di evento RDS. Aurora aggiorna il database entro una finestra di manutenzione dopo la scadenza obbligatoria per l'aggiornamento.

Se l'impostazione Aggiornamento automatico della versione secondaria non è abilitata, riceverai un promemoria e un evento del cluster Amazon RDS DB con una categoria `maintenance` e un ID di. `RDS-EVENT-0156` Aurora aggiorna il database entro la finestra di manutenzione successiva.

Tieni presente che dopo che una versione secondaria raggiunge la fine del supporto standard di Aurora, non verranno rilasciate ulteriori versioni di patch per quella versione secondaria. Per ricevere correzioni di bug critici o CVEs, è necessario eseguire l'aggiornamento a una versione secondaria con supporto standard.

Per ulteriori informazioni sugli aggiornamenti automatici delle versioni secondarie, consulta. Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora

Versioni delle patch di Amazon Aurora

Le versioni di Aurora usano lo schema *`major`*`.`*`minor`*`.`*`patch`*. Una versione patch di Aurora include importanti correzioni aggiunte a una versione secondaria dopo la sua versione iniziale (ad esempio, Aurora MySQL 3.04.0, 3.04.1,..., 3.04.3). Mentre ogni nuova versione secondaria fornisce nuove funzionalità Aurora, le nuove versioni di patch all'interno di una specifica versione secondaria vengono utilizzate principalmente per risolvere problemi importanti.

Per ulteriori informazioni sull'applicazione di patch, consulta Manutenzione di un cluster database Amazon Aurora.

Aggiornamento dei cluster Amazon Aurora DB

Con Amazon Aurora, puoi controllare e testare gli upgrade dei tuoi cluster DB. Amazon Aurora offre opzioni per gli aggiornamenti automatici delle versioni secondarie, il controllo manuale degli aggiornamenti, gli aggiornamenti richiesti e i test pre-aggiornamento. Puoi mantenere i cluster up-to-date con l'ultima versione secondaria, rimandare gli aggiornamenti non critici, forzare gli aggiornamenti per problemi gravi e convalidare il comportamento degli aggiornamenti in ambienti non di produzione. Le seguenti sezioni descrivono in dettaglio come gestire e testare gli aggiornamenti del cluster Aurora DB utilizzando queste funzionalità.

Aggiornamenti automatici delle versioni secondarie per Aurora

Gli aggiornamenti automatici delle versioni secondarie aggiornano periodicamente il database alle versioni recenti del motore di database. Tuttavia, l'aggiornamento potrebbe non includere sempre la versione più recente del motore di database. Se è necessario mantenere i database su versioni specifiche in momenti particolari, si consiglia di eseguire manualmente l'aggiornamento alle versioni del database necessarie in base alla pianificazione richiesta. In caso di problemi di sicurezza critici o quando una versione raggiunge la end-of-support data di scadenza, Amazon Aurora potrebbe applicare un aggiornamento della versione secondaria anche se non hai abilitato l'opzione Aggiornamento automatico della versione secondaria. Per ulteriori informazioni, consulta la documentazione di aggiornamento per il tuo motore di database specifico.

Consulta Aggiornamento della versione secondaria o del livello di patch di un cluster Aurora My DB SQL e Esecuzione di un aggiornamento di versione minore.

Puoi rimanere aggiornato con le versioni secondarie di Aurora attivando Aggiornamento automatico versione secondaria per ogni istanza database nel cluster Aurora. Aurora esegue l'aggiornamento automatico solo se tutte le istanze DB del cluster hanno questa impostazione attivata.

Se l'aggiornamento automatico della versione secondaria è Sì per il cluster DB, Aurora esegue l'aggiornamento automaticamente alla versione secondaria predefinita o a una versione secondaria più recente. Ad esempio, se la versione secondaria predefinita è 15.8 per Aurora PostgreSQL 15 ed esiste la versione 15.10, la destinazione per l'aggiornamento automatico potrebbe essere la 15.8 o la 15.10.

Aurora pianifica in genere gli aggiornamenti automatici due volte all'anno per i cluster DB che hanno abilitato l'aggiornamento automatico delle versioni secondarie. Questi aggiornamenti avvengono durante la finestra di manutenzione specificata per il cluster. Per ulteriori informazioni, consulta Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora.

Gli aggiornamenti automatici delle versioni secondarie vengono comunicati in anticipo tramite un evento cluster database Amazon RDS con una categoria di `maintenance` e un ID di `RDS-EVENT-0156`. Per ulteriori informazioni, consulta Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora.

Controllo manuale degli aggiornamenti dei cluster DB alle nuove versioni

Se hai abilitato l'impostazione Aggiornamento automatico della versione secondaria, Aurora aggiorna automaticamente il cluster DB alla versione secondaria predefinita o a una versione secondaria

più recente. Aurora in genere pianifica gli aggiornamenti automatici due volte all'anno per i cluster DB con l'impostazione Auto minor version upgrade abilitata. Questi aggiornamenti vengono avviati durante le finestre di manutenzione specificate dal cliente.

Per disattivare gli aggiornamenti automatici delle versioni secondarie, disabilita l'aggiornamento automatico delle versioni secondarie su qualsiasi istanza DB all'interno di un cluster Aurora. Aurora esegue un aggiornamento automatico della versione secondaria solo se tutte le istanze DB del cluster hanno l'impostazione abilitata.

> **Note**
>
> Per gli aggiornamenti obbligatori come la fine del ciclo di vita della versione minore, Aurora aggiorna il cluster DB anche se l'impostazione Aggiornamento automatico della versione secondaria è disabilitata. Riceverai un promemoria ma nessuna notifica sugli eventi RDS. Aurora aggiorna il cluster entro una finestra di manutenzione dopo la scadenza obbligatoria per l'aggiornamento.

Poiché presentano rischi relativi alla compatibilità, gli aggiornamenti delle versioni principali non vengono eseguiti automaticamente. È necessario avviarli, a meno che non sia presente una versione principale obsoleta. Si consiglia di testare a fondo le applicazioni con nuove versioni del database prima di aggiornare il cluster a una versione principale.

Per ulteriori informazioni sull'aggiornamento di un cluster database a una nuova versione principale di Aurora, consulta Aggiornamento dei cluster Amazon Aurora My DB SQL e Aggiornamento dei cluster database Amazon Aurora PostgreSQL.

Aggiornamenti obbligatori di Amazon Aurora

Per alcune correzioni critiche, Aurora potrebbe eseguire un aggiornamento gestito a un livello di patch più recente all'interno della stessa versione secondaria. In questo caso, Aurora aggiorna il cluster anche se l'aggiornamento automatico della versione secondaria è disattivato. Prima di farlo, Aurora comunica il processo di aggiornamento dettagliato. I dettagli includono la tempistica di determinati traguardi, l'impatto sui cluster di database e le azioni consigliate. Tali aggiornamenti gestiti avvengono automaticamente all'interno della finestra di manutenzione del cluster.

Test del cluster database con una nuova versione di Aurora prima di eseguire l'aggiornamento

È possibile testare il processo di aggiornamento e il funzionamento della nuova versione con l'applicazione e il carico di lavoro. Selezionare uno dei seguenti metodi:

- Clona il tuo cluster utilizzando la funzione di clonazione rapida del database di Amazon Aurora. Esegui l'aggiornamento ed eventuali test post-aggiornamento sul nuovo cluster.
- Esegui il ripristino da uno snapshot del cluster per creare un nuovo cluster Aurora. Puoi creare uno snapshot del cluster da un cluster Aurora esistente. Aurora crea automaticamente snapshot periodici per ogni cluster. È quindi possibile avviare un aggiornamento della versione per il nuovo cluster. Puoi sperimentare la copia aggiornata del cluster prima di decidere se aggiornare il cluster originale.

Per ulteriori informazioni sui modi per creare nuovi cluster per il test, consulta Clonazione di un volume per un cluster di database Amazon Aurora e Creazione di uno snapshot del cluster database.

Supporto per la versione di Amazon Aurora

Se il tuo database Amazon Aurora ha dipendenze complesse dal comportamento specifico del motore di database, ti consigliamo di eseguire test approfonditi prima di eseguire l'aggiornamento a versioni più recenti del motore di database. Esistono opzioni di supporto a lungo termine che consentono di mantenere i cluster DB su alcune versioni del motore Aurora anche dopo che sono state sostituite da versioni più recenti. Le seguenti sezioni spiegano le opzioni di supporto a lungo termine per i cluster Aurora DB.

Supporto a lungo termine per versioni secondarie selezionate di Amazon Aurora

Per ogni versione principale di Aurora, alcune versioni secondarie sono designate come versioni long-term-support (LTS) e rese disponibili per almeno tre anni. In altre parole, almeno una versione secondaria per versione principale è disponibile per più di 12 mesi. In genere, Aurora te lo ricorda sei mesi prima della fine di questo periodo. Aurora comunica quanto segue sul processo di aggiornamento.

- La tempistica di determinate tappe
- L'impatto sui cluster DB
- Azioni consigliate

Le notifiche con un preavviso inferiore a sei mesi comunicano questioni critiche, come i problemi di sicurezza, che richiedono un'azione più rapida.

Le versioni secondarie LTS includono solo correzioni critiche (attraverso versioni di patch). Una versione LTS non include nuove funzionalità rilasciate dopo la sua introduzione. Una volta all'anno, ai cluster database in esecuzione su una versione secondaria LTS viene assegnata una patch con versione più recente della versione LTS. Aurora corregge i cluster in modo da trarre vantaggio da correzioni cumulative di sicurezza e stabilità. In caso di correzioni critiche, Aurora potrebbe applicare più frequentemente patch a una versione secondaria LTS, ad esempio per motivi di sicurezza, che devono essere applicate.

> **Note**
>
> Se desideri rimanere su una versione secondaria LTS per tutta la durata del suo ciclo di vita, assicurati di disabilitare l'aggiornamento automatico della versione secondaria per le tue istanze DB. Per evitare l'aggiornamento automatico del cluster DB dalla versione secondaria LTS, deseleziona la casella di controllo Abilita aggiornamento automatico della versione secondaria su qualsiasi istanza DB del cluster Aurora.

Per i numeri di versione di tutte le versioni Aurora LTS, consulta Aurora I miei rilasci di supporto SQL a lungo termine () LTS e Utilizzo di una versione di Aurora PostgreSQL con supporto a lungo termine (LTS).

Amazon RDS Extended Support per versioni Aurora selezionate

Con Amazon RDS Extended Support, puoi continuare a eseguire il database su una versione principale del motore oltre la data di fine del supporto standard di Aurora a un costo aggiuntivo. Durante RDS Extended Support, Amazon RDS fornirà patch per Critical e High, secondo CVEs quanto definito dalle classificazioni di gravità CVSS del National Vulnerability Database (NVD). Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con .

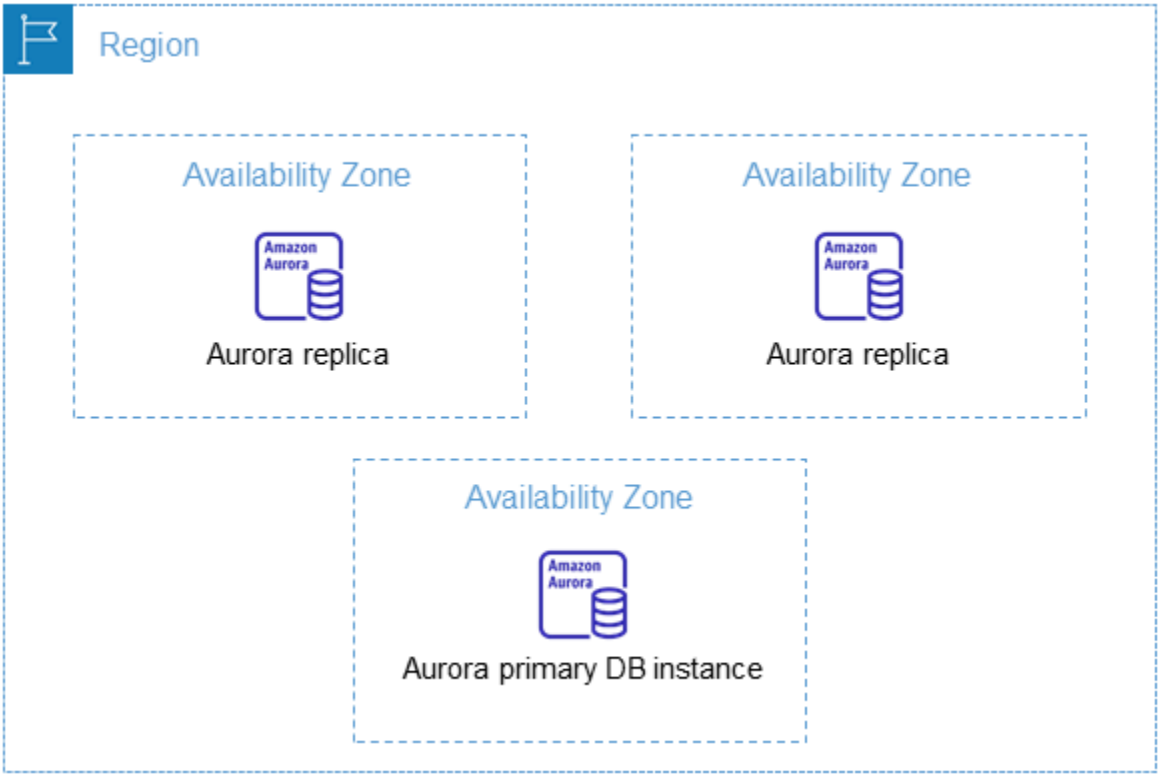
RDS Extended Support è disponibile solo su alcune versioni di Aurora. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora.

Regioni e zone di disponibilità

Le risorse di cloud computing Amazon sono ospitate in più ubicazioni in tutto il mondo. Queste località sono composte da AWS regioni e zone di disponibilità. Ciascuna regione AWS è un'area geografica distinta. Ogni AWS regione ha più località isolate note come zone di disponibilità.

> **Note**
>
> Per informazioni su come trovare le zone di disponibilità per una AWS regione, consulta Descrivi le tue zone di disponibilità nella EC2 documentazione di Amazon.

Amazon gestisce state-of-the-art data center ad alta disponibilità. Sebbene rari, possono verificarsi dei guasti che influiscano sulla disponibilità delle istanze database che si trovano nello stesso luogo. Se tutte le istanze database sono ospitate in un singolo luogo interessato da questo errore, nessuna delle istanze database sarà disponibile.



È importante ricordare che ogni AWS regione è completamente indipendente. Qualsiasi attività Amazon RDS avviata (ad esempio, la creazione di istanze di database o l'elenco delle istanze di database disponibili) viene eseguita solo nella regione predefinita corrente. AWS La AWS

regione predefinita può essere modificata nella console o impostando la variabile di ambiente. `AWS_DEFAULT_REGION` Oppure può essere sovrascritta utilizzando il `--region` parametro con AWS Command Line Interface ()AWS CLI. Per ulteriori informazioni, consulta la pagina relativa alla configurazione dell' AWS Command Line Interface, in particolare le sezioni relative alle variabili di ambiente e opzioni della riga di comando.

Amazon RDS supporta AWS regioni speciali chiamate AWS GovCloud (US). sono progettate per consentire ai clienti e agli enti governativi degli Stati Uniti di spostare nel cloud i carichi di lavoro più sensibili. Le regioni AWS GovCloud (US) fanno riferimento ai requisiti normativi e di compliance specifici del governo degli Stati Uniti. Per ulteriori informazioni, consulta Cos'è AWS GovCloud (US)?

Per creare o lavorare con un'istanza database Amazon RDS in una AWS regione specifica, utilizza l'endpoint di servizio regionale corrispondente.

> **Note**
>
> Aurora non supporta Local Zones

AWS Regioni

Ogni AWS regione è progettata per essere isolata dalle altre AWS regioni. Questo progetto permette di raggiungere la maggiore stabilità e tolleranza ai guasti possibile.

Quando si visualizzano le risorse, vengono visualizzate solo le risorse legate alla AWS regione specificata. Questo perché AWS le regioni sono isolate l'una dall'altra e non replichiamo automaticamente le risorse tra le AWS regioni.

Disponibilità nelle regioni

Quando si lavora con un cluster di database Aurora utilizzando l'interfaccia a riga di comando o le operazioni API, assicurarsi di specificare il relativo endpoint regionale.

Argomenti

- Disponibilità nelle regioni Aurora MySQL
- Disponibilità nelle regioni Aurora PostgreSQL

Disponibilità nelle regioni Aurora MySQL

La tabella seguente mostra le AWS regioni in cui Aurora MySQL è attualmente disponibile e l'endpoint per ciascuna regione.

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
Stati Uniti occidentali (California settentrionale)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
Africa (Cape Town)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Asia Pacific	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
(Hyderabad)				
Asia Pacifico (Giacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Malesia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Asia Pacifico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacifico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Osaka-Locale)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Asia Pacifico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asia Pacifico (Tailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asia Pacifico (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Centrale)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Canada occidentale (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Francoforte)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europa (Londra)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Europa (Milano)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	
Europa (Parigi)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS	
Europa (Spagna)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS	
Europa (Stoccolma)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS	
Europa (Zurigo)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS	
Israele (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS	
Messico (centrale)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS	
Medio Oriente (Bahrein)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS	
Medio Oriente (Emirati Arabi Uniti)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Sud America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Disponibilità nelle regioni Aurora PostgreSQL

La tabella seguente mostra le AWS regioni in cui Aurora PostgreSQL è attualmente disponibile e l'endpoint per ciascuna regione.

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Stati Uniti occidentali (California settentrionale)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
Africa (Cape Town)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Asia Pacific (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	
Asia Pacifico (Giacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Malesia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Asia Pacifico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacifico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Asia Pacifico (Osaka-Locale)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacifico (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Asia Pacifico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asia Pacifico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo	
Asia Pacifico (Tailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asia Pacifico (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Centrale)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Canada occidentale (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europa (Francoforte)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europa (Londra)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	
Europa (Milano)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	
Europa (Parigi)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo
Europa (Spagna)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
Europa (Stoccolma)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
Europa (Zurigo)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS
Israele (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
Messico (centrale)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Medio Oriente (Bahrein)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS
Medio Oriente (Emirati Arabi Uniti)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS
Sud America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS

Nome della regione	Regione	Endpoint	Protocollo	
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Zone di disponibilità

Una zona di disponibilità è una località isolata in una data Regione AWS specificata. Ogni regione ha più zone di disponibilità (AZ) progettate per fornire un'elevata disponibilità per la regione. Una AZ è identificata dal codice AWS regionale seguito da una lettera identificativa (ad esempio,`us-east-1a`). Se crei il tuo VPC e le sottoreti anziché utilizzare il VPC di default, definisci ogni sottorete in una AZ specifica. Quando crei un cluster database Aurora, Aurora crea l'istanza primaria in una delle sottoreti del gruppo di sottoreti database del VPC, associando così l'istanza a una specifica zona di disponibilità scelta da Aurora.

Ogni cluster Aurora DB ospita copie del proprio storage in tre file separati AZs selezionati automaticamente da Aurora dal gruppo di sottoreti del AZs database. Ogni istanza DB del cluster deve trovarsi in una di queste tre istanze. AZs

Quando crei un'istanza DB nel tuo cluster, Aurora sceglie automaticamente una AZ appropriata per quell'istanza se non specifichi una AZ.

Utilizza il EC2 comando describe-availability-zonesAmazon come segue per descrivere le zone di disponibilità all'interno della regione specificata abilitate per il tuo account.

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

Ad esempio, per descrivere le zone di disponibilità all'interno della regione Stati Uniti orientali (Virginia settentrionale) (us-east-1) abilitate per il tuo account, esegui il comando seguente:

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

Per informazioni su come specificare l'AZ quando si crea un cluster o vi si aggiungono istanze, consulta Configurazione della rete per il cluster database.

Fuso orario locale per i cluster DB Amazon Aurora

Per impostazione predefinita, il fuso orario di un cluster DB di Amazon Aurora è in formato UTC (Universal Time Coordinated). Tuttavia, puoi impostare il fuso orario delle istanze del cluster DB sul fuso orario locale dell'applicazione.

Per modificare il fuso orario locale per un cluster database, imposta il parametro del fuso orario su uno dei valori supportati. Imposti questo parametro nel gruppo di parametri del tuo cluster database.

- Per Aurora MySQL, il nome di questo parametro è `time_zone`. Per informazioni sulle best practice per l'impostazione del parametro `time_zone`, consulta Ottimizzazione delle operazioni di timestamp.
- Per Aurora PostgreSQL, il nome di questo parametro è `timezone`.

Quando imposti il parametro di fuso orario per un cluster database, tutte le istanze nel cluster database cambiano per utilizzare il nuovo fuso orario locale. In alcuni casi, è possibile che altri cluster database Aurora utilizzino lo stesso gruppo di parametri del cluster. In tal caso, tutte le istanze in tali cluster database cambiano per utilizzare il nuovo fuso orario locale. Per ulteriori informazioni sui parametri a livello di cluster, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora.

Dopo aver impostato il fuso orario locale, tutte le nuove connessioni al database riflettono la modifica. In alcuni casi, potresti avere connessioni aperte al database quando modifichi il fuso orario locale. In tal caso, non vedrai l'aggiornamento del fuso orario locale finché non avrai chiuso la connessione e aperto una nuova connessione.

Se si esegue la replica tra AWS regioni, il cluster DB di origine della replica e la replica utilizzano gruppi di parametri diversi. I gruppi di parametri sono unici per una regione. AWS Per utilizzare lo

stesso fuso orario locale per ogni istanza, assicurati di impostare il parametro del fuso orario nei gruppi di parametri sia per l'origine della replica che per la replica.

Quando ripristini cluster database da una snapshot cluster database, il fuso orario locale è impostato su UTC. Puoi aggiornare il fuso orario impostandolo sul fuso orario locale dopo il completamento del ripristino. In alcuni casi, è possibile eseguire un ripristino point-in-time del cluster database. In tal caso, il fuso orario locale per il cluster database ripristinato corrisponde all'impostazione del fuso orario per il gruppo di parametri cluster database ripristinato.

Nella tabella seguente sono elencati alcuni dei valori che puoi usare per impostare il fuso orario locale. Per elencare tutti i fusi orari disponibili, puoi utilizzare le seguenti query SQL:

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

> **Note**
>
> Per alcuni fusi orari, i valori di tempo di alcuni intervalli di date potrebbero essere riportati in modo non corretto, come indicato nella tabella. Per i fusi orari australiani, l'abbreviazione restituita riporta un valore obsoleto, come indicato nella tabella.

Time zone (Fuso orario)	Note
`Africa/Harare`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 28/02/1903 21:49:40 GMT al 28/02/1903 21:55:48 GMT.
`Africa/Monrovia`	
`Africa/Nairobi`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 31/12/1939 21:30:00 GMT al 31/12/1959 21:15:15 GMT.
`Africa/Windhoek`	
`America/Bogota`	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 23/11/1914 04:56:16 GMT al 23/11/1914 04:56:20 GMT.

Time zone (Fuso orario)	Note
America/Caracas	
America/Chihuahua	
America/Cuiaba	
America/Denver	
America/Fortaleza	In alcuni casi, per un cluster database nella Regione Sud America (San Paolo), l'ora non viene visualizzata correttamente per un fuso orario del Brasile modificato di recente. In tal caso, reimposta il parametro del fuso orario del cluster database su America/Fortaleza .
America/Guatemala	
America/Halifax	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 27/10/1918 05:00:00 GMT al 31/10/1918 05:00:00 GMT.
America/Manaus	Se il cluster DB si trova nel fuso orario del Sud America (Cuiaba) e l'ora prevista non viene visualizzata correttamente per il fuso orario del Brasile modificato di recente, reimpostare il parametro fuso orario del cluster DB su America/Manaus .
America/Matamoros	
America/Monterrey	
America/Montevideo	
America/Noronha	
America/Phoenix	

Time zone (Fuso orario)	Note
America/Tijuana	
Asia/Ashgabat	
Asia/Baghdad	
Asia/Baku	
Asia/Bangkok	
Asia/Beirut	
Asia/Calcutta	
Asia/Kabul	
Asia/Karachi	
Asia/Kathmandu	
Asia/Muscat	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 31/12/1919 20:05:36 GMT al 31/12/1919 20:05:40 GMT.
Asia/Riyadh	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 13/03/1947 20:53:08 GMT al 31/12/1949 20:53:08 GMT.
Asia/Seoul	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/11/1904 15:30:00 GMT al 07/09/1945 15:00:00 GMT.
Asia/Shanghai	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 31/12/1927 15:54:08 GMT al 02/06/1940 16:00:00 GMT.
Asia/Singapore	
Asia/Taipei	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/09/1937 16:00:00 GMT al 29/09/1979 15:00:00 GMT.
Asia/Tehran	

Time zone (Fuso orario)	Note
Asia/Tokyo	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/09/1937 15:00:00 GMT al 31/12/1937 15:00:00 GMT.
Asia/Ulaanbaatar	
Atlantic/Azores	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 24/05/1911 01:54:32 GMT al 01/01/1912 01:54:32 GMT.
Australia/Adelaide	L'abbreviazione di questo fuso orario viene restituita come CST anziché ACDT/ACST.
Australia/Brisbane	L'abbreviazione di questo fuso orario viene restituita come EST anziché AEDT/AEST.
Australia/Darwin	L'abbreviazione di questo fuso orario viene restituita come CST anziché ACDT/ACST.
Australia/Hobart	L'abbreviazione di questo fuso orario viene restituita come EST anziché AEDT/AEST.
Australia/Perth	L'abbreviazione di questo fuso orario viene restituita come WST anziché AWDT/. AWST
Australia/Sydney	L'abbreviazione di questo fuso orario viene restituita come EST anziché AEDT/AEST.
Brazil/East	
Canada/Saskatchewan	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 27/10/1918 08:00:00 GMT al 31/10/1918 08:00:00 GMT.
Europe/Amsterdam	
Europe/Athens	

Time zone (Fuso orario)	Note
Europe/Dublin	
Europe/Helsinki	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 30/04/1921 22:20:08 GMT al 30/04/1921 22:20:11 GMT.
Europe/Paris	
Europe/Prague	
Europe/Sarajevo	
Pacific/Auckland	
Pacific/Guam	
Pacific/Honolulu	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 21/05/1933 11:30:00 GMT al 30/09/1945 11:30:00 GMT.
Pacific/Samoa	Questa impostazione del fuso orario potrebbe restituire valori non corretti dal 01/01/1911 11:22:48 GMT al 01/01/1950 11:30:00 GMT.
US/Alaska	
US/Central	
US/Eastern	
US/East-Indiana	
US/Pacific	
UTC	

Funzionalità supportate in Amazon Aurora by e Regione AWS Aurora DB engine

I motori di database compatibili con Aurora MySQL e PostgreSQL supportano diverse funzionalità e opzioni di Amazon Aurora e Amazon RDS. Il supporto varia a seconda delle versioni specifiche di ciascun motore del database e delle Regioni AWS. Per identificare il supporto e la disponibilità della versione del motore di database Aurora per una determinata funzionalità Regione AWS, è possibile utilizzare le seguenti sezioni.

Alcune di queste funzionalità sono esclusive di Aurora. Ad esempio, Aurora Serverless, i database globali Aurora e il supporto per l'integrazione con i servizi di AWS machine learning non sono supportati da Amazon RDS. Altre funzionalità, come Amazon RDS Proxy, sono supportate sia da Amazon Aurora che da Amazon RDS.

Regioni e motori DB supportati

- Convenzioni tabella
- Regioni supportate e motori Aurora DB per implementazioni Blue/Green
- Regioni supportate e motori Aurora DB per configurazioni di storage in cluster
- Regioni supportate e motori Aurora DB per i flussi di attività del database
- Regioni supportate e motori Aurora DB per l'esportazione dei dati del cluster in Amazon S3
- Regioni supportate e motori Aurora DB per l'esportazione di dati di snapshot su Amazon S3
- Regioni e motori DB supportati per i database globali Aurora
- Regioni supportate e motori Aurora DB per IAM l'autenticazione del database
- Regioni supportate e motori Aurora DB per l'autenticazione Kerberos
- Regioni e motori DB supportati per l'apprendimento automatico Aurora
- Regioni supportate e motori Aurora DB per Performance Insights
- Regioni supportate e motori Aurora DB per integrazioni zero-ETL con Amazon Redshift
- Regioni supportate e motori Aurora DB per Amazon RDS Proxy
- Regioni supportate e motori Aurora DB per l'integrazione di Secrets Manager
- Regioni supportate e motori Aurora DB per Aurora Serverless v2
- Aurora Serverless v1
- Regioni supportate e motori Aurora DB per RDS Data API
- Regioni supportate e motori Aurora DB per l'applicazione di patch senza tempi di inattività () ZDP

- Regioni e motori DB supportati per le funzionalità native del motore Aurora

Convenzioni tabella

Le tabelle nelle sezioni delle caratteristiche utilizzano i seguenti modelli per specificare i numeri di versione e il livello di supporto:

- Versione x.y – È supportata la sola versione specifica.
- Versione x.y e successive: sono supportate la versione specificata e tutte le relative versioni secondarie successive. Ad esempio, "versione 10.11 e successive" significa che sono supportate le versioni 10.11, 10.11.1 e 10.12.

Regioni supportate e motori Aurora DB per implementazioni Blue/Green

Un'implementazione blu/verde copia un ambiente di database di produzione in un ambiente di gestione temporanea separato e sincronizzato. Utilizzando Amazon RDS Blue/Green Deployments, puoi apportare modifiche al database nell'ambiente di staging senza influire sull'ambiente di produzione. Ad esempio, è possibile aggiornare la versione principale o secondaria del motore di database, modificare i parametri del database o apportare modifiche allo schema nell'ambiente di gestione temporanea. Quando sei pronto, puoi promuovere l'ambiente di gestione temporanea nel nuovo ambiente di database di produzione. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Implementazioni blu/verdi con Aurora My SQL

La funzionalità Blue/Green Deployments è disponibile per tutte le versioni di Aurora My in generale SQL Regioni AWS.

Implementazioni blu/verdi con Aurora Postgre SQL

Le seguenti regioni e versioni del motore sono disponibili per le implementazioni Blue/Green con Aurora Postgre. SQL

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Tutti Regioni AWS	Versione 16.1 e successive	Versione 15.4 e successive	Versione 14.9 e successive	Versione 13.12 e successive	Versione 12.16 e successive	Versione 11.21 e successive

Regioni supportate e motori Aurora DB per configurazioni di storage in cluster

Amazon Aurora dispone di due configurazioni di storage in cluster DB, Aurora I/O-Optimized e Aurora Standard. Per ulteriori informazioni, vedereConfigurazioni dell'archiviazione per i cluster database Amazon Aurora.

Aurora I/O-Optimized

Aurora I/O-Optimized è disponibile in tutto Regioni AWS per le seguenti versioni di Amazon Aurora:

- Aurora My SQL versione 3.03.1 e successive
- Aurora Postgre SQL versioni 16.1 e successive, 15.2 e successive, 14.7 e successive e 13.10 e successive

Aurora Standard

Aurora Standard è disponibile in tutte Regioni AWS per tutte le versioni di Aurora My e SQL Aurora Postgre. SQL

Regioni supportate e motori Aurora DB per i flussi di attività del database

Utilizzando i flussi di attività del database in Aurora, è possibile monitorare e impostare gli allarmi per l'attività di audit nel database Oracle in uso. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

I flussi di attività del database non sono supportati per le seguenti funzionalità:

- Aurora Serverless v1

- Aurora Serverless v2
- Babelfish per Aurora Postgre SQL

Argomenti

- Flussi di attività del database con Aurora My SQL
- Flussi di attività del database con Aurora Postgre SQL

Flussi di attività del database con Aurora My SQL

Le seguenti regioni e versioni del motore sono disponibili per i flussi di attività del database con SQL Aurora My.

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Stati Uniti orientali (Ohio)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Stati Uniti occidentali (California settentrionale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
US West (Oregon)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Africa (Città del Capo)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Hong Kong)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacific (Hyderabad)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Giacarta)	Tutte le versioni disponibili	Aurora versione 2.11 e successive

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Osaka-Locale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Seoul)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Singapore)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Sydney)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Asia Pacifico (Tokyo)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Canada (Centrale)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Canada occidentale (Calgary)	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile
Europa (Francoforte)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Irlanda)	Tutte le versioni disponibili	Aurora versione 2.11 e successive

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Europa (Londra)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Milano)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Parigi)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Spagna)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Stoccolma)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Europa (Zurigo)	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Medio Oriente () UAE	Tutte le versioni disponibili	Aurora versione 2.11 e successive
Sud America (San Paolo)	Tutte le versioni disponibili	Aurora versione 2.11 e successive

Flussi di attività del database con Aurora Postgre SQL

Le seguenti regioni e versioni del motore sono disponibili per i flussi di attività del database con SQL Aurora Postgre.

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Postgree 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Stati Uniti orientali (Virginia settentri onale)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Stati Uniti orientali (Ohio)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidenta li (Californ ia settentri onale)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
US West (Oregon)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Africa (Città del Capo)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hong Kong)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Postgree 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
						11.13 e successive
Asia Pacific (Hyderabad)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Giacarta)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Malesia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Postgree 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Asia Pacifico (Osaka-Locale)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Seoul)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Singapore)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Sydney)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Tokyo)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Postgree 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Canada (Centrale)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Canada occidentale (Calgary)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Francoforte)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Irlanda)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Londra)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Postgree 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Europa (Milano)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Parigi)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Spagna)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Stoccolma)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Europa (Zurigo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Postgree 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Medio Oriente (Bahrein)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Medio Oriente () UAE	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive
Sud America (San Paolo)	Versione 16.1 e successive	Versione 15.2 e successive	Tutte le versioni disponibili	Tutte le versioni disponibili	Tutte le versioni disponibili	Versione 11.9 e versione 11.13 e successive

Regioni supportate e motori Aurora DB per l'esportazione dei dati del cluster in Amazon S3

È possibile esportare i dati del cluster database Aurora in un bucket Amazon S3. Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni, consulta Esportazione dei dati del cluster database in Amazon S3.

L'esportazione dei dati del cluster in S3 è disponibile nelle seguenti Regioni AWS:

- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti orientali (Ohio)
- Stati Uniti occidentali (California settentrionale)
- US West (Oregon)

- Asia Pacifico (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacifico (Giacarta)
- Asia Pacifico (Melbourne)
- Asia Pacifico (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Cina (Ningxia)
- Europa (Francoforte)
- Europa (Irlanda)
- Europe (London)
- Europa (Parigi)
- Europa (Spagna)
- Europa (Stoccolma)
- Europa (Zurigo)
- Israele (Tel Aviv)
- Medio Oriente () UAE
- Sud America (San Paolo)

Argomenti

- Esportazione dei dati del cluster su S3 con Aurora My SQL
- Esportazione dei dati del cluster su S3 con Aurora Postgre SQL

Esportazione dei dati del cluster su S3 con Aurora My SQL

Tutte le versioni del SQL motore Aurora My attualmente disponibili supportano l'esportazione dei dati del cluster DB su Amazon S3. Per ulteriori informazioni sulle versioni, consulta le note di rilascio per Aurora My. SQL

Esportazione dei dati del cluster su S3 con Aurora Postgre SQL

Tutte le versioni del SQL motore Aurora Postgre attualmente disponibili supportano l'esportazione dei dati del cluster DB su Amazon S3. Per ulteriori informazioni sulle versioni, consulta le Note di rilascio per Aurora Postgre. SQL

Regioni supportate e motori Aurora DB per l'esportazione di dati di snapshot su Amazon S3

È possibile esportare i dati dello snapshot del cluster database Aurora in un bucket Amazon S3. È possibile esportare snapshot manuali e snapshot automatici di sistema. Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

L'esportazione di istantanee in S3 è disponibile in tutte le versioni Regioni AWS tranne quanto segue:

- Asia Pacifico (Malesia)
- AWS GovCloud (Stati Uniti orientali)
- AWS GovCloud (Stati Uniti occidentali)

Argomenti

- Esportazione dei dati delle istantanee su S3 con Aurora My SQL
- Esportazione dei dati delle istantanee su S3 con Aurora Postgre SQL

Esportazione dei dati delle istantanee su S3 con Aurora My SQL

Tutte le versioni di Aurora My SQL engine attualmente disponibili supportano l'esportazione di dati di snapshot del cluster DB su Amazon S3. Per ulteriori informazioni sulle versioni, consulta le note di rilascio per Aurora My. SQL

Esportazione dei dati delle istantanee su S3 con Aurora Postgre SQL

Tutte le versioni del SQL motore Aurora Postgre attualmente disponibili supportano l'esportazione di dati di snapshot del cluster DB su Amazon S3. Per ulteriori informazioni sulle versioni, consulta le Note di rilascio per Aurora Postgre. SQL

Regioni e motori DB supportati per i database globali Aurora

Un database globale Aurora è un singolo database che si estende su più database Regioni AWS, che consente letture globali a bassa latenza e disaster recovery da qualsiasi interruzione a livello regionale. Fornisce una tolleranza agli errori integrata per la distribuzione perché l'istanza DB non si basa su una singola Regione AWS, ma su più regioni e diverse zone di disponibilità. Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.

Argomenti

- Database globali Aurora con Aurora MySQL
- Database globali Aurora con Aurora PostgreSQL

Database globali Aurora con Aurora MySQL

Le regioni e le versioni del motore riportate di seguito sono disponibili per i database globali Aurora con Aurora MySQL.

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Stati Uniti orientali (Ohio)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
US West (Oregon)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Africa (Città del Capo)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Asia Pacifico (Hong Kong)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Asia Pacific (Hyderabad)	Versione 3.02.0 e successive	Versione 2.11.2 e successive
Asia Pacifico (Giacarta)	Versione 3.01.0 e successive	Versione 2.07.6 e successive
Asia Pacifico (Malesia)	Versione 3.04.0 e successive	Versione 2.07.6 e successive
Asia Pacifico (Melbourne)	Versione 3.03.0 e successive	Versione 2.07.6 e successive
Asia Pacifico (Mumbai)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Osaka-Locale)	Versione 3.01.0 e successive	Versione 2.07.3 e versioni successive
Asia Pacifico (Seoul)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Singapore)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Sydney)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Asia Pacifico (Tokyo)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Canada (Centrale)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Canada occidentale (Calgary)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Cina (Pechino)	Versione 3.01.0 e successive	Versione 2.07.2 e versioni successive
Cina (Ningxia)	Versione 3.01.0 e successive	Versione 2.07.2 e versioni successive
Europa (Francoforte)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Irlanda)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Londra)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Milano)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Europa (Parigi)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Spagna)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Europa (Stoccolma)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
Europa (Zurigo)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Israele (Tel Aviv)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Medio Oriente (Bahrein)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.02.0 e successive	Versione 2.07.6 e successive
Sud America (San Paolo)	Versione 3.01.0 e successive	Versione 2.07.1 e versioni successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
AWS GovCloud (Stati Uniti orientali)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.01.0 e successive	Versione 2.07.0 e versioni successive

Database globali Aurora con Aurora PostgreSQL

Le regioni e le versioni del motore riportate di seguito sono disponibili per i database globali Aurora con Aurora PostgreSQL.

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali (California	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
settentrionale)							
US West (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Seoul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Cina (Pechino)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Cina (Ningxia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Zurigo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Israele (Tel Aviv)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (Stati Uniti orientali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regioni supportate e motori Aurora DB per IAM l'autenticazione del database

Con l'autenticazione del IAM database in Aurora, puoi autenticarti nel tuo cluster DB utilizzando AWS Autenticazione del database Identity and Access Management (IAM). Questo metodo di autenticazione non richiede l'uso di una password quando si esegue la connessione al cluster database. Utilizzi invece un token di autenticazione. Per ulteriori informazioni, consulta Autenticazione del database IAM .

Argomenti

- IAMautenticazione del database con Aurora My SQL
- IAMautenticazione del database con Aurora Postgre SQL

IAMautenticazione del database con Aurora My SQL

IAMl'autenticazione del database con Aurora My SQL è disponibile in tutte le regioni per le seguenti versioni:

- Aurora My SQL 3 — Tutte le versioni disponibili
- Aurora My SQL 2 — Tutte le versioni disponibili

IAMautenticazione del database con Aurora Postgre SQL

IAMl'autenticazione del database con Aurora Postgre SQL è disponibile in tutte le regioni per le seguenti versioni del motore:

- Aurora Postgre SQL 16 — Tutte le versioni disponibili
- Aurora Postgre SQL 15 — Tutte le versioni disponibili
- Aurora Postgre SQL 14 — Tutte le versioni disponibili
- Aurora Postgre SQL 13 — Tutte le versioni disponibili
- Aurora Postgre SQL 12 — Tutte le versioni disponibili
- Aurora Postgre SQL 11 — Tutte le versioni disponibili

Regioni supportate e motori Aurora DB per l'autenticazione Kerberos

Se si utilizza l'autenticazione Kerberos con Aurora, è possibile supportare l'autenticazione esterna degli utenti del database mediante Kerberos e Microsoft Active Directory. L'utilizzo di Kerberos e Active Directory offre i vantaggi dell'autenticazione Single Sign-On centralizzata degli utenti dei database. Kerberos e Active Directory sono disponibili con AWS Directory Service for Microsoft Active Directory, una funzionalità di. AWS Directory Service Per ulteriori informazioni, consulta Autenticazione Kerberos.

Argomenti

- Autenticazione Kerberos con Aurora MySQL
- Autenticazione Kerberos con Aurora PostgreSQL

Autenticazione Kerberos con Aurora MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per l'autenticazione Kerberos con Aurora MySQL.

Regione	Aurora MySQL versione 3
Stati Uniti orientali (Virginia settentrionale)	Versione 3.03.0 e successive
Stati Uniti orientali (Ohio)	Versione 3.03.0 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.03.0 e successive
US West (Oregon)	Versione 3.03.0 e successive
Africa (Città del Capo)	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile
Asia Pacifico (Malesia)	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	Versione 3.03.0 e successive
Asia Pacifico (Osaka-Locale)	Non disponibile
Asia Pacifico (Seoul)	Versione 3.03.0 e successive
Asia Pacifico (Singapore)	Versione 3.03.0 e successive
Asia Pacifico (Sydney)	Versione 3.03.0 e successive
Asia Pacifico (Tokyo)	Versione 3.03.0 e successive
Canada (Centrale)	Versione 3.03.0 e successive
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Versione 3.03.0 e successive
Cina (Ningxia)	Versione 3.03.0 e successive
Europa (Francoforte)	Versione 3.03.0 e successive

Regione	Aurora MySQL versione 3
Europa (Irlanda)	Versione 3.03.0 e successive
Europa (Londra)	Versione 3.03.0 e successive
Europa (Milano)	Non disponibile
Europa (Parigi)	Versione 3.03.0 e successive
Europa (Spagna)	Non disponibile
Europa (Stoccolma)	Versione 3.03.0 e successive
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Medio Oriente (Bahrein)	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile
Sud America (San Paolo)	Versione 3.03.0 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.03.0 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.03.0 e successive

Autenticazione Kerberos con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per l'autenticazione Kerberos con Aurora PostgreSQL.

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
settentrionale)						
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
US West (Oregon)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Malesia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Seoul)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada (Centrale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Pechino)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Ningxia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Milano)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Parigi)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Spagna)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Zurigo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regioni e motori DB supportati per l'apprendimento automatico Aurora

Utilizzando l'apprendimento automatico di Amazon Aurora, puoi integrare il tuo cluster Aurora DB con uno dei seguenti servizi di AWS machine learning, a seconda delle tue esigenze. Ciascuno di essi supporta casi d'uso specifici di machine learning.

Amazon Bedrock è un servizio completamente gestito che rende disponibili i principali modelli di base delle aziende di intelligenza artificiale API tramite strumenti per sviluppatori per aiutare a creare e scalare applicazioni di intelligenza artificiale generativa.

Amazon Comprehend è un servizio di elaborazione del linguaggio naturale (NLP) utilizzato per estrarre informazioni dai documenti. L'uso di machine learning di Aurora con Amazon Comprehend consente di determinare il sentiment del testo nelle tabelle del database.

SageMaker L'intelligenza artificiale è un servizio di apprendimento automatico completo. I data scientist utilizzano Amazon SageMaker AI per creare, addestrare e testare modelli di machine learning per una varietà di attività di inferenza, come il rilevamento delle frodi. Utilizzando l'apprendimento automatico di Aurora con SageMaker AI, gli sviluppatori di database possono richiamare la funzionalità SageMaker AI nel codice. SQL

Non tutti Regioni AWS supportano tutti i servizi di machine learning. Solo alcuni Regioni AWS supportano l'apprendimento automatico di Aurora e quindi forniscono l'accesso a questi servizi da un cluster Aurora DB. Anche il processo di integrazione per machine learning di Aurora differisce dal motore di database. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora.

Argomenti

- Apprendimento automatico Aurora con Aurora My SQL
- Apprendimento automatico Aurora con Aurora Postgre SQL

Apprendimento automatico Aurora con Aurora My SQL

Amazon Bedrock è supportato solo su Aurora SQL My versione 3.06 e successive. Per informazioni sulla disponibilità regionale per Amazon Bedrock, consulta Model support by Regione AWS nella Amazon Bedrock User Guide.

L'apprendimento automatico Aurora con Amazon Comprehend e SageMaker Amazon AI è supportato per SQL Aurora My Regioni AWS nell'elenco riportato nella tabella. Oltre a disporre della tua versione di Aurora MySQL, Regione AWS devono supportare anche il servizio che desideri utilizzare. Per un

elenco delle aree Regioni AWS in cui è disponibile Amazon SageMaker AI, consulta gli endpoint e le quote di Amazon SageMaker AI nel. Riferimenti generali di Amazon Web Services Per un elenco delle aree Regioni AWS in cui è disponibile Amazon Comprehend, consulta gli endpoint e le quote di Amazon Comprehend nel. Riferimenti generali di Amazon Web Services

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Stati Uniti orientali (Virginia settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Stati Uniti orientali (Ohio)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
US West (Oregon)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Africa (Città del Capo)	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacific (Hyderabad)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Giacarta)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Malesia)	Versione 3.04.0 e successive	Non disponibile
Asia Pacifico (Melbourne)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Mumbai)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Osaka-Locale)	Versione 3.01.0 e successive	Versione 2.07.3 e versioni successive

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Asia Pacifico (Seoul)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Singapore)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Sydney)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Asia Pacifico (Tokyo)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Canada (Centrale)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Canada occidentale (Calgary)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Cina (Pechino)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Cina (Ningxia)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Francoforte)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Irlanda)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Londra)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Milano)	Non disponibile	Non disponibile
Europa (Parigi)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Europa (Spagna)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Stoccolma)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Europa (Zurigo)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Israele (Tel Aviv)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Medio Oriente (Bahrein)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Medio Oriente () UAE	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
Sud America (San Paolo)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.01.0 e successive	Versione 2.07 e versioni successive

Apprendimento automatico Aurora con Aurora Postgre SQL

Per informazioni sul supporto delle versioni per Amazon Bedrock su Aurora SQL Postgre, consulta. Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock

L'apprendimento automatico Aurora con Amazon Comprehend e SageMaker Amazon AI è supportato per SQL Aurora Regioni AWS Postgre nell'elenco riportato nella tabella. Oltre a disporre della tua versione di Aurora PostgreSQL, Regione AWS devono supportare anche il servizio che desideri utilizzare. Per un elenco delle aree Regioni AWS in cui è disponibile Amazon SageMaker AI, consulta gli endpoint e le quote di Amazon SageMaker AI nel. Riferimenti generali di Amazon Web Services

Per un elenco delle aree Regioni AWS in cui è disponibile Amazon Comprehend, consulta gli endpoint e le quote di Amazon Comprehend nel. Riferimenti generali di Amazon Web Services

Le seguenti regioni e versioni del motore sono disponibili per l'apprendimento automatico di Aurora con Aurora Postgre. SQL

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Stati Uniti orientali (Virginia settentrionale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Stati Uniti orientali (Ohio)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
US West (Oregon)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Africa (Città del Capo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e	Versione 13.3 e	Versione 12.4 e	Versione 11.9, 11.12

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
(Hong Kong)			versioni successive	versioni successive	versioni successive	e versioni successive
Asia Pacific (Hyderabad)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Giacarta)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Malesia)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Melbourne)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Mumbai)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Osaka-Locale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Asia Pacifico (Seoul)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Singapore)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Sydney)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Asia Pacifico (Tokyo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Canada (Centrale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Canada occidentale (Calgary)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Cina (Pechino)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Cina (Ningxia)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Francofo rte)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Irlanda)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Londra)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Milano)	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Parigi)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Spagna)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
Europa (Stoccolma)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Europa (Zurigo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Israele (Tel Aviv)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Medio Oriente (Bahrein)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Medio Oriente () UAE	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
Sud America (San Paolo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Regione	Aurora Postgree 16 SQL	Aurora Postgree 15 SQL	Aurora Postgree 14 SQL	Aurora Poster 13 SQL	Aurora Poster 12 SQL	Aurora Poster 11 SQL
AWS GovCloud (Stati Uniti occidentali)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.3 e versioni successive	Versione 12.4 e versioni successive	Versione 11.9, 11.12 e versioni successive

Regioni supportate e motori Aurora DB per Performance Insights

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
Se non si intraprende alcuna azione, DB che raggruppano in cluster utilizzando Performance Insights utilizzeranno per impostazione predefinita la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Performance Insights espande le sue potenzialità grazie alle funzionalità di monitoraggio esistenti di Amazon RDS a supporto delle operazioni di analisi delle prestazioni del database. Il pannello di controllo di Performance Insights consente di visualizzare il carico del database sull'istanza database Amazon RDS e filtrare il carico in base alle attese, alle istruzioni, agli host o agli utenti. Per ulteriori informazioni, consulta Panoramica di Performance Insights su Amazon Aurora.

Per informazioni sull'assistenza alla regione, al motore di database e alla classe di istanza per le funzionalità Performance Insights, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights.

Argomenti

- Performance Insights con Aurora MySQL
- Performance Insights con Aurora PostgreSQL
- Performance Insights con Aurora Serverless

Performance Insights con Aurora MySQL

> **Note**
>
> Il supporto della versione del motore è diverso per Performance Insights con Aurora MySQL se la query parallela è attivata. Per ulteriori informazioni sulla query parallela, consultare Interrogazione parallela per Amazon Aurora My SQL.

Argomenti

- Performance Insights con Aurora MySQL e query parallela disattivata
- Performance Insights con Aurora MySQL e query parallela attivata

Performance Insights con Aurora MySQL e query parallela disattivata

Di seguito sono riportate le versioni di motore e le regioni disponibili per Approfondimenti sulle prestazioni con Aurora MySQL e la query parallela disattivata.

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni
US West (Oregon)	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Hong Kong)	Tutte le versioni	Tutte le versioni
Asia Pacific (Hyderabad)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Giacarta)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Malesia)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Osaka-Locale)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Seoul)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Tutte le versioni	Tutte le versioni
Cina (Pechino)	Tutte le versioni	Tutte le versioni

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Cina (Ningxia)	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni
Europa (Milano)	Tutte le versioni	Tutte le versioni
Europa (Parigi)	Tutte le versioni	Tutte le versioni
Europa (Spagna)	Tutte le versioni	Tutte le versioni
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni
Europa (Zurigo)	Tutte le versioni	Tutte le versioni
Israele (Tel Aviv)	Tutte le versioni	Tutte le versioni
Medio Oriente (Bahrein)	Tutte le versioni	Tutte le versioni
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni	Tutte le versioni
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni

Performance Insights con Aurora MySQL e query parallela attivata

Di seguito sono riportate le versioni di motore e le regioni disponibili per Approfondimenti sulle prestazioni con Aurora MySQL e la query parallela attivata.

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Non disponibile	Versione 2.09.0 e versioni successive
Stati Uniti orientali (Ohio)	Non disponibile	Versione 2.09.0 e versioni successive
Stati Uniti occidentali (California settentrionale)	Non disponibile	Versione 2.09.0 e versioni successive
US West (Oregon)	Non disponibile	Versione 2.09.0 e versioni successive
Africa (Città del Capo)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Hong Kong)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacific (Hyderabad)	Non disponibile	Tutte le versioni
Asia Pacifico (Giacarta)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Malesia)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Melbourne)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Mumbai)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Osaka-Locale)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Seoul)	Non disponibile	Versione 2.09.0 e versioni successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Singapore)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Sydney)	Non disponibile	Versione 2.09.0 e versioni successive
Asia Pacifico (Tokyo)	Non disponibile	Versione 2.09.0 e versioni successive
Canada (Centrale)	Non disponibile	Versione 2.09.0 e versioni successive
Canada occidentale (Calgary)	Non disponibile	Versione 2.09.0 e versioni successive
Cina (Pechino)	Non disponibile	Versione 2.09.0 e versioni successive
Cina (Ningxia)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Francoforte)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Irlanda)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Londra)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Milano)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Parigi)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Spagna)	Non disponibile	Versione 2.09.0 e versioni successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Stoccolma)	Non disponibile	Versione 2.09.0 e versioni successive
Europa (Zurigo)	Non disponibile	Versione 2.09.0 e versioni successive
Israele (Tel Aviv)	Non disponibile	Versione 2.09.0 e versioni successive
Medio Oriente (Bahrein)	Non disponibile	Versione 2.09.0 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Versione 2.09.0 e versioni successive
Sud America (San Paolo)	Non disponibile	Versione 2.09.0 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Versione 2.09.0 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Versione 2.09.0 e versioni successive

Performance Insights con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per Approfondimenti sulle prestazioni con Aurora PostgreSQL.

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Stati Uniti orientali (Virginia	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
settentrionale)							
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
US West (Oregon)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Hong Kong)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacific (Hyderabad)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asia Pacifico (Giacarta)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Malesia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Melbourne)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Osaka-Locale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Seoul)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Canada occidentale (Calgary)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Pechino)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Cina (Ningxia)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Francoforte)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Milano)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Europa (Parigi)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Spagna)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Stoccolma)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Europa (Zurigo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Israele (Tel Aviv)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Medio Oriente (Bahrein)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Medio Oriente (Emirati Arabi Uniti)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
Sud America (San Paolo)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
AWS GovCloud (Stati Uniti orientali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni
AWS GovCloud (Stati Uniti occidentali)	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni	Tutte le versioni

Performance Insights con Aurora Serverless

Aurora Serverless v2 supporta Performance Insights per tutte le versioni compatibili con MySQL e compatibili con PostgreSQL. Si consiglia di impostare la capacità minima su almeno 2 unità di capacità Aurora ()ACUs.

Aurora Serverless v1 non supporta Performance Insights.

Regioni supportate e motori Aurora DB per integrazioni zero-ETL con Amazon Redshift

Le integrazioni Zero-ETL di Amazon Aurora con Amazon Redshift sono una soluzione completamente gestita per rendere disponibili i dati transazionali in Amazon Redshift dopo la scrittura su un cluster Aurora. Per ulteriori informazioni, consulta Integrazioni Zero-ETL.

Argomenti

- Integrazioni Aurora MySQL zero-ETL
- Integrazioni Aurora PostgreSQL Zero-ETL

Integrazioni Aurora MySQL zero-ETL

Le seguenti regioni e versioni del motore sono disponibili per le integrazioni Aurora MySQL Zero-ETL con Amazon Redshift.

Regione	Aurora MySQL versione 3
Stati Uniti orientali (Virginia settentrionale)	Versione 3.05.2 e successive
Stati Uniti orientali (Ohio)	Versione 3.05.2 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.05.2 e successive
US West (Oregon)	Versione 3.05.2 e successive
Africa (Città del Capo)	Versione 3.05.2 e successive
Asia Pacifico (Hong Kong)	Versione 3.05.2 e successive
Asia Pacific (Hyderabad)	Versione 3.05.2 e successive
Asia Pacifico (Giacarta)	Versione 3.05.2 e successive
Asia Pacifico (Malesia)	Non supportato
Asia Pacifico (Melbourne)	Versione 3.05.2 e successive
Asia Pacifico (Mumbai)	Versione 3.05.2 e successive
Asia Pacifico (Osaka-Locale)	Versione 3.05.2 e successive
Asia Pacifico (Seoul)	Versione 3.05.2 e successive
Asia Pacifico (Singapore)	Versione 3.05.2 e successive
Asia Pacifico (Sydney)	Versione 3.05.2 e successive
Asia Pacifico (Tokyo)	Versione 3.05.2 e successive
Canada (Centrale)	Versione 3.05.2 e successive
Canada occidentale (Calgary)	Versione 3.05.2 e successive

Regione	Aurora MySQL versione 3
Cina (Pechino)	Versione 3.05.2 e successive
Cina (Ningxia)	Versione 3.05.2 e successive
Europa (Francoforte)	Versione 3.05.2 e successive
Europa (Irlanda)	Versione 3.05.2 e successive
Europa (Londra)	Versione 3.05.2 e successive
Europa (Milano)	Versione 3.05.2 e successive
Europa (Parigi)	Versione 3.05.2 e successive
Europa (Spagna)	Versione 3.05.2 e successive
Europa (Stoccolma)	Versione 3.05.2 e successive
Europa (Zurigo)	Versione 3.05.2 e successive
Israele (Tel Aviv)	Versione 3.05.2 e successive
Medio Oriente (Bahrein)	Versione 3.05.2 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.05.2 e successive
Sud America (San Paolo)	Versione 3.05.2 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

Integrazioni Aurora PostgreSQL Zero-ETL

Le seguenti regioni e versioni del motore sono disponibili per le integrazioni Aurora PostgreSQL Zero-ETL con Amazon Redshift.

Regione	Aurora PostgreSQL 16
Stati Uniti orientali (Ohio)	Versione 16.4 e successive
Stati Uniti orientali (Virginia settentrionale)	Versione 16.4 e successive
Stati Uniti occidentali (California settentrionale)	Versione 16.4 e successive
US West (Oregon)	Versione 16.4 e successive
Africa (Città del Capo)	Versione 16.4 e successive
Asia Pacifico (Hong Kong)	Versione 16.4 e successive
Asia Pacific (Hyderabad)	Versione 16.4 e successive
Asia Pacifico (Giacarta)	Versione 16.4 e successive
Asia Pacifico (Melbourne)	Versione 16.4 e successive
Asia Pacifico (Mumbai)	Versione 16.4 e successive
Asia Pacifico (Osaka-Locale)	Versione 16.4 e successive
Asia Pacifico (Seoul)	Versione 16.4 e successive
Asia Pacifico (Singapore)	Versione 16.4 e successive
Asia Pacifico (Sydney)	Versione 16.4 e successive
Asia Pacifico (Tokyo)	Versione 16.4 e successive
Canada (Centrale)	Versione 16.4 e successive
Canada occidentale (Calgary)	Versione 16.4 e successive
Cina (Pechino)	Versione 16.4 e successive
Cina (Ningxia)	Versione 16.4 e successive
Europa (Francoforte)	Versione 16.4 e successive

Regione	Aurora PostgreSQL 16
Europa (Irlanda)	Versione 16.4 e successive
Europa (Londra)	Versione 16.4 e successive
Europa (Milano)	Versione 16.4 e successive
Europa (Parigi)	Versione 16.4 e successive
Europa (Spagna)	Versione 16.4 e successive
Europa (Stoccolma)	Versione 16.4 e successive
Europa (Zurigo)	Versione 16.4 e successive
Israele (Tel Aviv)	Versione 16.4 e successive
Medio Oriente (Bahrein)	Versione 16.4 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 16.4 e successive
Sud America (San Paolo)	Versione 16.4 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

Regioni supportate e motori Aurora DB per Amazon RDS Proxy

Amazon RDS Proxy è un proxy di database completamente gestito e ad alta disponibilità che rende le applicazioni più scalabili mediante il pooling e la condivisione di connessioni di database consolidate. Per ulteriori informazioni su RDS Proxy, consulta Proxy Amazon RDS per Aurora.

Argomenti

- Amazon RDS Proxy con Aurora MySQL
- Amazon RDS Proxy con Aurora PostgreSQL

Amazon RDS Proxy con Aurora MySQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per RDS Proxy con Aurora MySQL.

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Stati Uniti orientali (Ohio)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
US West (Oregon)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Africa (Città del Capo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Hong Kong)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacific (Hyderabad)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Giacarta)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Malesia)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Melbourne)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Mumbai)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Asia Pacifico (Osaka-Locale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Seoul)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Singapore)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Sydney)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Tailandia)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Asia Pacifico (Tokyo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Canada (Centrale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Canada occidentale (Calgary)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Cina (Pechino)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Cina (Ningxia)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Francoforte)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Irlanda)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Londra)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Milano)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Parigi)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Spagna)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Stoccolma)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Europa (Zurigo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Israele (Tel Aviv)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Medio Oriente (Bahrein)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Messico (centrale)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
Sud America (San Paolo)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.01.0 e successive	Versione 2.07 e versione 2.11 e successive

Amazon RDS Proxy con Aurora PostgreSQL

Di seguito sono riportate le versioni di motore e le regioni disponibili per RDS Proxy con Aurora PostgreSQL.

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti orientali (Ohio)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Stati Uniti occidentali (California settentrionale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
US West (Oregon)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Africa (Città del Capo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Hong Kong)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacific (Hyderabad)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Giacarta)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Malesia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Melbourne)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Mumbai)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Osaka-Locale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Seoul)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Singapore)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Sydney)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Asia Pacifico (Tailandia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asia Pacifico (Tokyo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada (Centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Canada occidentale (Calgary)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Cina (Pechino)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Cina (Ningxia)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Francoforte)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Irlanda)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Londra)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Milano)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Parigi)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Spagna)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Europa (Stoccolma)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europa (Zurigo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Israele (Tel Aviv)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Bahrein)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regione	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Messico (centrale)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
Sud America (San Paolo)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 17.4 e successive	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.4 e successive	Versione 12.8 e successive	Versione 11.9 e versione 11.13 e successive

Regioni supportate e motori Aurora DB per l'integrazione di Secrets Manager

Con AWS Secrets Manager, è possibile sostituire le credenziali codificate nel codice, incluse le password del database, con una chiamata a API Secrets Manager per recuperare il segreto a livello di codice. Per ulteriori informazioni su Secrets Manager, vedere AWS Secrets Manager Guida per l'utente.

Puoi specificare che Amazon Aurora gestisca la password dell'utente master in Secrets Manager per un cluster database Aurora. Aurora genera la password, la memorizza in Secrets Manager e la ruota regolarmente. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager.

L'integrazione con Secrets Manager è disponibile in tutti Regioni AWS.

Regioni supportate e motori Aurora DB per Aurora Serverless v2

Aurora Serverless v2 è una funzionalità di auto-scaling on-demand progettata per offrire un approccio conveniente all'esecuzione di carichi di lavoro intermittenti o imprevedibili su Amazon Aurora. Si dimensiona automaticamente la capacità in base alle esigenze dell'applicazione. La scalabilità è più rapida e granulare rispetto a Aurora Serverless v1. Con Aurora Serverless v2, ogni cluster può contenere un'istanza Writer DB e più istanze Reader DB. È possibile combinare Aurora Serverless v2 e istanze DB tradizionali con provisioning all'interno dello stesso cluster. Per ulteriori informazioni, consulta Utilizzo Aurora Serverless v2.

Argomenti

- Aurora Serverless v2 con Aurora MySQL
- Aurora Serverless v2 con Aurora PostgreSQL

Aurora Serverless v2 con Aurora MySQL

Le seguenti regioni e versioni del motore sono disponibili per Aurora Serverless v2 con Aurora MySQL.

Regione	Aurora MySQL versione 3
Stati Uniti orientali (Virginia settentrionale)	Versione 3.02.0 e successive

Regione	Aurora MySQL versione 3
Stati Uniti orientali (Ohio)	Versione 3.02.0 e successive
Stati Uniti occidentali (California settentrionale)	Versione 3.02.0 e successive
US West (Oregon)	Versione 3.02.0 e successive
Africa (Città del Capo)	Versione 3.02.0 e successive
Asia Pacifico (Hong Kong)	Versione 3.02.0 e successive
Asia Pacific (Hyderabad)	Versione 3.02.3 e successive
Asia Pacifico (Giacarta)	Versione 3.02.0 e successive
Asia Pacifico (Malesia)	Versioni 3.04.3, 3.05.2, 3.06.1, 3.07.1 e successive
Asia Pacifico (Melbourne)	Versione 3.02.3 e successive
Asia Pacifico (Mumbai)	Versione 3.02.0 e successive
Asia Pacifico (Osaka-Locale)	Versione 3.02.0 e successive
Asia Pacifico (Seoul)	Versione 3.02.0 e successive
Asia Pacifico (Singapore)	Versione 3.02.0 e successive
Asia Pacifico (Sydney)	Versione 3.02.0 e successive
Asia Pacifico (Tokyo)	Versione 3.02.0 e successive
Canada (Centrale)	Versione 3.02.0 e successive
Canada occidentale (Calgary)	Versioni 3.04.0 e successive
Cina (Pechino)	Versione 3.02.2 e successive
Cina (Ningxia)	Versione 3.02.2 e successive
Europa (Francoforte)	Versione 3.02.0 e successive

Regione	Aurora MySQL versione 3
Europa (Irlanda)	Versione 3.02.0 e successive
Europa (Londra)	Versione 3.02.0 e successive
Europa (Milano)	Versione 3.02.0 e successive
Europa (Parigi)	Versione 3.02.0 e successive
Europa (Spagna)	Versione 3.02.3 e successive
Europa (Stoccolma)	Versione 3.02.0 e successive
Europa (Zurigo)	Versione 3.02.3 e successive
Israele (Tel Aviv)	Versioni 3.02.3 e successive, 3.03.1 e successive
Medio Oriente (Bahrein)	Versione 3.02.0 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 3.02.3 e successive
Sud America (San Paolo)	Versione 3.02.0 e successive
AWS GovCloud (Stati Uniti orientali)	Versione 3.02.2 e successive
AWS GovCloud (Stati Uniti occidentali)	Versione 3.02.2 e successive

I limiti ACU superiore e inferiore per Aurora Serverless v2 la capacità può variare a seconda della versione del motore. Per informazioni dettagliate, consultare Aurora Serverless v2 capacità.

Aurora Serverless v2 con Aurora PostgreSQL

Le seguenti regioni e versioni del motore sono disponibili per Aurora Serverless v2 con Aurora PostgreSQL.

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Stati Uniti orientali (Virginia settentrionale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Stati Uniti orientali (Ohio)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Stati Uniti occidentali (California settentrionale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
US West (Oregon)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Africa (Città del Capo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Hong Kong)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacific (Hyderabad)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Asia Pacifico (Giacarta)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Malesia)	Versione 16.1 e successive	Versione 15.4 e successive	Versione 14.6, 14.9 e successive	Versione 13.9, 13.12 e successive

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia Pacifico (Melbourne)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Asia Pacifico (Mumbai)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Osaka-Locale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Seoul)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Singapore)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Sydney)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Asia Pacifico (Tokyo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Canada (Centrale)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Canada occidentale (Calgary)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.6, 14.8 e successive	Versione 13.9, 13.11 e successive

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Cina (Pechino)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Cina (Ningxia)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Francoforte)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Irlanda)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Londra)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Milano)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Parigi)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Europa (Spagna)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Europa (Stoccolma)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Europa (Zurigo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Israele (Tel Aviv)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Medio Oriente (Bahrein)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
Medio Oriente (Emirati Arabi Uniti)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.6 e successive	Versione 13.9 e successive
Sud America (San Paolo)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
AWS GovCloud (Stati Uniti orientali)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive
AWS GovCloud (Stati Uniti occidentali)	Versione 16.1 e successive	Versione 15.2 e successive	Versione 14.3 e versioni successive	Versione 13.6 e versioni successive

I limiti ACU superiore e inferiore per Aurora Serverless v2 la capacità può variare a seconda della versione del motore. Per informazioni dettagliate, consulta Aurora Serverless v2 capacità.

Aurora Serverless v1

Important

AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Consigliamo vivamente di aggiornarne uno Aurora Serverless v1 Cluster DB a Aurora Serverless v2 prima

di tale data. L'aggiornamento può comportare una modifica del numero di versione principale del motore di database. Pertanto, è importante pianificare, testare e implementare questo passaggio prima della end-of-life data. A partire dall'8 gennaio 2025, i clienti non saranno più in grado di crearne di nuovi Aurora Serverless v1 cluster o istanze con AWS Management Console o la CLI. Per informazioni sul processo di migrazione, consulta. Aggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2

Aurora Serverless v2 scala più rapidamente e in modo più granulare. Aurora Serverless v2 ha anche una maggiore compatibilità con altre funzionalità di Aurora, come le istanze Reader DB. Puoi saperne di più Aurora Serverless v2 nella sezione Utilizzo Aurora Serverless v2.

Aurora Serverless v1 è una funzionalità di auto-scaling su richiesta progettata per offrire un approccio conveniente all'esecuzione di carichi di lavoro intermittenti o imprevedibili su Amazon Aurora. Avvia, arresta e dimensiona automaticamente la capacità in base alle esigenze dell'applicazione usando una singola istanza database in ogni cluster. Per ulteriori informazioni, consulta Usare Amazon Aurora Serverless v1.

Argomenti

- Aurora Serverless v1 con Aurora MySQL
- Aurora Serverless v1 con Aurora PostgreSQL

Aurora Serverless v1 con Aurora MySQL

Le seguenti regioni e versioni del motore sono disponibili per Aurora Serverless v1 con Aurora MySQL.

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	Non disponibile	Versione 2.11.4
Stati Uniti orientali (Ohio)	Non disponibile	Versione 2.11.4
Stati Uniti occidentali (California settentrionale)	Non disponibile	Versione 2.11.4
US West (Oregon)	Non disponibile	Versione 2.11.4

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Africa (Città del Capo)	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile	Non disponibile
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Non disponibile	Versione 2.11.4
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile
Asia Pacifico (Seoul)	Non disponibile	Versione 2.11.4
Asia Pacifico (Singapore)	Non disponibile	Versione 2.11.4
Asia Pacifico (Sydney)	Non disponibile	Versione 2.11.4
Asia Pacifico (Tokyo)	Non disponibile	Versione 2.11.4
Canada (Centrale)	Non disponibile	Versione 2.11.4
Canada occidentale (Calgary)	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Versione 2.11.4
Europa (Francoforte)	Non disponibile	Versione 2.11.4
Europa (Irlanda)	Non disponibile	Versione 2.11.4
Europa (Londra)	Non disponibile	Versione 2.11.4
Europa (Milano)	Non disponibile	Non disponibile

Regione	Aurora MySQL versione 3	Aurora MySQL versione 2
Europa (Parigi)	Non disponibile	Versione 2.11.4
Europa (Spagna)	Non disponibile	Non disponibile
Europa (Stoccolma)	Non disponibile	Non disponibile
Europa (Zurigo)	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Non disponibile
Sud America (San Paolo)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile

Aurora Serverless v1 con Aurora PostgreSQL

Le seguenti regioni e versioni del motore sono disponibili per Aurora Serverless v1 con Aurora PostgreSQL.

Regione	Aurora PostgreSQL 13
Stati Uniti orientali (Virginia settentrionale)	Versione 13.12
Stati Uniti orientali (Ohio)	Versione 13.12
Stati Uniti occidentali (California settentrionale)	Versione 13.12
US West (Oregon)	Versione 13.12

Regione	Aurora PostgreSQL 13
Africa (Città del Capo)	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile
Asia Pacifico (Malesia)	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	Versione 13.12
Asia Pacifico (Osaka-Locale)	Non disponibile
Asia Pacifico (Seoul)	Versione 13.12
Asia Pacifico (Singapore)	Versione 13.12
Asia Pacifico (Sydney)	Versione 13.12
Asia Pacifico (Tokyo)	Versione 13.12
Canada (Centrale)	Versione 13.12
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Non disponibile
Cina (Ningxia)	Non disponibile
Europa (Francoforte)	Versione 13.12
Europa (Irlanda)	Versione 13.12
Europa (Londra)	Versione 13.12
Europa (Milano)	Non disponibile

Regione	Aurora PostgreSQL 13
Europa (Parigi)	Versione 13.12
Europa (Spagna)	Non disponibile
Europa (Stoccolma)	Non disponibile
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Medio Oriente (Bahrein)	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile
Sud America (San Paolo)	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

Regioni supportate e motori Aurora DB per RDS Data API

RDS Data API (Data API) fornisce un'interfaccia di servizi Web a un cluster Amazon Aurora DB. Anziché gestire le connessioni al database dalle applicazioni client, è possibile eseguire comandi SQL su un endpoint HTTPS. Per ulteriori informazioni, consulta Utilizzo dell'API dati Amazon RDS.

Argomenti

- API dati con Aurora PostgreSQL Serverless v2 e provisioning
- API dati con Aurora MySQL Serverless v2 e provisioning
- API dati con Aurora PostgreSQL Serverless v1
- API dati con Aurora MySQL Serverless v1

API dati con Aurora PostgreSQL Serverless v2 e provisioning

Le seguenti regioni e versioni del motore sono disponibili per Data API con Aurora PostgreSQL Serverless v2 e cluster DB con provisioning.

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Stati Uniti orientali (Virginia settentrionale)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Stati Uniti orientali (Ohio)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Stati Uniti occidentali (California settentrionale)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
US West (Oregon)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Africa (Città del Capo)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Hong Kong)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Malesia)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asia Pacifico (Osaka-Locale)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Seoul)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Singapore)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Sydney)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Asia Pacifico (Tokyo)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Canada (Centrale)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Canada occidentale (Calgary)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Pechino)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Francoforte)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Irlanda)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Londra)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Milano)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive

Regione	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Europa (Parigi)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Spagna)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Europa (Stoccolma)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Europa (Zurigo)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Medio Oriente (Emirati Arabi Uniti)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
Sud America (San Paolo)	Versione 16.1 e successive	Versione 15.3 e successive	Versione 14.8 e successive	Versione 13.11 e successive
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile	Non disponibile	Non disponibile

API dati con Aurora MySQL Serverless v2 e provisioning

Le seguenti regioni e versioni del motore sono disponibili per Data API con Aurora MySQL Serverless v2 e cluster DB con provisioning.

Regione	Aurora MySQL versione 3			
Stati Uniti orientali (Ohio)	Versione 3.07 e successive			
Stati Uniti orientali (Virginia settentrionale)	Versione 3.07 e successive			
Stati Uniti occidentali (California settentrionale)	Versione 3.07 e successive			
US West (Oregon)	Versione 3.07 e successive			
Africa (Città del Capo)	Versione 3.07 e successive			
Asia Pacifico (Hong Kong)	Versione 3.07 e successive			
Asia Pacific (Hyderabad)	Non disponibile			
Asia Pacifico (Giacarta)	Versione 3.07 e successive			
Asia Pacifico (Malesia)	Versione 3.07 e successive			
Asia Pacifico (Melbourne)	Non disponibile			
Asia Pacifico (Mumbai)	Versione 3.07 e successive			

Regione	Aurora MySQL versione 3			
Asia Pacifico (Osaka-Locale)	Versione 3.07 e successive			
Asia Pacifico (Seoul)	Versione 3.07 e successive			
Asia Pacifico (Singapore)	Versione 3.07 e successive			
Asia Pacifico (Sydney)	Versione 3.07 e successive			
Asia Pacifico (Tokyo)	Versione 3.07 e successive			
Canada (Centrale)	Versione 3.07 e successive			
Canada occidentale (Calgary)	Non disponibile			
Cina (Pechino)	Non disponibile			
Cina (Ningxia)	Non disponibile			
Europa (Francoforte)	Versione 3.07 e successive			
Europa (Irlanda)	Versione 3.07 e successive			
Europa (Londra)	Versione 3.07 e successive			
Europa (Milano)	Versione 3.07 e successive			

Regione	Aurora MySQL versione 3			
Europa (Parigi)	Versione 3.07 e successive			
Europa (Spagna)	Non disponibile			
Europa (Stoccolma)	Versione 3.07 e successive			
Europa (Zurigo)	Non disponibile			
Israele (Tel Aviv)	Non disponibile			
Medio Oriente (Bahrein)	Versione 3.07 e successive			
Medio Oriente (Emirati Arabi Uniti)	Versione 3.07 e successive			
Sud America (San Paolo)	Versione 3.07 e successive			
AWS GovCloud (Stati Uniti orientali)	Non disponibile			
AWS GovCloud (Stati Uniti occidentali)	Non disponibile			

API dati con Aurora PostgreSQL Serverless v1

Le seguenti regioni e versioni del motore sono disponibili per Data API con Aurora PostgreSQL Serverless v1.

Regione	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Stati Uniti orientali (Virginia settentrionale)	Versione 13.9	Versione 11.18
Stati Uniti orientali (Ohio)	Versione 13.9	Versione 11.18
Stati Uniti occidentali (California settentrionale)	Versione 13.9	Versione 11.18
US West (Oregon)	Versione 13.9	Versione 11.18
Africa (Città del Capo)	Non disponibile	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile	Non disponibile
Asia Pacifico (Malesia)	Non disponibile	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile	Non disponibile
Asia Pacifico (Mumbai)	Versione 13.9	Versione 11.18
Asia Pacifico (Osaka-Locale)	Non disponibile	Non disponibile
Asia Pacifico (Seoul)	Versione 13.9	Versione 11.18
Asia Pacifico (Singapore)	Versione 13.9	Versione 11.18
Asia Pacifico (Sydney)	Versione 13.9	Versione 11.18
Asia Pacifico (Tokyo)	Versione 13.9	Versione 11.18
Canada (Centrale)	Versione 13.9	Versione 11.18
Cina (Pechino)	Non disponibile	Non disponibile
Cina (Ningxia)	Non disponibile	Non disponibile

Regione	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Europa (Francoforte)	Versione 13.9	Versione 11.18
Europa (Irlanda)	Versione 13.9	Versione 11.18
Europa (Londra)	Versione 13.9	Versione 11.18
Europa (Milano)	Non disponibile	Non disponibile
Europa (Parigi)	Versione 13.9	Versione 11.18
Europa (Spagna)	Non disponibile	Non disponibile
Europa (Stoccolma)	Non disponibile	Non disponibile
Europa (Zurigo)	Non disponibile	Non disponibile
Israele (Tel Aviv)	Non disponibile	Non disponibile
Medio Oriente (Bahrein)	Non disponibile	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile	Non disponibile
Sud America (San Paolo)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile	Non disponibile

API dati con Aurora MySQL Serverless v1

Le seguenti regioni e versioni del motore sono disponibili per Data API con Aurora MySQL Serverless v1.

Regione	Aurora MySQL versione 2
Stati Uniti orientali (Virginia settentrionale)	versione 2.11.3
Stati Uniti orientali (Ohio)	versione 2.11.3
Stati Uniti occidentali (California settentrionale)	versione 2.11.3
US West (Oregon)	versione 2.11.3
Africa (Città del Capo)	Non disponibile
Asia Pacifico (Hong Kong)	Non disponibile
Asia Pacific (Hyderabad)	Non disponibile
Asia Pacifico (Giacarta)	Non disponibile
Asia Pacifico (Malesia)	Non disponibile
Asia Pacifico (Melbourne)	Non disponibile
Asia Pacifico (Mumbai)	versione 2.11.3
Asia Pacifico (Osaka-Locale)	Non disponibile
Asia Pacifico (Seoul)	versione 2.11.3
Asia Pacifico (Singapore)	versione 2.11.3
Asia Pacifico (Sydney)	versione 2.11.3
Asia Pacifico (Tokyo)	versione 2.11.3
Canada (Centrale)	versione 2.11.3
Canada occidentale (Calgary)	Non disponibile
Cina (Pechino)	Non disponibile
Cina (Ningxia)	versione 2.11.3

Regione	Aurora MySQL versione 2
Europa (Francoforte)	versione 2.11.3
Europa (Irlanda)	versione 2.11.3
Europa (Londra)	versione 2.11.3
Europa (Milano)	Non disponibile
Europa (Parigi)	versione 2.11.3
Europa (Spagna)	Non disponibile
Europa (Stoccolma)	Non disponibile
Europa (Zurigo)	Non disponibile
Israele (Tel Aviv)	Non disponibile
Medio Oriente (Bahrein)	Non disponibile
Medio Oriente (Emirati Arabi Uniti)	Non disponibile
Sud America (San Paolo)	Non disponibile
AWS GovCloud (Stati Uniti orientali)	Non disponibile
AWS GovCloud (Stati Uniti occidentali)	Non disponibile

Regioni supportate e motori Aurora DB per l'applicazione di patch senza tempi di inattività () ZDP

L'esecuzione di aggiornamenti per i cluster database Aurora MySQL comporta la possibilità di un'interruzione quando il database viene arrestato e durante l'aggiornamento. Per impostazione predefinita, se si avvia l'aggiornamento mentre il database è occupato, si perdono tutte le connessioni e le transazioni elaborate dal cluster di database. Se si aspetta che il database sia inattivo per eseguire l'aggiornamento, potrebbe essere necessario attendere a lungo.

La funzionalità zero-downtime patching (ZDP) tenta, con il massimo impegno, di preservare le connessioni client tramite un aggiornamento Aurora. Se ZDP viene completata correttamente, le sessioni dell'applicazione vengono conservate e il motore di database si riavvia durante l'aggiornamento. Il riavvio del motore del database può causare un calo del throughput della durata di alcuni secondi fino a circa un minuto.

Per informazioni dettagliate sulle condizioni e sulle versioni del motore in cui ZDP è disponibile per SQL gli upgrade Aurora My, consulta. Utilizzo dell'applicazione di patch senza tempi di inattività

Per informazioni dettagliate sulle condizioni e sulle versioni del motore in cui ZDP è disponibile per gli aggiornamenti Aurora PostgreSQL, vedere. Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività

Regioni e motori DB supportati per le funzionalità native del motore Aurora

I motori di database Aurora supportano inoltre caratteristiche e funzionalità aggiuntive specifiche per Aurora. Alcune funzionalità native del motore potrebbero avere supporto limitato o privilegi limitati per un particolare motore di database Aurora, versione o regione.

Argomenti

- Funzionalità native del motore per Aurora My SQL
- Funzionalità native del motore per Aurora Postgre SQL

Funzionalità native del motore per Aurora My SQL

Di seguito sono elencate le funzionalità native del motore di Aurora My. SQL

- Auditing avanzato
- Backtrack
- Query di fault injection
- Inoltro di scrittura all'interno del cluster
- Query parallela

Funzionalità native del motore per Aurora Postgre SQL

Di seguito sono riportate le funzionalità native del motore di Aurora Postgre. SQL

- Babelfish
- Query di fault injection
- Gestione del piano di query

Connessioni endpoint Amazon Aurora

Amazon Aurora in genere utilizza un cluster di istanze database anziché una singola istanza. Ogni connessione viene gestita da un'istanza database specifica. Quando ti connetti a un cluster Aurora, il nome host e la porta specificati puntano a un handler intermedio chiamato endpoint. Aurora utilizza il meccanismo endpoint per astrarre queste connessioni. Pertanto, non è necessario codificare tutti i nomi host o scrivere una logica personalizzata per bilanciare e reindirizzare le connessioni quando alcune istanze DB non sono disponibili.

Per determinate attività di Aurora, le diverse istanze o i gruppi di istanze svolgono ruoli differenti. Ad esempio, l'istanza principale gestisce tutte le istruzioni Data Definition Language (DDL) e Data Manipulation Language (). DML e fino a 15 repliche di Aurora gestiscono il traffico di query di sola lettura.

Argomenti

- Tipi di endpoint di Aurora
- Visualizzazione degli endpoint per un cluster Aurora
- Come gli endpoint Aurora funzionano con elevata disponibilità
- Endpoint del cluster per Amazon Aurora
- Endpoint Reader per Amazon Aurora
- Endpoint di istanza per Amazon Aurora
- Endpoint personalizzati per Amazon Aurora

Tipi di endpoint di Aurora

Usando gli endpoint puoi associare ogni connessione all'istanza o al gruppo di istanze appropriato in base al caso d'uso. Ad esempio, per eseguire DDL istruzioni è possibile connettersi a qualsiasi istanza sia l'istanza principale. Per eseguire le interrogazioni, è possibile connettersi all'endpoint del lettore, con Aurora che esegue automaticamente il bilanciamento della connessione tra tutte le repliche Aurora. Per i cluster con istanze database con capacità o configurazioni diverse, puoi connetterti agli endpoint personalizzati associati a diversi sottoinsiemi di istanze database. Per la

diagnosi o l'ottimizzazione, puoi connetterti a un endpoint di istanza specifico per esaminare i dettagli su una determinata istanza database.

Un endpoint è rappresentato come uno specifico di Aurora URL che contiene un indirizzo host e una porta. Di seguito sono riportati i tipi di endpoint disponibili da un cluster database Aurora.

Endpoint del cluster

Connect all'istanza principale del cluster per sviluppare e testare applicazioni ed eseguire trasformazioni come `INSERT` istruzioni e DDL ETL operazioni. DML Trova la posizione dell'endpoint del cluster utilizzando AWS Management Console AWS CLI, o Amazon RDSAPI, come descritto inVisualizzazione degli endpoint per un cluster Aurora.

Per ulteriori informazioni sugli endpoint del cluster, consulta. Endpoint del cluster per Amazon Aurora

Endpoint di lettura

Eseguire interrogazioni. Aurora esegue automaticamente il bilanciamento della connessione tra tutte le repliche Aurora. Trova la posizione dell'endpoint del lettore utilizzando AWS Management Console AWS CLI, o Amazon RDSAPI, come descritto inVisualizzazione degli endpoint per un cluster Aurora.

Per ulteriori informazioni sugli endpoint dei lettori, consulta. Endpoint Reader per Amazon Aurora

Endpoint dell'istanza

Esamina i dettagli su un'istanza DB specifica per la diagnosi o l'ottimizzazione. Puoi trovare la posizione dell'endpoint dell'istanza per ciascuna istanza AWS Management Console solo nella pagina dei dettagli dell'istanza.

Per ulteriori informazioni sugli endpoint delle istanze, consulta. Endpoint di istanza per Amazon Aurora

Endpoint personalizzato

Connect a diversi sottoinsiemi di istanze DB sul cluster DB. Ciò è utile quando si hanno diverse capacità e configurazioni di istanze all'interno del cluster DB. Trova le posizioni degli endpoint personalizzate utilizzando AWS Management Console AWS CLI, o Amazon RDSAPI, come descritto inVisualizzazione degli endpoint per un cluster Aurora.

Per ulteriori informazioni sugli endpoint personalizzati, consulta. Endpoint personalizzati per Amazon Aurora

Endpoint Aurora Global Database Writer

Aurora Global Database dispone di un tipo speciale di endpoint che ha lo stesso scopo dell'endpoint cluster di un cluster Aurora autonomo. Gestisce sia le richieste di scrittura che quelle di lettura. Quando un cluster secondario diventa il nuovo cluster primario a causa di uno switchover o di un failover, Aurora commuta automaticamente questo endpoint in modo che punti all'endpoint del cluster del nuovo cluster primario, nell'altro. Regione AWS In questo modo, non è necessario codificare la AWS regione nella stringa di connessione dell'applicazione e non è necessario modificare la stringa di connessione quando cambia il layout del database globale. Aurora crea questo endpoint quando si configura un database globale Aurora, ad esempio scegliendo Aggiungi regione per un cluster Aurora in. AWS Management Console

Per informazioni su come utilizzare questo tipo di endpoint con Aurora Global Database, consulta. Connessione al database globale di Amazon Aurora

Visualizzazione degli endpoint per un cluster Aurora

Sebbene sia possibile trovare la posizione dell'endpoint dell'istanza solo nella pagina dei dettagli dell'istanza in AWS Management Console, è possibile utilizzare la console o Amazon RDS API per trovare le posizioni degli endpoint cluster, reader e personalizzati. AWS CLI

Console

Nella pagina dei dettagli dell'istanza relativa al cluster AWS Management Console, trova l'endpoint del cluster, l'endpoint reader ed eventuali endpoint personalizzati. L'endpoint dell'istanza viene visualizzato nella pagina dei dettagli di ogni istanza. Quando ti connetti, aggiungi il numero di porta associato, dopo i due punti, al nome dell'endpoint mostrato nella pagina di dettaglio.

AWS CLI

Con AWS CLI, puoi trovare lo scrittore, il lettore e tutti gli endpoint personalizzati nell'output del comando. describe-db-clusters Ad esempio, il comando seguente mostra gli attributi degli endpoint per tutti i cluster nella regione corrente. AWS

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

Amazon RDS API

Con Amazon RDSAPI, recuperi gli endpoint chiamando l'operazione D escribeDBCluster Endpoints.

Come gli endpoint Aurora funzionano con elevata disponibilità

Per i cluster in cui è importante l'elevata disponibilità, utilizza l'endpoint del cluster per connessioni di lettura/scrittura o per usi generici e l'endpoint di lettura per connessioni di sola lettura. Gli endpoint di scrittura e lettura gestiscono il failover delle istanze DB meglio degli endpoint di istanza. A differenza degli endpoint istanza, gli endpoint di scrittura e lettura modificano automaticamente l'istanza database a cui si connettono se un'istanza database nel cluster diventa non disponibile. Per ulteriori informazioni sugli endpoint cluster e reader, vedere e. Endpoint del cluster per Amazon Aurora Endpoint Reader per Amazon Aurora

In caso di errore dell'istanza database primaria di un cluster database, Aurora esegue automaticamente il failover su una nuova istanza database primaria. Questa operazione viene eseguita promuovendo una replica Aurora esistente in una nuova istanza database primaria oppure creando una nuova istanza database primaria. Se si verifica un failover, è possibile utilizzare l'endpoint del cluster per riconnettersi all'istanza DB primaria appena promossa o creata oppure utilizzare l'endpoint reader per riconnettersi a una delle repliche Aurora nel cluster DB. Durante un failover, l'endpoint di lettura potrebbe dirigere le connessioni alla nuova istanza database primaria di un cluster database per un breve periodo di tempo dopo che una replica di Aurora viene promossa a nuova istanza database primaria.

Se progetti la tua logica applicativa per gestire le connessioni agli endpoint di istanza, puoi rilevare a livello di codice o manualmente il set risultante di istanze database disponibili nel cluster database. Utilizzo dell'describe-db-clusters AWS CLI comando o DescribeDBClustersRDSAPIoperazione per trovare il cluster DB e gli endpoint dei lettori, le istanze DB, se le istanze DB sono lettori e i relativi livelli di promozione. Puoi quindi confermare le classi di istanza dopo il failover e collegarti a un endpoint di istanza appropriato.

Per ulteriori informazioni sui failover, consulta Tolleranza ai guasti di un cluster DB Aurora.

Per ulteriori informazioni sull'alta disponibilità in Amazon Aurora, consulta. Elevata disponibilità di Amazon Aurora

Endpoint del cluster per Amazon Aurora

Per endpoint del cluster (o endpoint di scrittura) si intende un endpoint per un cluster di database Aurora che si connette all'istanza database primaria corrente di quel cluster. Questo endpoint è l'unico in grado di eseguire operazioni di scrittura come DDL le istruzioni. Per questo motivo, l'endpoint del cluster è quello a cui ti connetti quando imposti un cluster per la prima volta o quando il cluster contiene una sola istanza database.

Ciascun cluster database Aurora ha un endpoint del cluster e un'istanza database primaria.

L'endpoint del cluster viene utilizzato per tutte le operazioni di scrittura sul cluster DB, inclusi inserimenti, aggiornamenti, eliminazioni e modifiche. DDL Puoi anche utilizzare l'endpoint del cluster per le operazioni di lettura, come ad esempio le query.

L'endpoint del cluster fornisce un failover del cluster per connessioni di lettura-scrittura al cluster database. In caso di errore dell'istanza database primaria corrente di un cluster database, Aurora esegue automaticamente il failover su una nuova istanza database primaria. Durante un failover, il cluster database continua a servire le richieste di connessione all'endpoint del cluster dalla nuova istanza database primaria, riducendo al minimo l'interruzione del servizio.

L'esempio seguente illustra un endpoint del cluster per un cluster Aurora My DB. SQL

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

Ogni cluster Aurora dispone di un singolo endpoint cluster integrato, il cui nome e altri attributi sono gestiti da Aurora. Non puoi creare, eliminare o modificare questo tipo di endpoint.

L'endpoint del cluster viene utilizzato per amministrare il cluster, eseguire operazioni di estrazione, trasformazione, caricamento (ETL) o sviluppare e testare applicazioni. L'endpoint del cluster si connette all'istanza primaria del cluster. L'istanza principale è l'unica istanza DB in cui è possibile creare tabelle e indici, eseguire `INSERT` istruzioni ed eseguire altre operazioni. DDL DML

L'indirizzo IP fisico a cui punta l'endpoint del cluster cambia quando il meccanismo di failover promuove una nuova istanza DB come read/write primary instance for the cluster. If you use any form of connection pooling or other multiplexing, be prepared to flush or reduce the time-to-live for any cached DNS information. Doing so ensures that you don't try to establish a read/write connessione a un'istanza DB che non è più disponibile o che ora è di sola lettura dopo un failover.

Endpoint Reader per Amazon Aurora

Un endpoint di lettura per un cluster Aurora DB fornisce supporto per il bilanciamento delle connessioni per connessioni di sola lettura al cluster DB. Puoi utilizzare l'endpoint di lettura per le operazioni di lettura, come ad esempio le query. Elaborando tali istruzioni nelle repliche Aurora di sola lettura, questo endpoint riduce il sovraccarico sull'istanza primaria. Consente inoltre al cluster di ridimensionare la capacità di gestire query `SELECT` simultanee, proporzionale al numero di repliche Aurora nel cluster. Ogni cluster database Aurora ha un endpoint di lettura.

Se il cluster contiene una o più repliche Aurora, l'endpoint di lettura bilancia ogni richiesta di connessione tra le repliche Aurora. In tal caso, è possibile eseguire solo istruzioni di sola lettura come `SELECT` in quella sessione. Se il cluster contiene solo un'istanza primaria e nessuna replica Aurora, l'endpoint di lettura si connette all'istanza primaria. In tal caso, è possibile eseguire operazioni di scrittura attraverso l'endpoint.

L'esempio seguente mostra un endpoint di lettura per un cluster database Aurora MySQL.

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

Usi l'endpoint di lettura fornisce per le connessioni di sola lettura al cluster Aurora. Questo endpoint utilizza un meccanismo di bilanciamento delle connessioni per aiutare il cluster a gestire un carico di lavoro che richiede molte query. L'endpoint di lettura è quello che fornisci alle applicazioni che eseguono report o altre operazioni di sola lettura sul cluster.

L'endpoint di lettura bilancia le connessioni alle repliche Aurora disponibili in un cluster Aurora DB. Non bilancia le singole interrogazioni. Se desideri bilanciare ogni query per distribuire il carico di lavoro di lettura per un cluster DB, apri una nuova connessione all'endpoint di lettura per ogni query.

Ogni cluster Aurora ha un singolo endpoint di lettura integrato, il cui nome e altri attributi sono gestiti da Aurora. Non puoi creare, eliminare o modificare questo tipo di endpoint.

Se il cluster contiene solo una destinazione primaria (istanza o gruppo di shard DB) e nessuna replica Aurora, l'endpoint di lettura si connette all'istanza primaria. In tal caso, è possibile eseguire operazioni di scrittura tramite questo endpoint.

> **Tip**
>
> Tramite RDS Proxy, è possibile creare ulteriori endpoint di sola lettura per un cluster Aurora. Questi endpoint eseguono lo stesso tipo di bilanciamento delle connessioni dell'endpoint Aurora reader. Le applicazioni possono riconnettersi più rapidamente agli endpoint proxy

rispetto all'endpoint di lettura Aurora se le istanze del lettore non sono disponibili. Gli endpoint proxy possono anche sfruttare altre funzionalità proxy come il multiplexing. Per ulteriori informazioni, consulta Utilizzo degli endpoint di lettura con cluster Aurora.

Endpoint di istanza per Amazon Aurora

Un endpoint dell'istanza si connette a un'istanza database specifica all'interno di un cluster Aurora. Ogni istanza database in un cluster database dispone del proprio endpoint dell'istanza univoco. Pertanto esiste un endpoint dell'istanza per l'istanza database primaria corrente del cluster database e un endpoint dell'istanza per ciascuna replica di Aurora nel cluster database.

L'endpoint dell'istanza fornisce controllo diretto sulle connessioni al cluster database, per scenari in cui l'utilizzo dell'endpoint del cluster o dell'endpoint di lettura potrebbe non essere appropriato. Ad esempio, l'applicazione client potrebbe richiedere un bilanciamento più preciso delle connessioni in base al tipo di carico di lavoro. In questo caso, è possibile configurare più client per connettersi alle repliche di Aurora in un cluster database per distribuire i carichi di lavoro in lettura. Per un esempio che utilizza gli endpoint delle istanze per migliorare la velocità di connessione dopo un failover per Aurora Postgre, vedi. SQL Failover rapido con Amazon Aurora PostgreSQL Per un esempio che utilizza gli endpoint delle istanze per migliorare la velocità di connessione dopo un failover per Aurora My, SQL consulta MariaDB Connector/J failover support - case Amazon Aurora.

L'esempio seguente illustra un endpoint di istanza per un'istanza DB in un cluster Aurora My DB. SQL

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

Ogni istanza database in un cluster Aurora ha un proprio endpoint di istanza integrato, il cui nome e altri attributi sono gestiti da Aurora. Non puoi creare, eliminare o modificare questo tipo di endpoint. Potresti avere familiarità con gli endpoint delle istanze se utilizzi AmazonRDS. Tuttavia, in genere con Aurora si utilizzano gli endpoint di scrittura e lettura più spesso degli endpoint di istanza.

Nelle operazioni di day-to-day Aurora, il modo principale in cui si utilizzano gli endpoint delle istanze è diagnosticare problemi di capacità o prestazioni che riguardano un'istanza specifica in un cluster Aurora. Durante la connessione a un'istanza specifica, puoi esaminare le variabili di stato, i parametri e così via. Ciò può aiutarti a determinare cosa sta succedendo di diverso per quell'istanza da ciò che accade per le altre istanze nel cluster.

Nei casi di utilizzo avanzati, puoi configurare alcune istanze database in modo diverso rispetto ad altre. In questo caso, utilizza l'endpoint dell'istanza per connetterti direttamente a un'istanza più

piccola, più grande o con caratteristiche diverse rispetto alle altre. Inoltre, imposta la priorità di failover in modo che questa istanza database speciale sia l'ultima scelta da assumere come istanza primaria. In tali casi, consigliamo di utilizzare gli endpoint personalizzati anziché l'endpoint di istanza. Ciò semplifica la gestione delle connessioni e l'elevata disponibilità man mano che aggiungi altre istanze database al cluster.

Endpoint personalizzati per Amazon Aurora

L'endpoint personalizzato per un cluster Aurora rappresenta un set di istanze database selezionate. Quando ti connetti all'endpoint, Aurora esegue il bilanciamento della connessione e sceglie una delle istanze del gruppo per gestire la connessione. Puoi definire a quali istanze si riferisce questo endpoint e a quale scopo serve l'endpoint.

Un cluster database Aurora non ha endpoint personalizzati finché non ne viene creato uno. È possibile creare fino a cinque endpoint personalizzati per ogni cluster Aurora fornito oppure Aurora Serverless v2 cluster. Non è possibile utilizzare endpoint personalizzati per Aurora Serverless v1 i cluster.

L'endpoint personalizzato fornisce connessioni bilanciate al database basate su criteri diversi dalla capacità di sola lettura o lettura/scrittura delle istanze DB. Ad esempio, è possibile definire un endpoint personalizzato per connettersi a istanze che utilizzano una particolare classe di istanze o un particolare AWS gruppo di parametri DB. Poi potresti indicare a particolari gruppi di utenti questo endpoint personalizzato. Ad esempio, potresti indirizzare gli utenti interni alle istanze a bassa capacità per la generazione di report o l'esecuzione di query ad hoc (una tantum) e indirizzare il traffico di produzione alle istanze ad alta capacità.

Dal momento che è possibile eseguire la connessione a qualsiasi istanza database associata all'endpoint personalizzato, è consigliabile assicurarsi che tutte le istanze database all'interno del gruppo condividano alcune caratteristiche simili. Ciò garantisce che le prestazioni, la capacità di memoria e altre funzionalità siano coerenti per tutti coloro che si connettono a quell'endpoint.

Questa caratteristica è progettata per gli utenti esperti con tipi specializzati di carichi di lavoro in cui non è pratico mantenere identiche tutte le repliche di Aurora nel cluster. Con gli endpoint personalizzati, puoi prevedere la capacità dell'istanza database utilizzata per ciascuna connessione. Quando usi gli endpoint personalizzati, in genere non utilizzi l'endpoint di lettura per il cluster.

L'esempio seguente illustra un endpoint personalizzato per un'istanza DB in un cluster Aurora My DB. SQL

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

Gli endpoint personalizzati vengono utilizzati per semplificare la gestione delle connessioni quando il cluster contiene istanze database con capacità e impostazioni di configurazione diverse.

In precedenza, potreste aver utilizzato il CNAMES meccanismo per configurare gli alias di Domain Name Service (DNS) dal vostro dominio per ottenere risultati simili. Utilizzando endpoint personalizzati, è possibile evitare di aggiornare CNAME i record quando il cluster cresce o si riduce. Gli endpoint personalizzati significano anche che è possibile utilizzare connessioni crittografate (Transport Layer). Security/Secure Sockets Layer (TLS/SSL

Invece di utilizzare un'istanza database per ogni scopo specifico e connetterti al relativo endpoint dell'istanza, puoi creare più gruppi di istanze database specializzate. In questo caso, ciascun gruppo ha il proprio endpoint personalizzato. In questo modo, Aurora può eseguire il bilanciamento delle connessioni tra tutte le istanze dedicate ad attività come la reportistica o la gestione della produzione o le interrogazioni interne. Gli endpoint personalizzati distribuiscono le connessioni tra le istanze in modo passivo, restituendo l'indirizzo IP DNS di una delle istanze in modo casuale. Se una delle istanze database all'interno di un gruppo diventa non disponibile, Aurora indirizza le successive connessioni dell'endpoint personalizzato a una delle altre istanze database associate allo stesso endpoint.

Argomenti

- Considerazioni sugli endpoint personalizzati in Amazon Aurora
- Creazione di un endpoint personalizzato
- Visualizzazione degli endpoint personalizzati
- Modifica di un endpoint personalizzato
- Eliminazione di un endpoint personalizzato
- AWS CLI esempi di endpoint personalizzati per Amazon Aurora

Considerazioni sugli endpoint personalizzati in Amazon Aurora

Utilizza le seguenti sezioni per gestire, specificare le proprietà e utilizzare le regole di appartenenza per gli endpoint personalizzati.

Argomenti

- Gestione degli endpoint personalizzati

- Configurazione delle proprietà degli endpoint personalizzati
- Regole di appartenenza degli endpoint personalizzati

Gestione degli endpoint personalizzati

Dal momento che i cluster Aurora appena creati non hanno endpoint personalizzati, devi creare e gestire questi oggetti autonomamente. Puoi farlo utilizzando AWS Management Console AWS CLI, o Amazon RDSAPI.

> **Note**
>
> Inoltre, devi creare e gestire gli endpoint personalizzati per i cluster Aurora ripristinati da snapshot. Gli endpoint personalizzati non sono inclusi nello snapshot. Li crei nuovamente dopo il ripristino e scegli nuovi nomi per gli endpoint se il cluster ripristinato si trova nella stessa regione di quello originale.

Per lavorare con endpoint personalizzati da AWS Management Console, accedi alla pagina dei dettagli del tuo cluster Aurora e usa i controlli nella sezione Endpoint personalizzati.

Per lavorare con endpoint personalizzati da AWS CLI, puoi utilizzare queste operazioni:

- create-db-cluster-endpoint
- describe-db-cluster-endpoints
- modify-db-cluster-endpoint
- delete-db-cluster-endpoint

Per lavorare con endpoint personalizzati tramite Amazon RDSAPI, puoi utilizzare le seguenti funzioni:

- Endpoint C reateDBCluster
- D Punti finali escribeDBCluster
- Punto finale M odifyDBCluster
- Punto finale D eleteDBCluster

Configurazione delle proprietà degli endpoint personalizzati

La lunghezza massima di un nome di endpoint personalizzato è 63 caratteri. Il formato del nome è il seguente:

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

Non è possibile riutilizzare lo stesso nome di endpoint personalizzato per più di un cluster nella stessa Regione AWS. L'DNSidentificativo del cliente è un identificativo univoco associato al tuo Account AWS in un particolare. Regione AWS

A ogni endpoint personalizzato è associato un tipo che determina quali istanze database possono essere associate all'endpoint. Attualmente, il tipo può essere `READER` o. `ANY` Le seguenti considerazioni si applicano ai tipi di endpoint personalizzato:

- Non puoi selezionare il tipo di endpoint personalizzato nella AWS Management Console. Tutti gli endpoint personalizzati che crei tramite AWS Management Console hanno un tipo di`ANY`.

  Puoi impostare e modificare il tipo di endpoint personalizzato utilizzando AWS CLI o Amazon RDSAPI.
- Solo le istanze database di lettura possono far parte di un endpoint personalizzato `READER`.
- Le istanze database di lettura e di scrittura possono entrambe fare parte di un endpoint personalizzato `ANY`. Aurora dirige le connessioni agli endpoint cluster con tipo `ANY` a qualsiasi istanza database associata con uguale probabilità. Il tipo `ANY` si applica ai cluster che utilizzano qualsiasi topologia di replica.
- Se tenti di creare un endpoint personalizzato con un tipo non appropriato in base alla configurazione di replica per un cluster, Aurora restituisce un errore.

Regole di appartenenza degli endpoint personalizzati

Quando aggiungi un'istanza database a un endpoint personalizzato o la si rimuove da un endpoint personalizzato, tutte le connessioni esistenti all'istanza database rimangono attive.

È possibile definire un elenco di istanze database da includere o escludere da un endpoint personalizzato. Ci riferiamo a questi elenchi come elenchi statici e di esclusioni, rispettivamente. Puoi utilizzare il meccanismo di inclusione/esclusione per suddividere ulteriormente i gruppi di istanze database e assicurarti che il set di endpoint personalizzati copra tutte le istanze database del cluster. Ogni endpoint personalizzato può contenere solo uno di questi tipi di elenchi.

Nel: AWS Management Console

- La scelta è rappresentata dalla casella di controllo Attach future instances added to this cluster (Collega le istanze aggiunte in futuro a questo cluster). Quando lasci deselezionata la casella di controllo, l'endpoint personalizzato utilizza un elenco statico contenente solo le istanze database specificate nella pagina. Quando selezioni la casella di controllo, l'endpoint personalizzato utilizza un elenco di esclusioni. In questo caso, l'endpoint personalizzato rappresenta tutte le istanze database nel cluster (incluse quelle che aggiungi in seguito) tranne quelle lasciate deselezionate nella pagina.
- La console non consente di specificare il tipo di endpoint. Qualsiasi endpoint personalizzato creato utilizzando la console è di tipo `ANY`.

  Pertanto, Aurora non modifica l'appartenenza all'endpoint personalizzato quando le istanze database cambiano ruolo tra istanza di scrittura e istanza di lettura a causa di un failover o di una promozione.

In AWS CLI e Amazon RDSAPI:

- Puoi specificare il tipo di endpoint. Pertanto, quando il tipo di endpoint è impostato su`READER`, l'appartenenza all'endpoint viene regolata automaticamente durante i failover e le promozioni.

  Ad esempio, un endpoint personalizzato con tipo `READER` include una replica Aurora che viene quindi promossa a istanza Writer DB. La nuova istanza di scrittura non fa più parte dell'endpoint personalizzato.
- È possibile aggiungere singoli membri e rimuoverli dagli elenchi dopo che hanno cambiato ruolo. Utilizzate il modify-db-cluster-endpoint AWS CLI comando o l'operazione M Endpoint. odifyDBCluster API

Puoi associare un'istanza database a più di un endpoint personalizzato. Ad esempio, supponiamo che aggiungi una nuova istanza database a un cluster o che Aurora aggiunga automaticamente un'istanza database tramite il meccanismo di Auto Scaling. In questi casi, l'istanza database viene aggiunta a tutti gli endpoint personalizzati per i quali è idonea. Gli endpoint a cui viene aggiunta l'istanza DB si basano sul tipo di endpoint personalizzato di `READER` or`ANY`, più eventuali elenchi statici o di esclusione definiti per ciascun endpoint. Ad esempio, se l'endpoint include un elenco statico di istanze database, le repliche di Aurora aggiunte non vengono inserite nell'endpoint. Al contrario, se l'endpoint ha un elenco di esclusioni, le repliche di Aurora aggiunte vengono inserite

nell'endpoint, se non sono citate nell'elenco di esclusioni e i ruoli corrispondono al tipo dell'endpoint personalizzato.

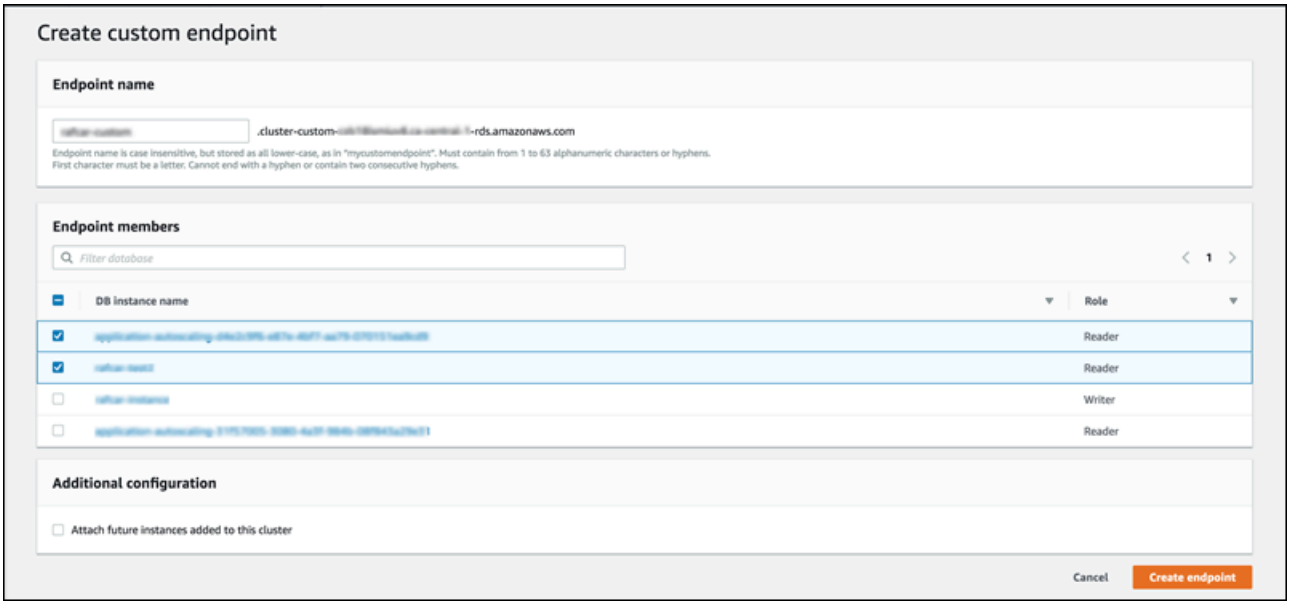
Se una replica di Aurora diventa non disponibile, rimane associata agli endpoint personalizzati. Ad esempio, continua a far parte dell'endpoint personalizzato quando non è integra, è stata interrotta, riavviata e così via. Tuttavia, non puoi connetterti alla replica tramite questi endpoint finché non diventa nuovamente disponibile.

Creazione di un endpoint personalizzato

Crea un endpoint personalizzato utilizzando AWS Management Console AWS CLI, o l'API Amazon RDS.

Console

Per creare un endpoint personalizzato con AWS Management Console, vai alla pagina dei dettagli del cluster e scegli l'**`Create custom endpoint`**azione nella sezione Endpoints. Scegli un nome per l'endpoint personalizzato, univoco per l'ID utente e la regione. Per selezionare un elenco di istanze database che rimane invariato anche quando il cluster si espande, lascia deselezionata la casella di controllo Attach future instances added to this cluster (Collega le istanze che in futuro saranno aggiunte a questo cluster). Quando selezioni questa casella di controllo, l'endpoint personalizzato aggiunge dinamicamente nuove istanze man mano che le aggiungi al cluster.



Non puoi selezionare il tipo di endpoint personalizzato nella AWS Management Console. Tutti gli endpoint personalizzati che crei tramite hanno un tipo di. AWS Management Console ANY

AWS CLI

Per creare un endpoint personalizzato con AWS CLI, esegui il comando. create-db-cluster-endpoint

Il seguente comando crea un endpoint personalizzato collegato a un cluster specifico. Inizialmente, l'endpoint è associato a tutte le istanze di replica di Aurora nel cluster. Un comando successivo lo associa a un set specifico di istanze database del cluster.

Per LinuxmacOS, oUnix:

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

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

Per Windows:

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

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

API RDS

Per creare un endpoint personalizzato con l'API RDS, esegui l'operazione Create DBCluster Endpoint.

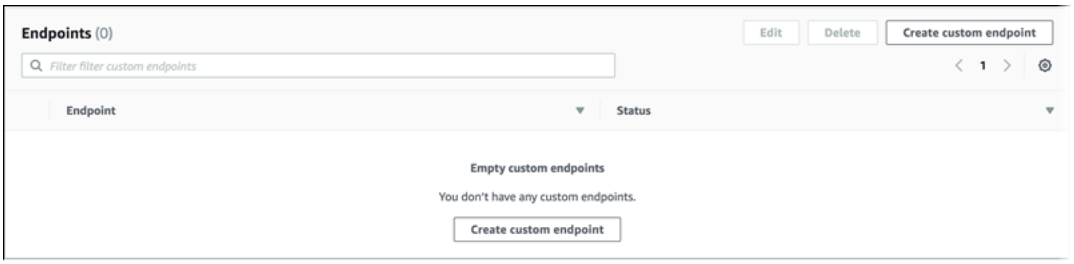
Visualizzazione degli endpoint personalizzati

Console

Per visualizzare gli endpoint personalizzati con AWS Management Console, vai alla pagina dei dettagli del cluster relativa al cluster e consulta la sezione Endpoints. Questa sezione contiene solo

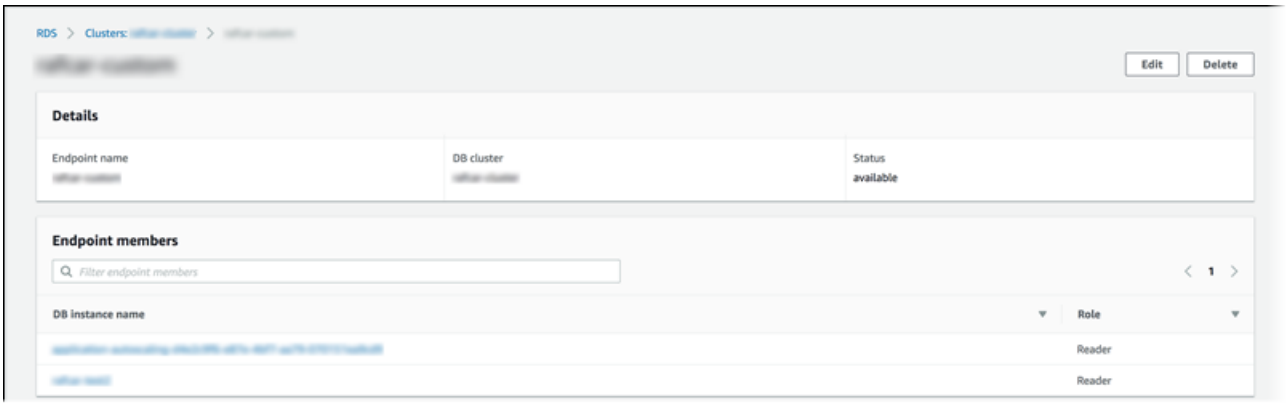
informazioni relative agli endpoint personalizzati. I dettagli per gli endpoint integrati sono elencati nella sezione Dettagli principale. Per vedere i dettagli di uno specifico endpoint personalizzato, selezionane il nome per visualizzare la relativa pagina dei dettagli.

La seguente schermata mostra l'elenco degli endpoint personalizzati per un cluster Aurora che è inizialmente vuoto.



Una volta creati gli endpoint personalizzati per il cluster, vengono visualizzati nella sezione Endpoint.

Facendo clic sulla pagina dei dettagli vengono mostrate le istanze database a cui è attualmente associato l'endpoint.



Per visualizzare altri dettagli sulle nuove istanze database aggiunte al cluster che vengono aggiunte automaticamente anche all'endpoint, apri la pagina Edit (Modifica) per l'endpoint.

AWS CLI

Per visualizzare gli endpoint personalizzati con AWS CLI, esegui il comando. describe-db-cluster-endpoints

Il seguente comando mostra gli endpoint personalizzati associati a un cluster specificato in una regione specifica. L'output include gli endpoint integrati e tutti gli endpoint personalizzati.

In Linux, macOS, oppure Unix:

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

In Windows:

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

Di seguito è illustrato un esempio di output di un comando `describe-db-cluster-endpoints`. `EndpointType` di `WRITER` o `READER` indica gli endpoint integrati di lettura-scrittura e di sola lettura per il cluster. L'`EndpointType` `CUSTOM` indica gli endpoint creati e scegli le istanze database associate. Il campo `StaticMembers` di uno degli endpoint non è vuoto, a indicare che è associato a un set preciso di istanze database. Il campo `ExcludedMembers` dell'altro endpoint non è vuoto, a indicare che l'endpoint è associato a tutte le istanze database che non sono elencate in `ExcludedMembers`. Questo secondo tipo di endpoint personalizzato aumenta per includere le nuove istanze man mano che le aggiungi al cluster.

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

```
   "ExcludedMembers": [],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "powers-of-2.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-08",
     "custom-endpoint-demo-01",
     "custom-endpoint-demo-02"
   ],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:powers-of-2"
  },
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "eight-and-higher",
   "ExcludedMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-02",
     "custom-endpoint-demo-07",
     "custom-endpoint-demo-05",
     "custom-endpoint-demo-03",
     "custom-endpoint-demo-06",
     "custom-endpoint-demo-01"
   ],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "eight-and-higher.cluster-custom-123456789012.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHYQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:eight-and-higher"
  }
 ]
}
```

RDS API

Per visualizzare gli endpoint personalizzati con RDSAPI, eseguire l'operazione D escribeDBCluster Endpoints.html.

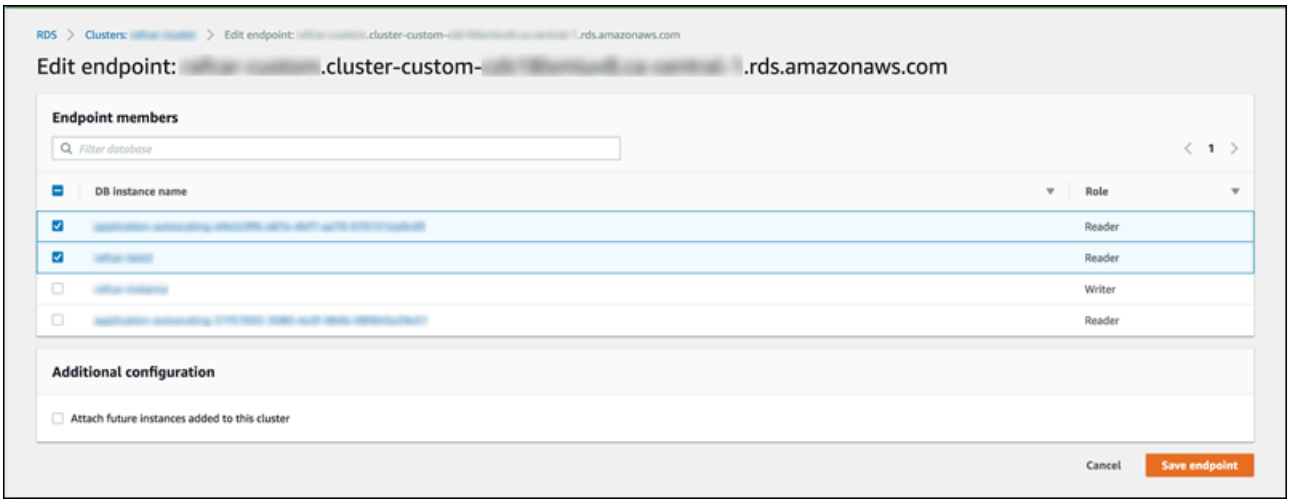
Modifica di un endpoint personalizzato

Puoi modificare le proprietà di un endpoint personalizzato per cambiare le istanze database associate all'endpoint. Puoi anche modificare un endpoint tra un elenco statico e un elenco di esclusioni. Per maggiori dettagli su queste proprietà dell'endpoint, vedi Regole di appartenenza degli endpoint personalizzati.

Puoi continuare a connetterti e utilizzare un endpoint personalizzato mentre sono in corso le modifiche di un'operazione di modifica.

Console

Per modificare un endpoint personalizzato con AWS Management Console, puoi selezionare l'endpoint nella pagina di dettaglio del cluster oppure visualizzare la pagina di dettaglio dell'endpoint e scegliere l'azione Modifica.



AWS CLI

Per modificare un endpoint personalizzato con AWS CLI, esegui il comando. modify-db-cluster-endpoint

I seguenti comandi modificano il set di istanze database che si applicano a un endpoint personalizzato e opzionalmente alterna il comportamento di elenco statico e di esclusione. I parametri

`--static-members` e `--excluded-members` usano un elenco separato da spazi di identificatori istanze database.

In Linux, macOS, oppure Unix:

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

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

In Windows:

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

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

RDS API

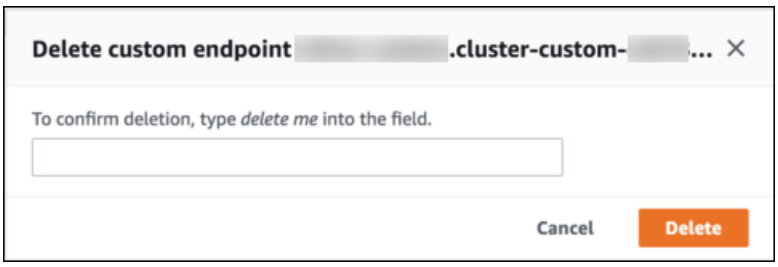
Per modificare un endpoint personalizzato con RDSAPI, eseguire l'operazione M odifyDBCluster Endpoint.html.

Eliminazione di un endpoint personalizzato

Elimina un endpoint personalizzato utilizzando AWS Management Console AWS CLI, o Amazon RDSAPI.

Console

Per eliminare un endpoint personalizzato con AWS Management Console, vai alla pagina dei dettagli del cluster, seleziona l'endpoint personalizzato appropriato e seleziona l'azione Elimina.



AWS CLI

Per eliminare un endpoint personalizzato con AWS CLI, esegui il comando. delete-db-cluster-endpoint

Il seguente comando elimina un endpoint personalizzato. Non è necessario specificare il cluster associato, ma devi specificare la regione.

In Linux, macOS, oppure Unix:

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

In Windows:

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

RDS API

Per eliminare un endpoint personalizzato con RDSAPI, esegui l'operazione D eleteDBCluster Endpoint.

AWS CLI esempi di endpoint personalizzati per Amazon Aurora

Il seguente tutorial utilizza AWS CLI esempi con la sintassi della shell Unix per mostrare come definire un cluster con diverse istanze DB «piccole» e alcune istanze DB «grandi» e creare endpoint personalizzati per connettersi a ciascun set di istanze DB. Per eseguire comandi simili sul proprio sistema, è necessario avere sufficiente dimestichezza con le nozioni di base sull'utilizzo dei cluster Aurora AWS CLI e sull'utilizzo per fornire valori propri per parametri quali regione, gruppo di sottorete e gruppo di sicurezza. VPC

Questo esempio dimostra le fasi della configurazione iniziale: la creazione di un cluster Aurora e l'aggiunta delle istanze database. Questo è un cluster eterogeneo, il che significa che non tutte le istanze database hanno la stessa capacità. La maggior parte delle istanze utilizza la classe di AWS istanzadb.r4.4xlarge, ma le ultime due istanze DB la utilizzano. db.r4.16xlarge Ciascuno di questi create-db-instance comandi di esempio stampa il proprio output sullo schermo e ne salva una copia JSON in un file per un'ispezione successiva.

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

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

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

Le istanze più grandi sono riservate a tipi specializzati di query per la creazione di report. Per rendere improbabile che vengano promosse all'istanza primaria, l'esempio seguente imposta il livello di promozione sulla priorità più bassa. In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

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

Supponiamo di voler utilizzare le due istanze "più grandi" solo per le query che richiedono più risorse. A tale scopo, puoi innanzitutto creare un endpoint di sola lettura personalizzato. Quindi puoi aggiungere un elenco statico di membri in modo che l'endpoint si connetta solo a tali istanze database. Queste istanze sono già nel livello di promozione più basso, rendendo improbabile che una di esse venga promossa a istanza primaria. Se una di esse venisse promossa a istanza primaria, diventerebbe irraggiungibile tramite questo endpoint perché abbiamo specificato il tipo `READER` anziché `ANY`.

L'esempio seguente illustra i comandi di creazione e modifica dell'endpoint e un JSON output di esempio che mostra lo stato iniziale e modificato dell'endpoint personalizzato.

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

$ aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier big-instances \
    --static-members custom-endpoint-demo-09 custom-endpoint-demo-10 --region $REGION
{
    "EndpointType": "CUSTOM",
    "ExcludedMembers": [],
    "DBClusterEndpointIdentifier": "big-instances",
```

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

L'endpoint READER predefinito per il cluster può connettersi alle istanze database piccole o grandi, rendendo poco pratico prevedere le prestazioni e la scalabilità delle query quando il cluster diventa occupato. Per dividere in modo pulito il carico di lavoro tra i gruppi di istanze database, puoi ignorare l'endpoint READER predefinito e creare un secondo endpoint personalizzato che si connette a tutte le altre istanze database. L'esempio seguente effettua ciò creando un endpoint personalizzato e quindi aggiungendo un elenco di esclusioni. Qualsiasi altra istanza database aggiunta al cluster successivamente verrà inserita automaticamente in questo endpoint. Il tipo ANY indica che questo endpoint è associato a otto istanze in totale: l'istanza primaria e altre sette repliche di Aurora. Se l'esempio avesse specificato il tipo READER, l'endpoint personalizzato sarebbe associato solo alle sette repliche di Aurora.

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

```
    "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY"
}

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

L'esempio seguente controlla lo stato degli endpoint per questo cluster. Il cluster ha ancora l'endpoint cluster originale, con `EndPointType` of`WRITER`, che è ancora utilizzabile per l'amministrazione e altre operazioni di ETL scrittura. Ha ancora il suo endpoint `READER` originale, che non usi perché ogni connessione verrebbe diretta a una istanza database piccola o grande. Gli endpoint personalizzati rendono questo comportamento prevedibile, con connessioni garantite per l'utilizzo di una delle istanze database piccole o grandi in base all'endpoint specificato.

```
$ aws rds describe-db-cluster-endpoints --region $REGION
{
    "DBClusterEndpoints": [
        {
            "EndpointType": "WRITER",
            "Endpoint": "custom-endpoint-demo.cluster-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "Status": "available",
            "DBClusterIdentifier": "custom-endpoint-demo"
        },
```

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

```
}
```

Gli esempi finali dimostrano come le successive connessioni del database agli endpoint personalizzati avvengono alle varie istanze database nel cluster Aurora. L'endpoint `small-instances` si connette sempre alle istanze database `db.r4.4xlarge` che sono gli host con un numero inferiore in questo cluster.

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

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

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

L'endpoint `big-instances` si connette sempre alle istanze database `db.r4.16xlarge` che sono i due host con il numero più alto in questo cluster.

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

```
+--------------------------+
| custom-endpoint-demo-10 |
+--------------------------+

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

Classi di istanze DB Amazon Aurora

La classe di istanza database determina la capacità di calcolo e memoria di un'istanza database Amazon Aurora. La classe di istanza database di cui hai bisogno dipende dalla potenza di elaborazione e dai requisiti di memoria specifici.

Una classe di istanza database è costituita dal tipo di classe di istanza database e dalla dimensione. Ad esempio, db.r6g è un tipo di classe di istanza DB ottimizzato per la memoria alimentato da processori Graviton2. AWS Nel tipo di classe di istanza db.r6g, db.r6g.2xlarge è una classe di istanza database. La dimensione di questa classe è 2xlarge.

Per ulteriori informazioni sui prezzi delle classi di istanza, consulta Prezzi di Amazon RDS.

Per ulteriori informazioni sui tipi di classi di istanze DB, sui motori DB supportati, sui supporti, sulla sezioni seguenti.

Argomenti

- Tipi di classi di istanza database
- Motori DB supportati per classi di istanza database
- Determinazione del supporto della classe di istanze DB in Regioni AWS
- Specifiche hardware per le classi di istanza database per Aurora

Tipi di classi di istanza database

Amazon Aurora supporta le classi di istanza database per i seguenti casi d'uso:

- Aurora Serverless v2
- Ottimizzato per la memoria
- Istanze a prestazioni espandibili
- Letture ottimizzate

Per ulteriori informazioni sui tipi di EC2 istanze Amazon, consulta Tipi di istanza nella EC2 documentazione di Amazon.

Aurora Serverless v2 tipo di classe di istanza

I seguenti Aurora Serverless v2 il tipo è disponibile:

- db.serverless — Un tipo speciale di classe di istanza DB utilizzato da Aurora Serverless v2. Aurora regola le risorse di calcolo, memoria e rete in modo dinamico in base al cambiamento del carico di lavoro. Per informazioni dettagliate sull'utilizzo, consulta Utilizzo Aurora Serverless v2.

Tipi di classe di istanza ottimizzata per la memoria

La famiglia X ottimizzata per la memoria supporta le seguenti classi di istanza:

- db.x2g — Classi di istanze ottimizzate per applicazioni a uso intensivo di memoria e alimentate da processori Graviton2. AWS Queste classi di istanza offrono un basso costo per GiB di memoria.

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton2. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.

La famiglia R ottimizzata per la memoria supporta i seguenti tipi di classi di istanza:

- db.r8g — AWS Classi di istanze basate su processori Graviton4. Queste classi di istanza sono ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria in . Queste istanze offrono istanze di dimensioni maggiori con fino a 3 volte più v CPUs e memoria rispetto alle istanze db.r7g di settima generazione basate su Graviton3. AWS

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton4. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.
- db.r7g — Classi di istanze basate su processori Graviton3. AWS Queste classi di istanza sono ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria in .

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton3. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.
- db.r7i — Classi di istanze basate su processori scalabili Intel Xeon di quarta generazione. Queste classi di istanze sono certificate SAP e sono ideali per l'esecuzione di carichi di lavoro che richiedono molta memoria MySQL e PostgreSQL. È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB basate su processori scalabili Intel Xeon di quarta generazione. Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.

- db.r6g — Classi di istanze basate su processori Graviton2. AWS Queste classi di istanza sono ideali per l'esecuzione di carichi di lavoro a uso intensivo di memoria in

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB basate sui processori Graviton2. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.
- db.r6i: classi di istanza ottimizzate basate su processori Intel Xeon scalabili di terza generazione Queste classi di istanze sono certificate SAP e sono ideali per carichi di lavoro che richiedono molta memoria MySQL e PostgreSQL.
- db.r4: classi di istanze ottimizzate per applicazioni a uso intensivo della memoria. Queste classi di istanza offrono reti migliorate e prestazioni Amazon Elastic Block Store (Amazon EBS). Sono alimentati dal sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggero.
- db.r4 — Queste classi di istanze sono supportate solo per le versioni Aurora MySQL 2.x e Aurora PostgreSQL 11 e 12. Per tutti i cluster Aurora DB che utilizzano classi di istanze DB db.r4, consigliamo di eseguire l'aggiornamento a una classe di istanze di generazione superiore il prima possibile.

  Le classi di istanze db.r4 non sono disponibili per Aurora I/O-Optimized configurazione dello storage del cluster.

Tipi di classe di istanza a prestazioni espandibili

Sono disponibili i tipi di classe di istanza database a prestazioni espandibili seguenti:

- db.t4g — Classi di istanze generiche basate su processori Graviton2 basati su ARM. AWS Queste classi di istanza offrono prestazioni di prezzo migliori rispetto alle classi di istanza database con prestazioni espandibili della generazione precedente per un ampio set di carichi di lavoro espandibili. Le istanze Amazon RDS di tipo db.t4g sono configurate per la modalità illimitata. Questo significa che possono espandersi oltre la linea di base in una finestra di 24 ore a un costo aggiuntivo.

  È possibile modificare un'istanza DB per utilizzare una delle classi di istanze DB alimentate dai processori Graviton2. AWS Per farlo, esegui gli stessi passaggi utilizzati per qualsiasi altra modifica dell'istanza database.
- db.t3: classi di istanza che forniscono un livello di prestazioni di base, con la possibilità di burst per un utilizzo completo della CPU. Le istanze di tipo db.t3 sono configurate per la modalità illimitata. Queste classi di istanza forniscono più capacità di calcolo rispetto alle classi di istanza db.t2

precedenti. Sono basate sul nuovo sistema AWS Nitro, una combinazione di hardware dedicato e hypervisor leggeri. Consigliamo di usare queste classi di istanza solo per i server di sviluppo e test o altri server non di produzione.

- db.t2: classi di istanze che forniscono un livello di prestazioni di base, con la possibilità di burst per un utilizzo completo della CPU. Le istanze db.t2 sono configurate per la modalità Unlimited. Consigliamo di usare queste classi di istanza solo per i server di sviluppo e test o altri server non di produzione.

  Le classi di istanze db.t2 non sono disponibili per Aurora I/O-Optimized configurazione dello storage del cluster.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per informazioni dettagliate sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Per le specifiche dell'hardware della classe di istanza database, consultare Specifiche hardware per le classi di istanza database per Aurora.

Tipi di classi di istanze Reads ottimizzati

I tipi di classe di istanza per letture ottimizzate disponibili sono i seguenti:

- db.r6gd — Classi di istanze alimentate da processori Graviton2. AWS Queste classi di istanze sono ideali per eseguire carichi di lavoro che richiedono molta memoria e offrono storage SSD a livello di blocco NVMe su base locale per applicazioni che richiedono storage locale ad alta velocità e bassa latenza.
- db.r6id: classi di istanza ottimizzate basate su processori Intel Xeon scalabili di terza generazione. Queste classi di istanza sono certificate SAP e ideali per l'esecuzione di carichi di lavoro con elevati requisiti di memoria. Offrono una memoria massima di 1 TiB e fino a 7,6 TB di storage SSD basato su collegamento diretto NVMe.

Motori DB supportati per classi di istanza database

Le tabelle seguenti mostrano le classi di istanze DB supportate per i motori DB di Amazon Aurora.

db.serverless — Aurora Serverless v2 classe di istanza con ridimensionamento automatico della capacità

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.serverless	Per informazioni, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v2.	Per informazioni, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v2.

db.x2g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton2 AWS

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.x2g.16xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.12xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.8xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.4xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
		successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.2xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.x2g.large	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

db.r6gd — Classi di istanze Optimized Reads basate su processori Graviton2 AWS

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6gd.16xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.12xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6gd.8xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.4xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.2xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6gd.xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive

db.r6i: classi di istanza letture ottimizzate

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6id.32xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive
db.r6id.24xlarge	No	16.1 e versioni successive, 15.4 e versioni successive, 14.9 e versioni successive

db.r8g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton4 AWS

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r8g. 48 x large	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.24xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r8g.16xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.12xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.8xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.4xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.2xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r8g.large	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

db.r7g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton3 AWS

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r7g.16xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.12xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.8xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.4xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.2xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.xlarge	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive
db.r7g.large	2.12.0 e versioni successive, 3.03.1 e versioni successive	16.1 e versioni successive, 15.2 e versioni successive, 14.7 e versioni successive, 13.10 e versioni successive

db.r7i — classi di istanze ottimizzate per la memoria

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r7i.48xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.24xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.16xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.12xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.8xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.4xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.2xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r7i.xlarge	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive
db.r7i.large	3.08.0 e versioni successive	16.3 e versioni successive, 15.7 e versioni successive, 14.12 e versioni successive, 13.15 e versioni successive

db.r6g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton2 AWS

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6g.16xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.12xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.8xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6g.4xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.2xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive
db.r6g.large	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.8 e versioni successive, 11.9, 11.12 e versioni successive

db.r6i: classi di istanze ottimizzate per la memoria

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6i.32xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
		successive, 12.9 e versioni successive
db.r6i.24xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.16xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.12xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.8xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.4xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r6i.2xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.xlarge	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive
db.r6i.large	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.5 e versioni successive, 12.9 e versioni successive

db.r5: classi di istanze ottimizzate per la memoria

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r5.24xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.16xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.12xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.8xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r5.4xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.2xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.xlarge	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili
db.r5.large	Tutte le versioni attualmente disponibili	Tutte le versioni attualmente disponibili

db.r4: classi di istanze ottimizzate per la memoria

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.r4.16xlarge	Tutte le versioni 2.x; non supportate nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.8xlarge	Tutte le versioni 2.x; non supportate nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.4xlarge	Tutte le versioni 2.x; non supportate nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.2xlarge	Tutte le versioni 2.x; non supportate nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.xlarge	Tutte le versioni 2.x; non supportate nelle versioni 3.x	Tutte le versioni 11 e 12 supportate
db.r4.large	Tutte le versioni 2.x; non supportate nelle versioni 3.x	Tutte le versioni 11 e 12 supportate

db.t4g: classi di istanze a prestazioni eccezionali basate su processori Graviton2 AWS

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.t4g.2xlarge	No	No
db.t4g.xlarge	No	No
db.t4g.large	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive
db.t4g.medium	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive
db.t4g.small	No	No

db.t3: classi di istanze a prestazioni espandibili

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.t3.2xlarge	No	No
db.t3.xlarge	No	No
db.t3.large	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.t3.medium	Tutte le versioni attualmente disponibili	16.1 e versioni successive, 15.2 e versioni successive, 14.3 e versioni successive, 13.3 e versioni successive, 12.7 e versioni successive, 11.12 e versioni successive
db.t3.small	Tutte le versioni 2.x; non supportate nelle versioni 3.x	No
db.t3.micro	No	No

db.t2: classi di istanze a prestazioni espandibili

Classe di istanza	Aurora Mia SQL	Aurora Postgre SQL
db.t2.medium	Tutte le versioni 2.x; non supportate nelle versioni 3.x	No
db.t2.small	Tutte le versioni 2.x; non supportate nelle versioni 3.x	No

Determinazione del supporto della classe di istanze DB in Regioni AWS

Per determinare le classi di istanza database supportate da ciascun motore di database in una specifica Regione AWS, puoi adottare uno di diversi approcci. Puoi utilizzare la AWS Management Console pagina RDSdei prezzi di Amazon o il AWS CLI comando describe-orderable-db-instance-options.

> **Note**
>
> Quando esegui operazioni con AWS Management Console, mostra automaticamente le classi di istanze DB supportate per uno specifico motore DB, una versione del motore DB

e Regione AWS. Esempi di operazioni che è possibile eseguire includono la creazione e la modifica di un'istanza database.

Indice

- Utilizzo della pagina RDS dei prezzi di Amazon per determinare il supporto della classe di istanze DB in Regioni AWS
- Utilizzo di AWS CLI per determinare il supporto delle classi di istanze DB in Regioni AWS
  - Elenco delle classi di istanza database supportate da una versione specifica del motore database in una Regione AWS
  - Elenco delle versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS

Utilizzo della pagina RDS dei prezzi di Amazon per determinare il supporto della classe di istanze DB in Regioni AWS

Puoi utilizzare la pagina Prezzi di Amazon Aurora per determinare le classi di istanza database supportate da ciascun motore del database in una Regione AWS specifica.

Per utilizzare la pagina dei prezzi per determinare le classi di istanza database supportate da ciascun modulo di gestione in una regione

1. Vai a Prezzi di Amazon Aurora.
2. Scegli un motore Amazon Aurora nella sezione Calcolatore dei prezzi AWS .
3. In Scegli una regione, scegli una Regione AWS.
4. In Opzione di configurazione del cluster, scegli un'opzione di configurazione.
5. Utilizza la sezione dedicata alle istanze compatibili per visualizzare le classi di istanze database supportate.
6. (Facoltativo) Scegli altre opzioni nella calcolatrice, quindi scegli Salva e visualizza riepilogo o Salva e aggiungi servizio.

Utilizzo di AWS CLI per determinare il supporto delle classi di istanze DB in Regioni AWS

È possibile utilizzare il AWS CLI per determinare quali classi di istanze DB sono supportate per motori DB specifici e versioni del motore DB in un Regione AWS.

Per utilizzare gli AWS CLI esempi seguenti, inserisci valori validi per il motore DB, la versione del motore DB, la classe di istanza DB e Regione AWS. Nella tabella seguente vengono illustrati i valori validi del motore DB.

Nome motore	Valore del motore nei CLI comandi	Ulteriori informazioni sulle versioni
La mia SQL Aurora compatibile con 5.7 e 8.0	`aurora-mysql`	Aggiornamenti del motore di database per Amazon Aurora My SQL versione 2 e aggiornamenti del motore di database per Amazon Aurora SQL My versione 3 nelle note di rilascio per Aurora My SQL
Aurora Postgre SQL	`aurora-postgresql`	Note di rilascio per Aurora Postgre SQL

Per informazioni sui Regione AWS nomi, vedere. AWS Regioni

Gli esempi seguenti mostrano come determinare il supporto della classe di istanze DB in un Regione AWS utilizzando il AWS CLI comando describe-orderable-db-instance-options.

Argomenti

- Elenco delle classi di istanza database supportate da una versione specifica del motore database in una Regione AWS
- Elenco delle versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS

Elenco delle classi di istanza database supportate da una versione specifica del motore database in una Regione AWS

Per elencare le classi di istanze DB supportate da una versione specifica del motore DB in un Regione AWS, esegui il comando seguente.

Per LinuxmacOS, oUnix:

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

Per Windows:

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

L'output mostra anche le modalità motore supportate per ogni classe di istanza database.

Ad esempio, il comando seguente elenca le classi di istanze DB supportate per la versione 13.6 del motore DB Aurora SQL Postgre negli Stati Uniti orientali (Virginia settentrionale).

Per, o: Linux macOS Unix

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

Per Windows:

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --engine-
version 15.3 ^
```

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

Elenco delle versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS

Per elencare le versioni del motore DB che supportano una classe di istanza database specifica in una Regione AWS, emettere il comando seguente.

Per LinuxmacOS, oUnix:

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

Per Windows:

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

L'output mostra anche le modalità del motore supportate per ogni versione del motore DB.

Ad esempio, il comando seguente elenca le versioni del motore DB del motore DB Aurora Postgre che supportano la classe di istanze SQL DB db.r5.large negli Stati Uniti orientali (Virginia settentrionale).

LinuxPermacOS, oUnix:

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

```
    --region us-east-1
```

Per Windows:

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

Specifiche hardware per le classi di istanza database per Aurora

di questa sezione, puoi trovare dettagli hardware sulle classi di istanze DB Amazon RDS per Aurora.

Per informazioni sul supporto del motore del database di Aurora per ciascuna classe di istanza database, consulta Motori DB supportati per classi di istanza database.

Argomenti

- Terminologia hardware per le classi di istanze DB per Aurora
- Specifiche hardware per le classi di istanze ottimizzate per la memoria
- Specifiche hardware per le classi di istanze Burstable-Performance

Terminologia hardware per le classi di istanze DB per Aurora

La terminologia seguente viene utilizzata per descrivere le specifiche dell'hardware per le classi di istanza database:

VPCU

Il numero di unità di elaborazione centrale virtuali ()CPUs. Una CPU virtuale è un'unità di capacità che puoi usare per confrontare le classi di istanza database. Invece di acquistare o affittare un determinato processore da utilizzare per vari mesi o anni, si affitta la capacità su base oraria. L'obiettivo è quello di rendere disponibile una quantità coerente e specifica di capacità di CPU, entro i limiti dell'hardware effettivo sottostante.

ECU

La misura relativa della potenza di elaborazione intera di un' EC2 istanza Amazon. Per consentire agli sviluppatori di confrontare facilmente la capacità della CPU tra diverse classi di istanze,

abbiamo definito un'Amazon EC2 Compute Unit. La quantità di CPU allocata a una particolare istanza è espressa in termini di queste unità di EC2 calcolo. Attualmente una ECU fornisce una capacità della CPU equivalente a quella di un processore Opteron GHz 2007 Opteron 1.0-1,2 o 2007 Xeon.

Memoria (GiB)

La RAM, in gibibyte, allocata all'istanza database. Spesso c'è un rapporto costante tra memoria e vCPU. A titolo esemplificativo, prendi la classe di istanza db.r4, che ha una memoria in rapporto vCPU simile alla classe di istanza db.r5. Tuttavia, per la maggior parte dei casi d'uso la classe di istanza db.r5 fornisce prestazioni migliori e più costanti rispetto alla classe di istanza db.r4.

Quantità max Larghezza di banda EBS (Mbps)

La larghezza di banda EBS massima in megabit al secondo. Dividendo il valore per 8, puoi ottenere il throughput previsto in megabyte al secondo.

> **Note**
>
> Questa figura si riferisce alla larghezza di banda I/O per l'archiviazione locale all'interno dell'istanza database. Non si applica alla comunicazione con il volume Aurora del cluster.

Larghezza di banda di rete

La velocità di rete relativa ad altre classi di istanza database.

Per informazioni sull'utilizzo dei CloudWatch parametri di Amazon per monitorare il throughput dell'istanza Aurora DB, consulta e. Valutazione dell'utilizzo delle istanze DB per Aurora My SQL con i parametri di Amazon CloudWatch Valutazione dell'utilizzo delle istanze DB per Aurora CloudWatch PostgreSQL con metriche

Specifiche hardware per le classi di istanze ottimizzate per la memoria

Le tabelle seguenti mostrano le specifiche di calcolo, memoria, archiviazione e larghezza di banda per le classi di istanze ottimizzate per la memoria.

db.x2g — classi di istanze ottimizzate per la memoria con processori Graviton2 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.x2g.16xlarge	64	—	1.024	Solo ottimizzata per EBS	19.000	25
db.x2g.12xlarge	48	—	768	Solo ottimizzata per EBS	14.250	20
db.x2g.8xlarge	32	—	512	Solo ottimizzata per EBS	9.500	12
db.x2g.4xlarge	16	—	256	Solo ottimizzata per EBS	4.750	Fino a 10
db.x2g.2xlarge	8	—	128	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.x2g.xlarge	4	—	64	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.x2g.large	2	—	32	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10

db.r8g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton4 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r8g.48xlarge	192	—	1536	Solo ottimizzata per EBS	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r8g.24xlarge	96	—	768	Solo ottimizzata per EBS	30.000	40
db.r8g.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	30
db.r8g.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	22,5
db.r8g.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	15
db.r8g.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r8g.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r8g.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r8g.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r7i — classi di istanze ottimizzate per la memoria basate su processori scalabili Intel Xeon di quarta generazione

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r7i.48xlarge	192	—	1536	Solo ottimizzata per EBS	40.000	50
db.r7i.24xlarge	96	—	768	Solo ottimizzata per EBS	30.000	37,5
db.r7i.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	25
db.r7i.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	18,75
db.r7i.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	12,5
db.r7i.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7i.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7i.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7i.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r7g — classi di istanze ottimizzate per la memoria con processori Graviton3 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r7g.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	30
db.r7g.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	22.5
db.r7g.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	15
db.r7g.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r7g.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 15
db.r7g.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r7g.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r6id — classi di istanze ottimizzate per la memoria con processori scalabili Intel Xeon di terza generazione e storage SSD

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6id.32xlarge	128	—	1,024	SSD NVMe 4x1900	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6id.24xlarge	96	—	768	NVMe SSD 4 x 1425	30.000	37,5

db.r6gd — classi di istanze ottimizzate per la memoria con processori Graviton2 e storage SSD AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6gd.16xlarge	64	—	512	2 x NVMe 1900 SSD	19.000	25
db.r6gd.12xlarge	48	—	384	2 x SSD 1425 NVMe	13.500	20
db.r6gd.8xlarge	32	—	256	1 x SSD 1900 NVMe	9.000	12
db.r6gd.4xlarge	16	—	128	1 x SSD 950 NVMe	4.750	Fino a 10
db.r6gd.2xlarge	8	—	64	1 x SSD 474 NVMe	Fino a 4.750	Fino a 10
db.r6gd.xlarge	4	—	32	1 x 237 SSD NVMe	Fino a 4.750	Fino a 10

db.r6g — classi di istanze ottimizzate per la memoria con processori Graviton2 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6g.16xlarge	64	—	512	Solo ottimizzata per EBS	19.000	25
db.r6g.12xlarge	48	—	384	Solo ottimizzata per EBS	13.500	20
db.r6g.8xlarge	32	—	256	Solo ottimizzata per EBS	9.000	12
db.r6g.4xlarge	16	—	128	Solo ottimizzata per EBS	4.750	Fino a 10
db.r6g.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r6g.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r6g.large	2	—	16	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10

db.r6i: classi di istanza ottimizzata per la memoria con processori Intel Xeon scalabili di terza generazione

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6i.32xlarge	128	—	1,024	Solo ottimizzata per EBS	40.000	50

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r6i.24xlarge	96	—	768	Solo ottimizzata per EBS	30.000	37,5
db.r6i.16xlarge	64	—	512	Solo ottimizzata per EBS	20.000	25
db.r6i.12xlarge	48	—	384	Solo ottimizzata per EBS	15.000	18.75
db.r6i.8xlarge	32	—	256	Solo ottimizzata per EBS	10.000	12,5
db.r6i.4xlarge	16	—	128	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r6i.2xlarge	8	—	64	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r6i.xlarge	4	—	32	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5
db.r6i.large	2	—	16	Solo ottimizzata per EBS	Fino a 10.000	Fino a 12,5

db.r5: classi di istanze ottimizzate per la memoria

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r5.24xlarge	96	347	768	Solo ottimizzata per EBS	19.000	25

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r5.16xlarge	64	264	512	Solo ottimizzata per EBS	13.600	20
db.r5.12xlarge	48	173	384	Solo ottimizzata per EBS	9.500	12
db.r5.8xlarge	32	132	256	Solo ottimizzata per EBS	6.800	10
db.r5.4xlarge	16	71	128	Solo ottimizzata per EBS	4.750	Fino a 10
db.r5.2xlarge	8	38	64	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r5.xlarge	4	19	32	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10
db.r5.large	2	10	16	Solo ottimizzata per EBS	Fino a 4.750	Fino a 10

db.r4: classi di istanza ottimizzata per la memoria con processori Intel Xeon scalabili

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r4.16xlarge	64	195	488	Solo ottimizzata per EBS	14.000	25
db.r4.8xlarge	32	99	244	Solo ottimizzata per EBS	7,000	10

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.r4.4xlarge	16	53	122	Solo ottimizzata per EBS	3,500	Fino a 10
db.r4.2xlarge	8	27	61	Solo ottimizzata per EBS	1.700	Fino a 10
db.r4.xlarge	4	13,5	30,5	Solo ottimizzata per EBS	850	Fino a 10
db.r4.large	2	7	15,25	Solo ottimizzata per EBS	425	Fino a 10

Specifiche hardware per le classi di istanze Burstable-Performance

Le tabelle seguenti mostrano le specifiche di calcolo, memoria, archiviazione e larghezza di banda per le classi di istanze burstable-performance.

db.t4g — classi di istanze a prestazioni burstabili basate su processori Graviton2 AWS

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.t4g.large	2	—	8	Solo ottimizzata per EBS	Fino a 2.780	Fino a 5
db.t4g.medium	2	—	4	Solo ottimizzata per EBS	Fino a 2.085	Fino a 5

db.t3: classi di istanza a prestazioni espandibili

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.t3.large	2	Variab	8	Solo ottimizzata per EBS	Fino a 2.048	Fino a 5
db.t3.medium	2	Variab	4	Solo ottimizzata per EBS	Fino a 1.536	Fino a 5
db.t3.small	2	Variab	2	Solo ottimizzata per EBS	Fino a 1.536	Fino a 5

db.t2: classi di istanza a prestazioni espandibili

Classe di istanza	VPCU	ECU	Memoria (GiB)	Archiviazione delle istanze (GiB)	Quantità max Larghezza di banda EBS (Mbps)	Larghezza di banda della rete (Gbps)
db.t2.medium	2	Variab	4	Solo EBS	—	Moderata
db.t2.small	1	Variab	2	Solo EBS	—	Bassa

Archiviazione Amazon Aurora

Di seguito, sono riportare informazioni sul sottosistema di archiviazione Aurora. Aurora utilizza un'architettura di archiviazione distribuita e condivisa che è un fattore importante in termini di prestazioni, scalabilità e affidabilità per i cluster Aurora.

Argomenti

- Panoramica dell'archiviazione di Amazon Aurora
- Contenuto dei volumi del cluster
- Configurazioni dell'archiviazione per i cluster database Amazon Aurora

- Ridimensionamento automatico dello storage Aurora
- Come viene fatturato lo storage dei dati Aurora

Panoramica dell'archiviazione di Amazon Aurora

I dati Aurora vengono archiviati nel volume del cluster, che è un singolo volume virtuale che utilizza unità a stato solido ()SSDs. Un volume cluster è costituito da copie dei dati in tre zone di disponibilità in un'unica AWS regione. Poiché i dati vengono replicati automaticamente nelle zone di disponibilità, i dati risultano estremamente durevoli e poco soggetti ad andare perduti. La replica garantisce anche una maggiore disponibilità del database durante un failover perché le copie dei dati sono già presenti in altre zone di disponibilità e continuano a servire le richieste di dati alle istanze database del cluster database. La quantità di replica è indipendente dal numero di istanze database nel cluster.

Aurora utilizza lo storage locale separato per i file temporanei non persistenti. Sono inclusi i file utilizzati per scopi quali l'ordinamento di set di dati di grandi dimensioni durante l'elaborazione delle query e la creazione degli indici. Per ulteriori informazioni, consulta Limiti di storage temporaneo per Aurora MySQL e Limiti di storage temporaneo per Aurora PostgreSQL.

Contenuto dei volumi del cluster

Il volume del cluster Aurora contiene tutti i dati dell'utente, gli oggetti dello schema e i metadati interni come le tabelle di sistema e il log binario. Ad esempio, Aurora memorizza tutte le tabelle, gli indici, gli oggetti binari di grandi dimensioni (BLOBs), le stored procedure e così via per un cluster Aurora nel volume del cluster.

L'architettura di storage condivisa di Aurora rende i dati indipendenti dalle istanze database nel cluster. Ad esempio, puoi aggiungere rapidamente un'istanza database perché Aurora non crea una nuova copia dei dati della tabella. L'istanza database si connette al volume condiviso che contiene già tutti i dati. Puoi rimuovere un'istanza database da un cluster senza rimuovere alcun dato sottostante dal cluster. Solo quando elimini l'intero cluster Aurora rimuove i dati.

Configurazioni dell'archiviazione per i cluster database Amazon Aurora

Amazon Aurora dispone di due configurazioni dell'archiviazione per i cluster database:

- Aurora I/O-Optimized: rapporto prezzo/prestazioni e prevedibilità migliorati per applicazioni con uso intensivo di I/O. I prezzi sono calcolati sono calcolati solo in base all'utilizzo e all'archiviazione dei cluster database, senza costi aggiuntivi per le operazioni di I/O di lettura e scrittura.

Aurora I/O-Optimized è la scelta migliore quando la spesa di I/O è pari o superiore al 25% della spesa totale per il database Aurora.

Puoi scegliere Aurora I/O-Optimized quando si crea o si modifica un cluster DB con una versione del motore DB che supporta il Aurora I/O-Optimized configurazione del cluster. È possibile passare da Aurora I/O-Optimized in Aurora Standard in qualsiasi momento.

- Aurora Standard: prezzi convenienti per molte applicazioni con un utilizzo moderato di I/O. Oltre all'utilizzo e all'archiviazione dei cluster database, il prezzo è calcolato in base a una tariffa standard per 1 milione di richieste per le operazioni di I/O.

  Aurora Standard è la scelta migliore quando la spesa di I/O è inferiore al 25% della spesa totale per il database Aurora.

  È possibile passare da Aurora Standard in Aurora I/O-Optimized una volta ogni 30 giorni. Quando si passa da una all'altra Aurora Standard e Aurora I/O-Optimized opzioni di archiviazione per le istanze non-NVMe-based DB, non ci sono tempi di inattività. Tuttavia, per le istanze DB NVMe basate su istanze database, il passaggio da una all'altra Aurora I/O-Optimized e Aurora Standard le opzioni di archiviazione richiedono il riavvio del motore di database, che può causare un breve periodo di inattività.

Per informazioni Regione AWS e supporto delle versioni, vedereRegioni supportate e motori Aurora DB per configurazioni di storage in cluster.

Per informazioni sui prezzi relativi alle configurazioni dell'archiviazione per Amazon Aurora, consulta la pagina relativa ai prezzi di Amazon Aurora.

Per informazioni sulla scelta della configurazione dello storage durante la creazione di un cluster database, consulta Creazione di un cluster di database. Per informazioni sulla modifica della configurazione dello storage per un cluster database, consulta Impostazioni per Amazon Aurora.

Ridimensionamento automatico dello storage Aurora

I volumi dei cluster Aurora aumentano automaticamente quando aumenta la quantità di dati nel database. La dimensione massima per un volume cluster Aurora è di 128 tebibyte (TiB) o 64 TiB, a seconda della versione del motore DB. Per informazioni sulla dimensione massima per una versione specifica, consulta Limiti di dimensione Amazon Aurora. Questo dimensionamento automatico dello storage è combinato con un sottosistema di storage ad alte prestazioni e altamente distribuito. Ciò

rende Aurora la scelta ideale per i dati aziendali importanti, quando i tuoi obiettivi principali sono l'affidabilità e l'elevata disponibilità.

Per visualizzare lo stato del volume, consulta Visualizzazione dello stato del volume per un cluster DB Aurora MySQL o Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL . Per trovare modi per bilanciare i costi di storage rispetto ad altre priorità, Dimensionamento dello storage descrive come monitorare i parametri `AuroraVolumeBytesLeftTotal` di Amazon Aurora e come fare. `VolumeBytesUsed` CloudWatch

Quando i dati di Aurora vengono rimossi, lo spazio allocato per tali dati viene liberato. Esempi di rimozione di dati sono l'eliminazione o il troncamento di una tabella. Questa riduzione automatica dell'utilizzo dello storage consente di contenere al minimo i costi di storage.

> **Note**
>
> I limiti di archiviazione e il comportamento di ridimensionamento dinamico indicati in questa pagina si applicano alle tabelle persistenti e ad altri dati archiviati nel volume del cluster.
> Per Aurora PostgreSQL, i dati della tabella temporanea vengono archiviati nell'istanza DB locale.
> Per Aurora My SQL versione 2, i dati delle tabelle temporanee vengono archiviati per impostazione predefinita nel volume del cluster per le istanze Writer e nella memoria locale per le istanze Reader. Per ulteriori informazioni, consulta Motore di archiviazione per le tabelle temporanee su disco.
> Per Aurora My SQL versione 3, i dati della tabella temporanea vengono archiviati nell'istanza DB locale o nel volume del cluster. Per ulteriori informazioni, consulta Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.
> La dimensione massima delle tabelle temporanee che risiedono nell'archiviazione locale è limitata dalla dimensione massima dello spazio di archiviazione locale dell'istanza database. La dimensione dell'archiviazione locale dipende dalla classe di istanza utilizzata. Per ulteriori informazioni, consulta Limiti di storage temporaneo per Aurora MySQL e Limiti di storage temporaneo per Aurora PostgreSQL.

Alcune funzionalità di archiviazione, ad esempio la dimensione massima del volume del cluster e il ridimensionamento automatico quando i dati vengono rimossi, dipendono dalla versione di Aurora del cluster. Per ulteriori informazioni, consulta Dimensionamento dello storage. Puoi inoltre scoprire come evitare problemi di storage e come monitorare lo spazio di storage allocato e lo spazio libero nel cluster.

Come viene fatturato lo storage dei dati Aurora

Anche se un volume del cluster Aurora può crescere fino al limite di 128 tebibytes (TiB), il costo addebitato rimane limitato a quello dello spazio effettivamente utilizzato nel volume del cluster Aurora. Nelle versioni precedenti di Aurora, il volume del cluster poteva riutilizzare lo spazio liberato dalla rimozione dei dati, ma lo spazio di archiviazione allocato non diminuiva mai. Ora quando i dati di Aurora vengono rimossi, ad esempio eliminando una tabella o un database, lo spazio allocato complessivo diminuisce della quantità equivalente. Pertanto, puoi ridurre le spese di archiviazione eliminando tabelle, indici, database e così via che non sono più necessari.

> **Tip**
>
> Per le versioni precedenti senza la funzionalità di ridimensionamento dinamico, la reimpostazione dell'utilizzo dello storage per un cluster comportava l'esecuzione di un dump logico e il ripristino di un nuovo cluster. Tali operazioni possono richiedere molto tempo per un volume considerevole di dati. Se si verifica questa situazione, prendi in considerazione l'aggiornamento del cluster a una versione che supporta il ridimensionamento dinamico del volume.

Per informazioni su quali versioni di Aurora supportano il ridimensionamento dinamico e su come ridurre al minimo i costi di archiviazione monitorando l'utilizzo dello spazio di archiviazione del cluster, consulta Dimensionamento dello storage. Per informazioni sulla fatturazione dell'archiviazione di backup Aurora, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora. Per informazioni sui prezzi dello storage dati Aurora, consulta i prezzi di Amazon for RDS Aurora.

Affidabilità Amazon Aurora

Il design di Aurora assicura affidabilità, durevolezza e tolleranza agli errori. È possibile sfruttare varie opzioni per configurare l'architettura del cluster database Aurora in modo da ottimizzare la disponibilità, come aggiungere repliche di Aurora e posizionarle in zone di disponibilità diverse. Inoltre Aurora include una serie di caratteristiche automatiche che lo rendono una soluzione database estremamente affidabile.

Argomenti

- Riparazione automatica dello storage
- Cache delle pagine superstite

- Ripristino in seguito a riavvii non pianificati

Riparazione automatica dello storage

Dal momento che Aurora conserva varie copie dei dati in tre zone di disponibilità, il rischio di perdita dei dati a seguito di un problema con un disco è estremamente remoto. Aurora inoltre rileva automaticamente gli errori nei volumi dei dischi che formano il volume di cluster. In caso di errore in un volume del disco, Aurora ripara immediatamente il segmento. Quando Aurora ripara il segmento del disco, utilizza i dati in altri volumi che creano il volume del cluster per garantire che i dati del segmento riparato siano aggiornati. Di conseguenza, Aurora evita la perdita di dati e riduce la necessità di eseguire un point-in-time ripristino per il ripristino da un guasto del disco.

Cache delle pagine superstite

In Aurora, ogni cache delle pagine dell'istanza viene gestita con un processo separato dal database, che consente alla cache di pagina di sopravvivere in modo indipendente dal database. (La cache della pagina è anche chiamata pool di buffer InnoDB su Aurora My SQL e cache buffer su Aurora Postgre.) SQL

Nella remota eventualità di un errore del database, la cache delle pagine rimane in memoria. Questo consente di mantenere le pagine di dati "precaricate" nella cache delle pagine quando il database viene riavviato. Tale espediente consente di migliorare le prestazioni, perché elimina la necessità di query iniziali per eseguire operazioni di I/O di lettura per "precaricare" la cache delle pagine.

Per Aurora MySQL, il comportamento della cache delle pagine durante il riavvio e il failover è il seguente:

- È possibile riavviare l'istanza di writer senza riavviare le istanze del lettore.
  - Se le istanze di lettura non vengono riavviate al riavvio dell'istanza di scrittura, non perdono le cache delle pagine.
  - Se le istanze di lettura vengono riavviate al riavvio dell'istanza di scrittura, perdono le cache delle pagine.
- Quando un'istanza di lettura viene riavviata, entrambe le cache delle pagine sulle istanze di scrittura e lettura sopravvivono.
- Quando il cluster database esegue il failover, l'effetto è simile al riavvio di un'istanza di scrittura. Sulla nuova istanza di scrittura (in precedenza l'istanza di lettura) la cache delle pagine sopravvive, ma sull'istanza di lettura (in precedenza l'istanza di scrittura), la cache delle pagina non sopravvive.

Per Aurora PostgreSQL, è possibile utilizzare la gestione della cache del cluster per conservare la cache della pagina di un'istanza reader designata che diventa l'istanza writer dopo il failover. Per ulteriori informazioni, consulta Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL.

Ripristino in seguito a riavvii non pianificati

Aurora è progettato per eseguire il ripristino quasi istantaneo dopo un riavvio non pianificato per continuare a fornire i dati dell'applicazione senza il log binario. Aurora esegue il ripristino in modo asincrono su thread paralleli, in modo che il database sia aperto e immediatamente disponibile dopo un riavvio non pianificato.

Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora e Ottimizzazioni per ridurre i tempi di riavvio del database.

Di seguito sono riportate le considerazioni relative alla registrazione binaria e al ripristino da riavvio non pianificato su Aurora My: SQL

- L'attivazione dei log binari in Aurora incide direttamente sui tempi di ripristino dopo un riavvio non pianificato, perché obbliga l'istanza database a eseguire il ripristino dei log binari.
- Il tipo di log binari usati incide sulle dimensioni e sull'efficienza dell'operazione. Partendo dalla stessa quantità di attività del database, alcuni formati inseriscono nei log binari un numero maggiore di informazioni rispetto ad altri. Le seguenti impostazioni del parametro `binlog_format` modificano la quantità di dati inseriti nei log:
  - `ROW`– – Massima quantità di dati
  - `STATEMENT`– – Minima quantità di dati
  - `MIXED`– – Una quantità di dati moderata che in genere rappresenta il compromesso ottimale fra integrità dei dati e prestazioni

  La quantità di dati dei log binari influisce sui tempi di ripristino. Con l'inserimento di una grande quantità di dati nei log binari, l'istanza database deve elaborare più dati, allungando i tempi di ripristino.
- Per ridurre il sovraccarico di calcolo e migliorare i tempi di ripristino con la registrazione binaria, puoi utilizzare Enhanced binlog. Il binlog avanzato migliora i tempi di ripristino del database fino al 99%. Per ulteriori informazioni, consulta Configurazione di binlog avanzato per Aurora MySQL.
- Aurora non necessita dei log binari per replicare i dati all'interno di un cluster DB o per eseguire point-in-time restore (). PITR

- Se non hai bisogno del log binario per la replica esterna (o di uno stream esterno del log binario), ti consigliamo di impostare il parametro `binlog_format` su `OFF` per disattivare i log binari. In questo modo, puoi abbreviare i tempi di ripristino.

Per ulteriori informazioni sui log binari e la replica in Aurora, consulta Replica con Amazon Aurora. Per ulteriori informazioni sulle implicazioni dei diversi tipi di SQL replica My, consulta Vantaggi e svantaggi della replica basata su istruzioni e su righe nella documentazione My. SQL

Sicurezza di Amazon Aurora

La sicurezza di Amazon Aurora viene gestita su tre livelli:

- Per controllare chi può eseguire azioni di RDS gestione Amazon su cluster e istanze DB Aurora, usi (). AWS Identity and Access Management IAM Quando ti connetti AWS utilizzando le IAM credenziali, il tuo AWS account deve disporre IAM di politiche che concedano le autorizzazioni necessarie per eseguire le operazioni di RDS gestione di Amazon. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.

  Se utilizzi IAM per accedere alla RDS console Amazon, devi prima accedere AWS Management Console con le tue credenziali utente, quindi andare alla RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds.
- I cluster Aurora DB devono essere creati in un cloud privato virtuale (VPC) basato sul servizio Amazon. VPC Per controllare quali dispositivi e EC2 istanze Amazon possono aprire connessioni all'endpoint e alla porta dell'istanza DB per i cluster Aurora DB in unVPC, si utilizza un gruppo di sicurezza. VPC Puoi effettuare queste connessioni tra endpoint e porte utilizzando Transport Layer Security (TLS) /Secure Sockets Layer (). SSL Le regole del firewall aziendale possono inoltre determinare se i dispositivi in esecuzione nell'azienda possono aprire connessioni a un'istanza database. Per ulteriori informazioni su VPCs, consulta Amazon VPC e Aurora.
- Per autenticare gli accessi e le autorizzazioni per un cluster database Amazon Aurora puoi seguire uno degli approcci qui riportati oppure utilizzare una loro combinazione.
  - Puoi adottare lo stesso approccio di un'istanza DB autonoma di My o Postgre. SQL SQL

    Le tecniche per l'autenticazione degli accessi e delle autorizzazioni per le istanze DB autonome di My SQL o PostgreSQL, come l'uso di SQL comandi o la modifica delle tabelle dello schema del database, funzionano anche con Aurora. Per ulteriori informazioni, consulta Sicurezza con Amazon Aurora MySQL o Sicurezza con Amazon Aurora PostgreSQL.

- È possibile utilizzare l'autenticazione IAM del database.

  Con l'autenticazione del IAM database, ti autentichi nel tuo cluster Aurora DB utilizzando un utente IAM o un ruolo e un token di autenticazione. Il token di autenticazione è un valore univoco, generato tramite il processo di firma Signature Version 4. Utilizzando l'autenticazione IAM del database, puoi utilizzare le stesse credenziali per controllare l'accesso alle tue AWS risorse e ai tuoi database. Per ulteriori informazioni, consulta Autenticazione del database IAM .
- È possibile utilizzare l'autenticazione Kerberos per Aurora Postgre e Aurora My. SQL SQL

  Puoi usare Kerberos per autenticare gli utenti quando si connettono al tuo cluster Aurora Postgre e Aurora MySQL. SQLDB In questo caso, il cluster DB funziona con l'abilitazione dell'autenticazione Kerberos. AWS Directory Service for Microsoft Active Directory AWS Directory Service for Microsoft Active Directory viene anche chiamato. AWS Managed Microsoft AD Mantenere tutte le credenziali nella stessa directory consente di ridurre il tempo e l'impegno. È disponibile una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database. L'uso di una directory può inoltre migliorare il profilo di sicurezza complessivo. Per ulteriori informazioni, consulta Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL e Utilizzo dell'autenticazione Kerberos per Aurora My SQL.

Per informazioni sulla configurazione della sicurezza, vedi Sicurezza in Amazon Aurora.

Utilizzo SSL con i cluster Aurora DB

I cluster Amazon Aurora DB supportano connessioni Secure Sockets Layer (SSL) da applicazioni che utilizzano lo stesso processo e la stessa chiave pubblica delle istanze Amazon DB. RDS Per ulteriori informazioni, consulta Sicurezza con Amazon Aurora MySQL, Sicurezza con Amazon Aurora PostgreSQL o Usare TLS/SSL con Aurora Serverless v1.

Elevata disponibilità di Amazon Aurora

L'architettura Amazon Aurora prevede la separazione tra archiviazione e calcolo. Aurora include alcune funzionalità per l'alta disponibilità applicabili ai dati nel cluster database. I dati restano protetti anche se alcune o tutte le istanze database nel cluster smettono di essere disponibili. Altre funzionalità per l'alta disponibilità sono applicabili alle istanze database. Queste funzionalità assicurano la presenza di una o più istanze database per la gestione delle richieste al database provenienti dall'applicazione.

Argomenti

- Elevata disponibilità di dati Aurora
- Architetture ad alta disponibilità per istanze database Aurora
- Elevata disponibilità nelle regioni AWS con database globali Aurora
- Tolleranza ai guasti di un cluster DB Aurora
- Elevata disponibilità con Amazon RDS Proxy

Elevata disponibilità di dati Aurora

Aurora archivia le copie dei dati in un cluster di database in più zone di disponibilità in una singola Regione AWS. L'archiviazione avviene indipendentemente dal fatto che le istanze nel cluster database siano estese su più zone di disponibilità. Per ulteriori informazioni su Aurora, consulta Gestione di un cluster DB Amazon Aurora.

Quando i dati vengono scritti nell'istanza database primaria, Aurora replica in modo sincrono i dati nelle zone di disponibilità in sei nodi di storage associati al volume cluster. Questa operazione fornisce la ridondanza dei dati, elimina i blocchi I/O e riduce al minimo i picchi di latenza durante i backup di sistema. Eseguendo un'istanza database con disponibilità elevata, è possibile migliorare la disponibilità durante la manutenzione pianificata del sistema e consentire di proteggere i database da errori e interruzioni relative alle zone di disponibilità. Per ulteriori informazioni sulle zone di disponibilità, consulta Regioni e zone di disponibilità.

Architetture ad alta disponibilità per istanze database Aurora

Dopo aver creato l'istanza primaria (scrittura), è possibile creare fino a 15 repliche di Aurora in sola lettura. Le repliche Aurora sono note anche come istanze di lettore. Le repliche Aurora utilizzano la replica asincrona per supportare l'alta disponibilità senza influire sulle prestazioni dell'istanza primaria.

Durante day-to-day le operazioni, è possibile delegare parte del lavoro ad applicazioni ad alta intensità di lettura utilizzando le istanze del lettore per elaborare le query. `SELECT` Quando un problema riguarda l'istanza primaria, una di queste istanze del lettore prende il sopravvento come istanza primaria. Questo meccanismo è noto come failover. Molte funzionalità Aurora si applicano al meccanismo di failover. Ad esempio, Aurora rileva i problemi del database e attiva automaticamente il meccanismo di failover quando necessario. Aurora dispone inoltre di funzionalità che riducono i tempi di completamento del failover. In questo modo si riduce al minimo il tempo in cui il database non è disponibile per la scrittura durante un failover.

Aurora è progettata per eseguire il ripristino il più rapidamente possibile e il percorso più rapido per il ripristino è spesso il riavvio o il failover sulla stessa istanza DB. Il riavvio è più rapido e comporta un sovraccarico inferiore rispetto al failover.

Per utilizzare una stringa di connessione che rimane invariata anche quando un failover promuove una nuova istanza primaria, è necessario connettersi all'endpoint del cluster. L'endpoint del cluster rappresenta sempre l'istanza primaria corrente nel cluster. Per ulteriori informazioni sull'endpoint del cluster, consulta Connessioni endpoint Amazon Aurora.

> **Tip**
>
> All'interno di ciascuna di esse Regione AWS, le zone di disponibilità (AZs) rappresentano ubicazioni distinte l'una dall'altra per garantire l'isolamento in caso di interruzioni. Si consiglia di distribuire l'istanza primaria e le istanze di lettura nel cluster di database su più istanze AZs per migliorare la disponibilità del cluster di database. In questo modo, un problema che interessa un'intera AZ non causa un'interruzione del cluster.
> È possibile configurare un cluster DB Multi-AZ effettuando una scelta semplice al momento della creazione del cluster. Puoi usare l' AWS Management Console AWS CLI, the o l'API Amazon RDS. È inoltre possibile convertire un cluster Aurora DB esistente in un cluster DB Multi-AZ aggiungendo una nuova istanza DB reader e specificando una AZ diversa.

Elevata disponibilità nelle regioni AWS con database globali Aurora

Per un'elevata disponibilità su più piattaforme Regioni AWS, puoi configurare i database globali Aurora. Ogni database globale Aurora si estende su più database Regioni AWS, consentendo letture globali a bassa latenza e disaster recovery in caso di interruzioni su un. Regione AWS Aurora replica in modo asincrono tutti i dati e gli aggiornamenti dalla regione primaria Regione AWS a ciascuna delle regioni secondarie. Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.

Tolleranza ai guasti di un cluster DB Aurora

In base alla progettazione, un cluster DB Aurora è tollerante ai guasti. Il volume del cluster si estende su più zone di disponibilità (AZs) in un'unica Regione AWS zona di disponibilità e ogni zona di disponibilità contiene una copia dei dati del volume del cluster. Questa funzionalità consente al cluster DB di tollerare il guasto di una zona di disponibilità senza perdita di dati e con solo una breve interruzione del servizio.

In caso di guasto dell'istanza primaria di un cluster di database, Aurora esegue automaticamente il failover su una nuova istanza primaria, adottando uno dei seguenti due metodi:

- Promuovendo una replica Aurora esistente come nuova istanza primaria.
- Creando una nuova istanza primaria

Se il cluster di database include una o più repliche di Aurora, una di questeAurora verrà promossa come istanza primaria durante un evento di errore. Un evento di errore ha come conseguenza una breve interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Tuttavia, il servizio viene in genere ripristinato in meno di 60 secondi e spesso in meno di 30 secondi. Per aumentare la disponibilità del cluster DB, consigliamo di creare almeno una o più repliche Aurora in due o più due zone di disponibilità.

> **Tip**
>
> In Aurora MySQL, è possibile migliorare la disponibilità durante un failover disponendo di più di un'istanza DB di lettura in un cluster. In Aurora MySQL, Aurora riavvia solo l'istanza Writer DB e l'istanza reader su cui esegue il failover. Altre istanze di lettura nel cluster rimangono disponibili durante un failover per continuare a elaborare le query tramite connessioni all'endpoint di lettura.
> È inoltre possibile migliorare la disponibilità durante un failover utilizzando RDS Proxy con il cluster database di Aurora database. Per ulteriori informazioni, consulta Elevata disponibilità con Amazon RDS Proxy.

Puoi personalizzare l'ordine di promozione a istanza principale delle repliche Aurora dopo un guasto assegnando una priorità a ciascuna di esse. Le priorità vanno da 0, quella massima, a 15, quella minima. Se l'istanza primaria non va a buon fine, Amazon RDS promuove la replica di Aurora a istanza primaria con la massima priorità. Puoi modificare la priorità di una replica Aurora in qualsiasi momento. Modificando una priorità non attiverai un failover.

Più repliche Aurora possono condividere la stessa priorità, avendo come risultato livelli di promozione. Se due o più repliche di Aurora condividono la stessa priorità, Amazon RDS promuove quella di dimensioni maggiori. Se due o più repliche di Aurora condividono la stessa priorità e dimensioni, Amazon RDS ne promuove una arbitrariamente nello stesso livello di promozione.

> **Note**
>
> Nell'identificazione di un obiettivo di failover intervengono diversi fattori. Dopo cinque tentativi di failover falliti, i livelli di promozione non vengono più presi in considerazione.

Se il cluster database non contiene repliche Aurora, l'istanza primaria viene ricreata nello stesso AZ durante un evento di errore. Un evento di errore ha come conseguenza un'interruzione, durante la quale le operazioni di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione. Il servizio viene ripristinato quando crei la nuova istanza primaria. In genere, ciò richiede meno di 10 minuti. La promozione di una replica Aurora a istanza primaria è un'operazione molto più veloce rispetto alla creazione di una nuova istanza primaria.

Si assuma che l'istanza primaria nel cluster non sia disponibile a causa di un'interruzione che influisce su una intera zona di disponibilità. In questo caso, il modo per portare online una nuova istanza primaria varia in base all'utilizzo del cluster di una configurazione Multi-AZ:

- Se ti è stato fornito o Aurora Serverless v2 il cluster contiene qualsiasi istanza di lettore in un altro AZs, Aurora utilizza il meccanismo di failover per promuovere una di quelle istanze di lettore a diventare la nuova istanza primaria.
- Se hai effettuato il provisioning o Aurora Serverless v2 Il cluster contiene solo una singola istanza DB oppure, se l'istanza principale e tutte le istanze Reader si trovano nella stessa AZ, assicurati di creare manualmente una o più nuove istanze DB in un'altra AZ.
- Se il tuo cluster utilizza Aurora Serverless v1, Aurora crea automaticamente una nuova istanza DB in un'altra AZ. Tuttavia, questo processo comporta una sostituzione dell’host e richiede quindi un tempo di failover maggiore.

> **Note**
>
> Amazon Aurora supporta anche la replica tramite un database MySQL esterno o un'istanza database RDS MySQL. Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Elevata disponibilità con Amazon RDS Proxy

Con RDS Proxy, è possibile creare applicazioni in grado di tollerare in modo trasparente gli errori del database senza dover scrivere codice di gestione degli errori complesso. Il proxy instrada automaticamente il traffico verso una nuova istanza database preservando le connessioni alle applicazioni. Inoltre, ignora le cache DNS (Domain Name System) per ridurre i tempi di failover fino al 66% per i database Aurora Multi-AZ. Per ulteriori informazioni, consulta Proxy Amazon RDS per Aurora.

Replica con Amazon Aurora

Esistono varie opzioni di replica con Aurora. Ogni cluster di database Aurora dispone di una replica incorporata tra più istanze database nello stesso cluster. Inoltre, è possibile impostare la replica con il cluster Aurora come origine o destinazione. Quando si replicano dati all'interno o all'esterno di un cluster Aurora, è possibile scegliere tra funzionalità incorporate come database globali Aurora o meccanismi di replica tradizionali per i motori MySQL o PostgreSQL DB. È possibile scegliere le opzioni appropriate in base alle quali offrire la giusta combinazione di disponibilità elevata, convenienza e prestazioni per le esigenze specifiche. Le seguenti sezioni spiegano come e quando scegliere ogni tecnica.

Argomenti

- Repliche di Aurora
- Replica con Aurora MySQL
- Replica con Aurora PostgreSQL

Repliche di Aurora

Quando si crea una seconda istanza database (o una terza e così via) in un cluster di database con provisioning di Aurora, Aurora imposta automaticamente la replica dall'istanza database writer a tutte le altre istanze database. Queste altre istanze database sono di sola lettura e sono note come repliche di Aurora. Vengono anche definite istanze reader in relazione ai modi in cui è possibile combinare istanze database writer e reader all'interno di un cluster.

Le repliche di Aurora hanno due scopi principali. È possibile inviare query per ridimensionare le operazioni di lettura per l'applicazione. Generalmente, si esegue la connessione all'endpoint di lettura del cluster. In questo modo, Aurora può distribuire il carico per le connessioni di sola lettura su tutte

le repliche di Aurora presenti nel cluster. Le repliche di Aurora contribuiscono anche ad aumentare la disponibilità. Se l'istanza writer in un cluster non è disponibile, Aurora promuove automaticamente una delle istanze reader affinché prenda il suo posto come nuovo writer.

Un cluster di database di Aurora può contenere fino a 15 repliche di Aurora. Le repliche Aurora possono essere distribuite nelle zone di disponibilità su cui si estende un cluster DB all'interno di una regione. AWS

I dati nel cluster di database dispongono di funzionalità di elevata disponibilità e affidabilità, indipendentemente dalle istanze database nel cluster. Se hai familiarità con le funzionalità di storage di Aurora, consulta Panoramica dell'archiviazione di Amazon Aurora. Il volume del cluster di database si compone di più copie dei dati per il cluster di database. L'istanza primaria e le repliche di Aurora nel cluster di database vedono tutti i dati nel volume cluster come un singolo volume logico.

Di conseguenza, tutte le repliche di Aurora restituiscono gli stessi dati per i risultati di query con un ritardo di replica minimo. Questo ritardo di replica è solitamente molto inferiore a 100 millisecondi dopo che l'istanza primaria scrive un aggiornamento. Il ritardo di replica varia in base alla velocità di modifica del database. Pertanto, nei periodi in cui si verificano numerose operazioni di scrittura per il database, potresti riscontrare un aumento del ritardo di replica.

> **Note**
>
> Aurora Replica si riavvia automaticamente quando perde la comunicazione con l'istanza Writer DB per più di 60 secondi nelle seguenti versioni di Aurora PostgreSQL:
>
> - 14.6 e versioni precedenti
> - 13.9 e versioni precedenti
> - 12.13 e versioni precedenti
> - Tutte le versioni di Aurora PostgreSQL 11
>
> Con la funzionalità di disponibilità in lettura, le repliche Aurora non si riavviano automaticamente. Per ulteriori informazioni sulla funzionalità di disponibilità in lettura e sulle versioni in cui è applicabile, consulta. Miglioramento della disponibilità di lettura delle repliche Aurora

Le repliche Aurora funzionano bene per il dimensionamento della lettura perché sono dedicate completamente a operazioni di lettura nel volume cluster. Le operazioni di lettura sono gestite

dall'istanza primaria. Poiché il volume del cluster viene condiviso tra tutte le istanze database nel cluster di database, è necessaria solo una minima quantità di operazioni per replicare una copia dei dati per ciascuna replica di Aurora.

Per aumentare la disponibilità puoi servirti delle repliche di Aurora come target di failover. Pertanto se l'istanza primaria non va a buon fine, una replica di Aurora viene promossa a istanza primaria. Si verifica una breve interruzione durante la quale le richieste di lettura e scrittura inviate all'istanza primaria falliscono con un'eccezione.

Tuttavia, promuovere una replica di Aurora tramite failover è un'operazione molto più veloce rispetto a ricreare l'istanza primaria. Se il cluster di database Aurora non include nessuna replica Aurora, il cluster di database non sarà disponibile mentre l'istanza database esegue il ripristino dall'errore.

Quando si verifica il failover, alcune repliche di Aurora possono essere riavviate, a seconda della versione del motore di database. Ad esempio, in Aurora MySQL, Aurora riavvia solo l'istanza Writer DB e la destinazione di failover durante un failover. Per ulteriori informazioni sul comportamento di riavvio di diverse versioni del motore di database Aurora, consulta Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB. Per informazioni su cosa succede alle cache delle pagine durante il riavvio o il failover, consulta Cache delle pagine superstite.

Per gli scenari di disponibilità elevata, è consigliato creare una o più repliche di Aurora Dovrebbero essere della stessa classe delle istanze database dell'istanza primaria e in diverse zone di disponibilità per il cluster DB Aurora. Per ulteriori informazioni sulle repliche Aurora come destinazioni di failover, consulta Tolleranza ai guasti di un cluster DB Aurora.

Non è possibile creare una replica Aurora crittografata per un cluster database Aurora non crittografato. Non è possibile creare una replica Aurora non crittografata per un cluster database Aurora crittografato.

> **Tip**
>
> È possibile utilizzare le repliche di Aurora all'interno di un cluster di Aurora come unica forma di replica per mantenere i dati altamente disponibili. È possibile anche combinare la replica di Aurora incorporata con gli altri tipi di replica. In questo modo è possibile fornire un livello superiore di disponibilità elevata e distribuzione geografica dei dati.

Per informazioni dettagliate su come creare una replica di Aurora, consulta Aggiunta di repliche di Aurora a un cluster di database.

Replica con Aurora MySQL

Oltre alle repliche di Aurora, sono disponibili le seguenti opzioni di replica con Aurora MySQL:

- Cluster Aurora MySQL DB in diverse regioni. AWS
  - È possibile replicare i dati in più regioni utilizzando un database globale di Aurora. Per dettagli, consultare Elevata disponibilità nelle regioni AWS con database globali Aurora.
  - È possibile creare una replica di lettura Aurora di un cluster Aurora MySQL DB in una AWS regione diversa, utilizzando la replica del log binario MySQL (binlog). Ogni cluster può avere fino a cinque repliche di lettura create in questo modo, ciascuna in una regione geografica diversa.
- Due cluster database Aurora MySQL nella stessa regione , utilizzando una replica del log binario (binlog) MySQL.
- Una istanza database RDS per MySQL come origine di dati e un cluster database Aurora MySQL, creando una replica di lettura Aurora di una istanza database RDS per MySQL. In genere, è possibile utilizzare questo approccio per la migrazione a Aurora MySQL piuttosto che per le repliche in corso.

Per ulteriori informazioni sulla replica con Aurora MySQL, consulta Replica con Amazon Aurora MySQL.

Replica con Aurora PostgreSQL

Oltre alle repliche di Aurora, sono disponibili le seguenti opzioni di replica con Aurora PostgreSQL:

- Un cluster DB primario Aurora in una regione e fino a 10 cluster DB secondari di sola lettura in diverse regioni utilizzando un database globale Aurora. In altre parole, Aurora PostgreSQL non supporta le repliche Aurora tra regioni. Tuttavia, puoi utilizzare il database globale Aurora per scalare le funzionalità di lettura del cluster Aurora PostgreSQL DB su più di una regione e per raggiungere gli obiettivi di disponibilità. AWS Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.
- Due cluster di database Aurora PostgreSQL nella stessa regione, utilizzando la caratteristica di replica logica di PostgreSQL.
- Un'istanza del database RDS for PostgreSQL come origine dei dati e un cluster di database Aurora PostgreSQL, creando una replica di lettura Aurora di un'istanza database RDS for PostgreSQL. In genere, è possibile utilizzare questo approccio per la migrazione a Aurora PostgreSQL piuttosto che per le repliche in corso.

Per ulteriori informazioni sulla replica con Aurora PostgreSQL, consulta Replica con Amazon Aurora PostgreSQL.

DB per Aurora

Le istanze RDS fornite da Amazon in un cluster Amazon Aurora vengono fatturate in base ai seguenti componenti:

- Ore dell'istanza database (all'ora) – In base alla classe di istanza database (ad esempio, db.t2.small or db.m4.large). I prezzi sono indicati su base oraria, ma le fatture sono calcolate al secondo e mostrano gli orari in forma decimale. RDSI'utilizzo viene fatturato in incrementi di 1 secondo, con un minimo di 10 minuti. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.
- Storage (per GiB al mese) – Capacità di storage assegnata all'istanza database. Se la capacità di storage assegnata viene dimensionata nel corso del mese, l'addebito sarà ripartito proporzionalmente. Per ulteriori informazioni, consulta Archiviazione Amazon Aurora.
- Input/output (I/O) requests (per 1 million requests) – Total number of storage I/Orichieste che hai effettuato in un ciclo di fatturazione, per lo Solo configurazione del cluster DB.

  Per ulteriori informazioni sulla fatturazione I/O in Amazon Aurora, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.
- Storage di backup (per GiB al mese) – Lo storage di backup è lo storage associato ai backup di database automatici e a qualsiasi snapshot DB attivo acquisito. Estendendo il periodo di retention dei backup o creando ulteriori snapshot del database, si aumenta lo storage di backup consumato dal database. La fatturazione per secondo non si applica allo storage di backup (misurato in GB/mese).

  Per ulteriori informazioni, consulta Backup e ripristino di un cluster DB Amazon Aurora.
- Trasferimento dati (per GB): trasferimento di dati in entrata e in uscita dall'istanza DB da o verso Internet e altre regioni. AWS Per esempi utili, consulta il post del AWS blog Exploring Data Transfer Costs for AWS Managed Databases.

Amazon RDS offre le seguenti opzioni di acquisto per consentirti di ottimizzare i costi in base alle tue esigenze:

- Istanze on demand – Paghi all'ora per le ore dell'istanza database usate. I prezzi sono indicati su base oraria, ma le fatture sono calcolate al secondo e mostrano gli orari in forma decimale. RDSl'utilizzo viene ora fatturato in incrementi di 1 secondo, con un minimo di 10 minuti.
- Istanze riservate – Prenotando un'istanza database con durata di un anno o tre anni, puoi ricevere uno sconto significativo rispetto ai prezzi delle istanze database on demand. Grazie all'uso delle istanze riservate, è possibile avviare, eliminare e arrestare più istanze entro un'ora e ottenere il vantaggio delle istanze riservate per tutte le istanze.
- Aurora Serverless v2 – Aurora Serverless v2 fornisce capacità su richiesta in cui l'unità di fatturazione è l'unità di capacità Aurora (ACU) ore anziché le ore dell'istanza DB. Aurora Serverless v2 la capacità aumenta e diminuisce, entro un intervallo specificato dall'utente, a seconda del carico sul database. È possibile configurare un cluster in cui è presente tutta la capacità Aurora Serverless v2. Oppure puoi configurare una combinazione di Aurora Serverless v2 e istanze con provisioning su richiesta o riservate. Per informazioni su come Aurora Serverless v2 ACUslavoro, vediIn che modo Aurora Serverless v2 funzionamento.
- Aurora Postgre SQL Limitless Database — Aurora SQL Postgre Limitless Database è una funzionalità di scalabilità orizzontale automatizzata che supera il throughput di scrittura e i limiti di archiviazione di una singola istanza DB. Limitless Database distribuisce il carico di lavoro su più istanze di Aurora writer, pur mantenendo la facilità di funzionamento come un unico database. Limitless Database fornisce capacità su richiesta in cui l'unità di fatturazione è l'unità di capacità Aurora (ACU) ore in un gruppo di shard DB. Per informazioni sul funzionamento di Limitless Database, consulta. ACUs Creazione di un cluster DB che utilizza Aurora SQL Postgre Limitless Database

Per informazioni sui prezzi di Aurora, consulta la pagina dei prezzi di Aurora.

Argomenti

- Istanze database on demand per Aurora
- Istanze DB riservate per Amazon Aurora

Istanze database on demand per Aurora

Le istanze database on demand Amazon RDS vengono fatturate in base alla classe di istanza database, ad esempio db.t3.small o db.m5.large. Per informazioni sui prezzi di Amazon RDS, consulta la pagina del prodotto Amazon RDS.

La fatturazione per un'istanza database ha inizio non appena l'istanza database è disponibile. I prezzi sono calcolati in base a una tariffa oraria, mentre le fatture sono calcolate al secondo e mostrano i valori in formato decimale. L'utilizzo di Amazon RDS viene fatturato in incrementi di un secondo, con un minimo di 10 minuti. In caso di modifica della configurazione fatturabile, come l'elaborazione della scalabilità o la capacità di storage, viene effettuato l'addebito di un minimo di 10 minuti. La fatturazione continua finché l'istanza database non viene terminata, il che avviene quando elimini l'istanza database o in caso di errore dell'istanza database.

Se non vuoi più pagare per l'istanza database, devi arrestare o eliminarla in modo da evitare la fatturazione di altre ore dell'istanza database. Per ulteriori informazioni sugli stati dell'istanza database fatturata, consulta Visualizzazione dello stato dell'istanza database di in un cluster Aurora.

Istanze database arrestate

Mentre l'istanza database è arrestata, ti viene addebitato lo storage assegnato, incluso lo storage Provisioned IOPS. Ti viene addebitato anche lo storage dei backup, incluso quello per gli snapshot manuali e i backup automatici all'interno della finestra di retention specificata. Non è previsto alcun addebito per le ore dell'istanza database.

Istanze database Multi-AZ

Una configurazione Multi-AZ migliora la durabilità e la disponibilità dei dati effettuando automaticamente il provisioning e la manutenzione di una replica sincrona in standby in una zona di disponibilità diversa. Grazie alle risorse aggiuntive e alla maggiore disponibilità, le implementazioni Multi-AZ hanno un prezzo più elevato rispetto alle implementazioni Single-AZ e possono costare circa il doppio grazie all'istanza di standby aggiuntiva e alle risorse associate.

Considerate i seguenti importanti dettagli sui prezzi Multi-AZ:

- Costi di calcolo: fatturati per ora di istanza DB sia per l'istanza principale che per quella di standby.
- Costi di storage: addebitati per GB al mese per lo storage fornito sia per l'istanza principale che per quella di standby.

- Costi di trasferimento dei dati: la replica tra l'istanza principale e l'istanza di standby è inclusa nel costo, ma potrebbero essere applicati altri costi di trasferimento dei dati in base all'utilizzo.

Per stimare con precisione i costi mensili in base al caso d'uso specifico Regione AWS, puoi utilizzare il. Calcolatore dei prezzi AWS Questo strumento consente di inserire i dettagli di configurazione e fornisce una ripartizione completa dei costi.

> **Note**
>
> I prezzi sono soggetti a modifiche. Per up-to-date maggiori informazioni, consulta la pagina dei prezzi di Amazon RDS.

Istanze DB riservate per Amazon Aurora

Con le istanze database riservate puoi prenotare un'istanza database per un periodo di uno o tre anni. Le istanze database riservate offrono una notevole riduzione di prezzo rispetto alle istanze database on demand. Le istanze database riservate non sono istanze fisiche, ma piuttosto si tratta di uno sconto sulla fattura applicato all'uso di determinate istanze database nell'account. Gli sconti per le istanze database riservate sono legati al tipo di istanza e alla Regione AWS.

Il processo generale per l'uso delle istanze database riservate è il seguente: prima di tutto ottieni informazioni sulle offerte disponibili per le istanze database riservate, quindi acquisti un'offerta di istanza database riservata e infine ottieni informazioni sulle tue istanze database riservate esistenti.

Per informazioni sull'acquisto di istanze DB riservate e sulla visualizzazione della fatturazione per le istanze DB riservate, consulta le seguenti sezioni.

•
•

Panoramica delle istanze database riservate

Quando acquisti un'istanza database riservata in Amazon RDS, acquisti un impegno che ti permette di ottenere una tariffa scontata per un tipo di istanza database specifico, per la durata dell'istanza database riservata. Per usare un'istanza database Amazon RDS riservata, devi creare un'istanza database con una procedura analoga a quella per la creazione di un'istanza on demand.

La nuova istanza database creata deve avere le stesse specifiche dell'istanza database riservata relativamente a quanto segue:

- Regione AWS
- Motore DB (non è necessario che il numero di versione del motore DB corrisponda).
- Tipo di istanza database

Se le specifiche della nuova istanza database corrispondono a un'istanza database riservata esistente per il tuo account, viene fatturata la tariffa scontata per l'istanza database riservata. In caso contrario, l'istanza database viene fatturata in base a una tariffa on demand.

Puoi modificare un'istanza database che usi come istanza database riservata. Se la modifica rientra nelle specifiche dell'istanza database riservata, la parte o la totalità dello sconto viene comunque

applicata all'istanza database modificata. Se la modifica è al di fuori alle specifiche, ad esempio la modifica della classe di istanza, lo sconto non viene più applicato. Per ulteriori informazioni, consulta Istanze database riservate con dimensioni flessibili.

Argomenti

- Tipi offerta
- Flessibilità della configurazione del cluster database Aurora
- Istanze database riservate con dimensioni flessibili
- Esempi di fatturazione di istanze database riservate Aurora
- Eliminazione di un'istanza database riservata

Per ulteriori informazioni sulle istanze riservate e sui relativi prezzi, consulta Istanze riservate di Amazon RDS.

Tipi offerta

Le istanze database riservate disponibili sono di tre tipi, ovvero —nessun pagamento anticipato, pagamento anticipato parziale e pagamento anticipato dell'intero costo—per permetterti di ottimizzare i costi di Amazon RDS in base all'utilizzo previsto.

> **Note**
>
> Non tutte le classi di istanze RDS supportano tutti i tipi di offerta di istanze riservate. Ad esempio, alcune classi di istanze potrebbero non offrire l'opzione No Upfront. Per confermare la disponibilità, esamina le offerte di istanze riservate in AWS Management Console o utilizza il `describe-reserved-db-instances-offerings` AWS CLI comando.

Nessun pagamento anticipato

Questa opzione permette di accedere a un'istanza database riservata senza un pagamento anticipato. L'istanza riservata senza pagamento anticipato viene fatturata applicando una tariffa oraria scontata per ogni ora durante il periodo della prenotazione, indipendentemente dall'utilizzo, e non è richiesto alcun pagamento anticipato. Questa opzione è disponibile solo per le prenotazioni della durata di un anno.

Pagamento anticipato parziale

Questa opzione richiede il pagamento anticipato di una parte dell'istanza database riservata. Le ore rimanenti del periodo di prenotazione vengono fatturate a una tariffa oraria scontata, indipendentemente dall'utilizzo. Questa opzione sostituisce l'opzione precedente per utilizzo pesante.

Pagamento anticipato intero costo

Il pagamento viene effettuato per intero all'inizio del periodo della prenotazione e non vengono addebitati altri costi per il resto del periodo, indipendentemente dal numero di ore di utilizzo.

Se usi la fatturazione consolidata, tutti gli account all'interno dell'organizzazione vengono trattati come se fossero un account unico. Questo significa che tutti gli account di un'organizzazione possono usufruire del vantaggio in termini di costi orari delle istanze database riservate acquistate da un altro account. Per ulteriori informazioni sulla fatturazione consolidata, consulta Istanze database riservate di Amazon RDS nella AWS Guida per l'utente di Billing and Cost Management.

Flessibilità della configurazione del cluster database Aurora

È possibile utilizzare le istanze database riservate di Aurora con entrambe le configurazioni di cluster database:

- Aurora I/O-Optimized — Paghi solo per l'utilizzo e l'archiviazione dei tuoi cluster DB, senza costi aggiuntivi per le operazioni di I/O di lettura e scrittura.
- Aurora Standard — Oltre all'utilizzo e all'archiviazione dei cluster DB, paghi anche una tariffa standard per 1 milione di richieste per le operazioni di I/O.

Aurora tiene automaticamente conto della differenza di prezzo tra queste configurazioni. Aurora I/O-Optimized consuma il 30% in più di unità normalizzate all'ora rispetto a Aurora Standard.

Per ulteriori informazioni sulle configurazioni dell'archiviazione dei cluster Aurora, consultare Configurazioni dell'archiviazione per i cluster database Amazon Aurora. Per informazioni sui prezzi relativi alle configurazioni dell'archiviazione dei cluster Aurora, consulta la pagina dei prezzi di Amazon Aurora.

Istanze database riservate con dimensioni flessibili

Quando acquisti un'istanza database riservata, devi specificare la classe di istanza, ad esempio db.r5.large. Per altre informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora.

Se hai un'istanza database e devi dimensionarla per aumentarne la capacità, l'istanza database riservata viene applicata automaticamente all'istanza database ridimensionata. Ciò significa che le istanze database riservate vengono applicate automaticamente a tutte le dimensioni di classi di istanze database. Le istanze DB riservate flessibili in termini di dimensioni sono disponibili per le istanze DB con lo stesso motore di database. Regione AWS Le istanze database riservate con dimensioni flessibili sono scalabili solo nel loro tipo di classe istanza. Ad esempio, un'istanza database riservata per db.r5.large è applicabile a db.r5.xlarge, ma non a db.r6g.large, perché db.r5 e db.r6g sono tipi di classe istanza diversi.

Le istanze database riservate con dimensioni flessibili sono disponibili per i motori di database Aurora seguenti:

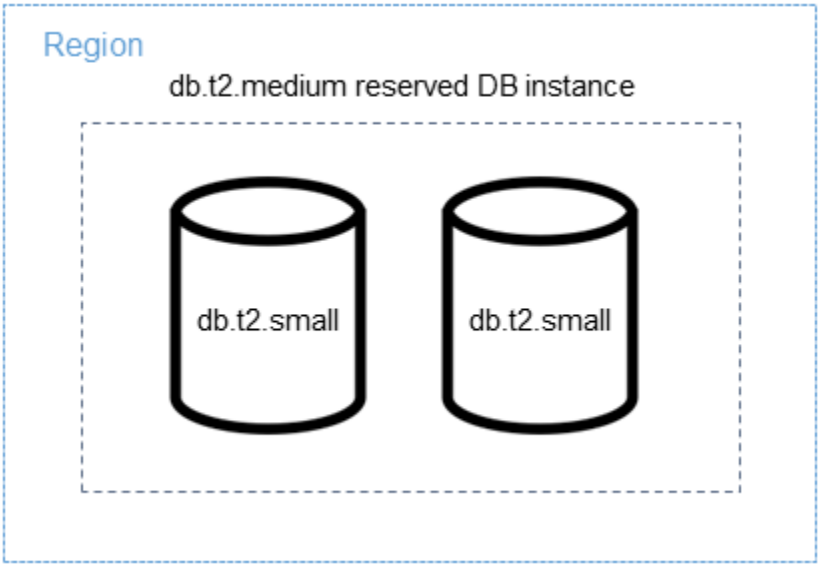
- Aurora MySQL
- Aurora PostgreSQL

È possibile confrontare l'utilizzo per le diverse dimensioni di istanze database riservate usando unità normalizzate all'ora. Ad esempio, un'unità di utilizzo in due istanze database db.r3.large equivale a 8 unità normalizzate di utilizzo all'ora in un'istanza db.r3.small. La tabella seguente mostra il numero di unità normalizzate all'ora per ogni dimensione di istanza database.

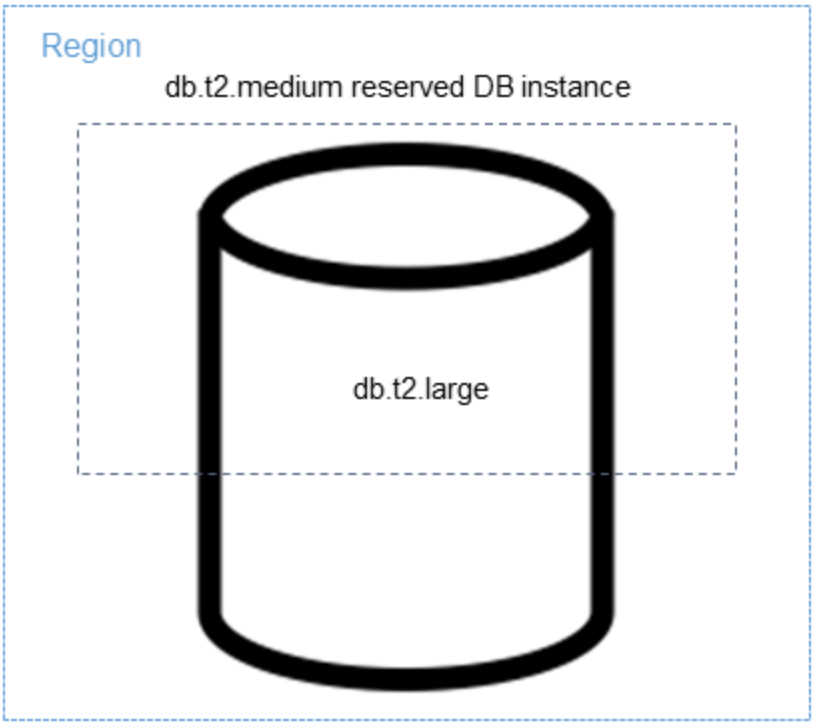
Dimensioni istanza	Unità normalizzate all'ora per un'istanza DB, Aurora Standard	Unità normalizzate all'ora per un'istanza DB, Aurora I/O-Optimized	Unità all'ora normalizzate per tre istanze DB (writer e due lettori), Aurora Standard	Unità orarie normalizzate per tre istanze DB (writer e due lettori), Aurora I/O-Optimized
small	1	1.3	3	3.9
medium	2	2.6	6	7.8
large	4	5.2	12	15,6

Dimensioni istanza	Unità normalizzate all'ora per un'istanza DB, Aurora Standard	Unità normalizzate all'ora per un'istanza DB, Aurora I/O-Optimized	Unità all'ora normalizzate per tre istanze DB (writer e due lettori), Aurora Standard	Unità orarie normalizzate per tre istanze DB (writer e due lettori), Aurora I/O-Optimized
xlarge	8	10.4	24	31,2
2xlarge	16	20,8	48	62,4
4xlarge	32	41,6	96	124,8
8xlarge	64	83,2	192	249,6
12xlarge	96	124,8	288	374,4
16xlarge	128	166,4	384	499,2
24xlarge	192	249,6	576	748,8
32xlarge	256	332,8	768	998,4

Supponiamo ad esempio che acquisti un'istanza database riservata `db.t2.medium` e che siano presenti due istanze database `db.t2.small` in esecuzione nel tuo account nella stessa Regione AWS. In questo caso, il vantaggio di fatturazione viene applicato per intero a entrambe le istanze.



In alternativa, se nel tuo account è in esecuzione un'`db.t2.large`istanza nello stesso account Regione AWS, il vantaggio di fatturazione viene applicato al 50 percento dell'utilizzo dell'istanza DB.



> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.

Esempi di fatturazione di istanze database riservate Aurora

Gli esempi seguenti illustrano i prezzi delle istanze DB riservate per i cluster Aurora DB che utilizzano entrambi i Aurora Standard e Aurora I/O-Optimized Configurazioni del cluster DB.

Esempio di utilizzo Aurora Standard

Il prezzo di un'istanza database riservata non prevede uno sconto per i costi associati all'archiviazione, ai backup e all'I/O. L'esempio seguente illustra il costo totale mensile di un'istanza database riservata:

- Una classe di istanza database Single-AZ db.r5.large riservata in Aurora MySQL nella regione Stati Uniti orientali (Virginia settentrionale) al costo di 0,19 USD all'ora o di 138,70 USD al mese
- Storage Aurora al costo di 0,10 USD per GiB al mese (per questo esempio, supponiamo 45,60 USD al mese)
- I/O Aurora al costo di 0,20 USD per 1 milione di richieste (per questo esempio, supponiamo 20 USD al mese)
- Storage di backup Aurora a 0,021 USD per GiB al mese (per questo esempio, supponiamo 30 USD al mese)

Sommando tutte queste opzioni (138,70 USD + 45,60 USD + 20 USD + 30 USD) all'istanza database riservata, il costo mensile totale è di 234,30 USD.

Se hai scelto di usare un'istanza database on demand anziché un'istanza database riservata, una classe di istanza database Single-AZ db.r5.large di Aurora MySQL nella regione Stati Uniti orientali (Virginia settentrionale) costa 0,29 USD all'ora o 217,50 USD al mese. Quindi, per un'istanza database on demand, sommando tutte queste opzioni (217,50 USD + 45,60 USD + 20 USD+ 30 USD), il costo mensile totale è di 313,10 USD. Risparmi quasi $79 al mese utilizzando l'istanza database riservata.

Esempio di utilizzo di un Aurora Standard Cluster DB con due istanze di lettura

Per utilizzare le istanze riservate per i cluster database Aurora, è sufficiente acquistare un'istanza riservata per ogni istanza database del cluster.

Estendendo il primo esempio, si dispone di un cluster database Aurora MySQL con un'istanza database di scrittura e due repliche Aurora, per un totale di tre istanze database nel cluster. Le due repliche Aurora non comportano costi aggiuntivi di archiviazione o backup. Se si acquistano tre istanze riservate Aurora MySQL db.r5.large, il costo sarà pari a 234,30 USD (per l'istanza database di scrittura) + 2 * (138,70 USD + 20 USD I/O per la replica Aurora), per un totale di 551,70 USD al mese.

Il costo on-demand corrispondente per un cluster database Aurora MySQL con un'istanza database di scrittura e due repliche Aurora è di $ 313,10 + 2 * ($ 217,50 + $ 20 I/O per istanza) per un totale di $ 788,10 al mese. Risparmi quasi $ 236,40 al mese utilizzando l'istanza database riservata.

Esempio di utilizzo Aurora I/O-Optimized

Puoi riutilizzare quello esistente Aurora Standard istanze DB riservate con Aurora I/O-Optimized. Per sfruttare appieno i vantaggi dei tuoi sconti sulle istanze riservate con Aurora I/O-Optimized, puoi acquistare il 30% di istanze riservate aggiuntive simili alle tue attuali istanze riservate.

La tabella seguente mostra esempi di come stimare le istanze riservate aggiuntive durante l'utilizzo Aurora I/O-Optimized. Se le istanze riservate richieste sono una frazione, puoi sfruttare la flessibilità delle dimensioni disponibile con le istanze riservate per ottenere un numero intero. In questi esempi, «corrente» si riferisce a Aurora Standard istanze riservate che hai ora. Le istanze riservate aggiuntive sono il numero di Aurora Standard istanze riservate che è necessario acquistare per mantenere gli attuali sconti sulle istanze riservate durante l'utilizzo Aurora I/O-Optimized.

DB instance class (Classe istanza database)	Attuali Aurora Standard istanze riservate	Istanze riservate richieste per Aurora I/O-Optimized	Istanze riservate aggiuntive necessarie	Istanze riservate aggiuntive necessarie, utilizzando la flessibilità delle dimensioni
db.r6g.large	10	10 * 1,3 = 13	3 * db.r6g.large	3 * db.r6g.large

DB instance class (Classe istanza database)	Attuali Aurora Standard istanze riservate	Istanze riservate richieste per Aurora I/O-Optimized	Istanze riservate aggiuntive necessarie	Istanze riservate aggiuntive necessarie, utilizzando la flessibilità delle dimensioni
db.r6g.4xlarge	20	20 * 1,3 = 26	6 * db.r6g.4xlarge	6 * db.r6g.4xlarge
db.r6g.12xlarge	5	5 * 1,3 = 6,5	1,5 * db.r6g.12xlarge	Uno per ogni istanza db.r6g.12xlarge, r6g.4xlarge e r6g.2xlarge (0,5 * db.r6g.12xlarge = 1 * db.r6g.4xlarge + 1 * db.r6g.2xlarge)
db.r6i.24xlarge	15	15 * 1,3 = 19,5	4,5 * db.r6i.24xlarge	4 * db.r6i.24xlarge + 1 * db.r6i.12xlarge (0,5 * db.r6i.24xlarge = 1 * db.r6i.12xlarge)

Esempio di utilizzo di un Aurora I/O-Optimized Cluster DB con due istanze di lettura

È presente un cluster database Aurora MySQL con un'istanza database di scrittura e due repliche Aurora, per un totale di tre istanze database nel cluster. Usano il Aurora I/O-Optimized Configurazione del cluster DB. Per utilizzare istanze database riservate per questo cluster, è necessario acquistare quattro istanze database riservate della stessa classe di istanze database. Tre istanze DB che utilizzano Aurora I/O-Optimized consumano 3,9 unità normalizzate all'ora, rispetto alle

3 unità normalizzate all'ora per tre istanze DB che utilizzano Aurora Standard. Tuttavia, si risparmiano i costi mensili di I/O per ogni istanza DB.

> **Note**
>
> I prezzi citati in questo esempio sono prezzi di esempio e potrebbero non corrispondere ai prezzi effettivi. Per informazioni sui prezzi di Aurora, consulta Prezzi di Amazon Aurora.

Eliminazione di un'istanza database riservata

Un'istanza database riservata può essere prenotata con un impegno per un periodo di un anno o di tre anni. Non è possibile annullare un'istanza database riservata. È comunque possibile eliminare un'istanza database coperta da uno sconto per istanza database riservata. Il processo di eliminazione di un'istanza database coperta da uno sconto per istanza database riservata è uguale a quello per l'eliminazione di qualsiasi altra istanza database.

I costi iniziali vengono fatturati indipendentemente dal fatto che si utilizzi le risorse.

Se elimini un'istanza database coperta da uno sconto per istanza database riservata, puoi avviare un'altra istanza database con specifiche compatibili. In questo caso, continuare a usufruire della tariffa scontata durante il periodo della prenotazione (un anno o tre anni).

Puoi utilizzare il AWS Management Console, il e il AWS CLI RDS API per lavorare con istanze DB riservate.

Console

È possibile utilizzarli AWS Management Console per lavorare con istanze DB riservate, come illustrato nelle seguenti procedure.

Per ottenere informazioni sui prezzi e sulle offerte disponibili per le istanze database riservate

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Reserved instances (Istanze riservate).
3. Scegliere Purchase Reserved DB Instance (Acquista istanza database riservata).
4. Per Product description (Descrizione prodotto) scegliere il motore di database e il tipo di licenza.
5. Per DB instance class (Classe istanza database), scegliere la classe di istanza database.

6. In Opzione di implementazione, specifica se desideri un'implementazione Single-AZ o Multi-AZ per le istanze database.

> **Note**
>
> Per le istanze Amazon Aurora riservate, l'opzione di implementazione è sempre impostata su Istanza database Single-AZ. Tuttavia, quando crei un cluster di database Aurora, l'opzione di implementazione predefinita è Crea una replica o una sessione di lettura Aurora in una zona di disponibilità diversa (Multi-AZ).
> È necessario acquistare un'istanza database riservata per ogni istanza che si intende utilizzare, incluse le repliche Aurora. Pertanto, per le implementazioni multi-AZ in Aurora, è necessario acquistare altre istanze database riservate.

7. In Periodo, scegli per quanto tempo riservare l'istanza database.
8. Per Offering type (Tipo di offerta), scegliere il tipo di offerta.

   Dopo aver selezionato il tipo di offerta, vengono visualizzate le informazioni sui prezzi.

> **Important**
>
> Scegliere Cancel (Annulla) per evitare di acquistare un'istanza database riservata con il conseguente addebito dei relativi costi.

Dopo avere ottenuto le informazioni sulle offerte disponibili per le istanze database riservate, puoi basarti su tali informazioni per acquistare un'offerta, come illustrato nella procedura seguente.

Per acquistare un'istanza database riservata

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Reserved instances (Istanze riservate).
3. Scegli Purchase reserved DB instance (Acquista istanza database riservata).
4. Per Product description (Descrizione prodotto) scegliere il motore di database e il tipo di licenza.
5. Per DB instance class (Classe istanza database), scegliere la classe di istanza database.
6. In Implementazione Multi-AZ, specifica se desideri un'implementazione Single-AZ o Multi-AZ per le istanze database.

> **Note**
>
> Per le istanze Amazon Aurora riservate, l'opzione di implementazione è sempre impostata su Istanza database Single-AZ. Quando si crea un cluster di database Amazon Aurora dall'istanza database riservata, il cluster di database viene creato automaticamente come Multi-AZ. Assicurati acquistare un'istanza database riservata per ogni istanza database che intendi utilizzare, incluse le repliche Aurora.

7. Per Term (Periodo), scegliere per quanto tempo prenotare l'istanza database.
8. Per Offering type (Tipo di offerta), scegliere il tipo di offerta.

   Dopo aver selezionato il tipo di offerta, vengono visualizzate le informazioni sui prezzi.
9. (Facoltativo) È possibile assegnare un identificatore alle istanze database riservate acquistate, per tenerne traccia. Per Reserved Id (ID istanza riservata), digitare un identificatore per l'istanza database riservata.
10. Scegli Invia.

    L'istanza database riservata viene acquistata, quindi visualizzata nell'elenco Reserved instances (Istanze riservate).

Dopo avere acquistato le istanze database riservate, puoi ottenere informazioni sulle tue istanze database riservate come illustrato nella procedura seguente.

Per ottenere informazioni sulle istanze DB riservate per il tuo account AWS

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Reserved Instances (Istanze riservate).

   Verranno visualizzate le istanze database riservate per l'account. Per visualizzare informazioni dettagliate su una particolare istanza database riservata, scegli l'istanza nell'elenco. Le informazioni dettagliate su quell'istanza verranno visualizzate nel riquadro dei dettagli nella parte inferiore della console.

AWS CLI

Puoi utilizzarlo AWS CLI per lavorare con istanze DB riservate, come mostrato negli esempi seguenti.

Example di recuperare le offerte disponibili per le istanze database riservate

Per ottenere informazioni sulle offerte disponibili di istanze DB riservate, chiamate il AWS CLI comando. [describe-reserved-db-instances-offerings]

```
aws rds describe-reserved-db-instances-offerings
```

Questa chiamata restituisce un output simile al seguente:

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

Dopo avere ottenuto le informazioni sulle offerte disponibili per le istanze database riservate, puoi basarti su tali informazioni per acquistare un'offerta.

Per acquistare un'istanza DB riservata, utilizza il AWS CLI comando [purchase-reserved-db-instances-offering]con i seguenti parametri:

- `--reserved-db-instances-offering-id` – L'ID dell'offerta da acquistare. Per ottenere l'ID dell'offerta, consulta l'esempio precedente.
- `--reserved-db-instance-id` – Puoi assegnare un identificatore alle istanze database riservate acquistate, per tenerne traccia.

Example di acquistare un'istanza database riservata

L'esempio seguente acquista l'offerta di istanze DB riservate con ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*e assegna l'identificatore di *MyReservation*.

PerLinux, omacOS: Unix

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Per Windows:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

Il comando restituisce un output simile al seguente:

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State           Description  Offering Type
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

Dopo avere acquistato le istanze database riservate, puoi ottenere informazioni sulle tue istanze database riservate.

Per ottenere informazioni sulle istanze DB riservate per il tuo AWS account, chiama il AWS CLI comando `describe-reserved-db-instances`, come mostrato nell'esempio seguente.

Example di ottenere le istanze DB riservate

```
aws rds describe-reserved-db-instances
```

Il comando restituisce un output simile al seguente:

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

RDS API

Puoi usare il RDS API per lavorare con istanze DB riservate:

- Per ottenere informazioni sulle offerte disponibili di istanze DB riservate, chiama l'ufficio RDS API operativo `DescribeReservedDBInstancesOfferings`Amazon.
- Dopo avere ottenuto le informazioni sulle offerte disponibili per le istanze database riservate, puoi basarti su tali informazioni per acquistare un'offerta. Chiama l'`PurchaseReservedDBInstancesOffering`RDSAPIoperazione con i seguenti parametri:
  - `--reserved-db-instances-offering-id` – L'ID dell'offerta da acquistare.
  - `--reserved-db-instance-id` – Puoi assegnare un identificatore alle istanze database riservate acquistate, per tenerne traccia.
- Dopo avere acquistato le istanze database riservate, puoi ottenere informazioni sulle tue istanze database riservate. Chiama l'`DescribeReservedDBInstances`RDSAPIoperazione.

Puoi visualizzare la fatturazione per le istanze database riservate nel pannello di controllo di fatturazione nella AWS Management Console.

Per visualizzare la fatturazione di istanze database riservate

1. Accedi alla AWS Management Console.
2. Dal menu account in alto a destra, scegliere Billing Dashboard (Pannello di controllo di fatturazione).
3. Scegliere Dettagli di fatturazione nell'angolo in alto a destra del pannello di controllo.
4. In Costi di servizio AWS , espandere Servizio di database relazionale.
5. Espandi la Regione AWS posizione in cui si trovano le tue istanze DB riservate, ad esempio US West (Oregon).

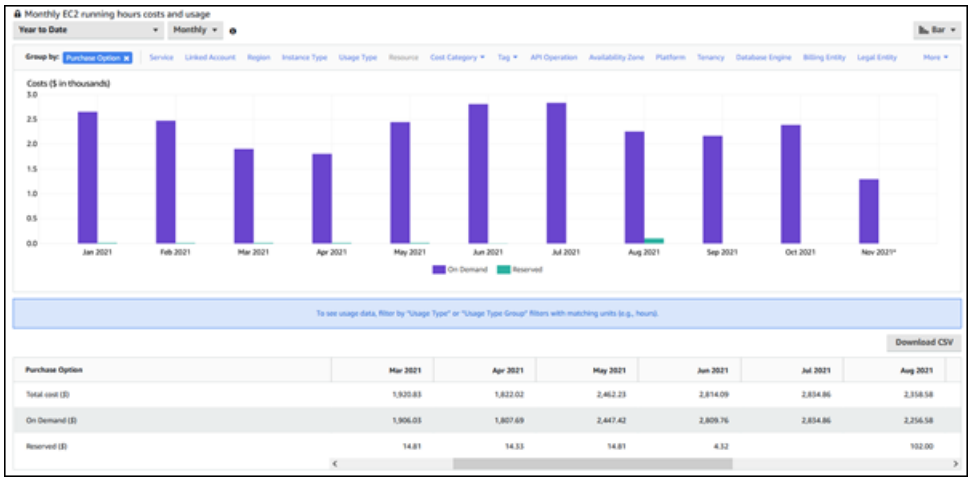
   Le tue istanze DB riservate e le relative tariffe orarie per il mese corrente sono mostrate in Amazon Relational Database Service per ***`Database Engine`*** Istanze riservate.

   L'istanza database riservata in questo esempio è stata acquistata con pagamento anticipato, quindi non ci sono costi orari.
6. Scegliere l'icona Cost Explorer (grafico a barre) accanto alla voce Istanze riservate.

Il Cost Explorer mostra il grafico dei costi e dell'utilizzo delle ore di EC2 funzionamento mensili.

7. Cancellare il filtro Usage Type Group (Gruppo tipi di utilizzo) a destra del grafico.
8. Scegliere il periodo di tempo e l'unità di tempo per la quale si desidera esaminare i costi di utilizzo.

   L'esempio seguente mostra i costi di utilizzo per istanze database on-demand e riservate per l'anno in corso per mese.



I costi delle istanza database riservate da gennaio a giugno 2021 sono oneri mensili per un'istanza con parziale pagamento anticipato, mentre il costo nell'agosto 2021 è un addebito una tantum per un'istanza con pagamento anticipato completo.

Lo sconto dell'istanza riservata per l'istanza con parziale pagamento anticipato è scaduto nel giugno 2021, ma l'istanza database non è stata eliminata. Dopo la data di scadenza, è stata semplicemente addebitata la tariffa on-demand.

Configurazione dell'ambiente per Amazon Aurora

Prima di usare Amazon Aurora per la prima volta, è necessario completare le seguenti operazioni:

Argomenti

- Registrati per un Account AWS
- Crea un utente con accesso amministrativo
- Concessione dell'accesso programmatico
- Determinazione dei requisiti
- Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza

Se ne possiedi già uno Account AWS, conosci i requisiti di Aurora e preferisci utilizzare le impostazioni predefinite per i gruppi di sicurezza IAM e VPC, passa subito a. Nozioni di base su Amazon Aurora

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la https://portal.aws.amazon.com/billing/registrazione.
2. Segui le istruzioni online.

   Parte della procedura di registrazione prevede la ricezione di una telefonata o di un messaggio di testo e l'immissione di un codice di verifica sulla tastiera del telefono.

   Quando ti iscrivi a un Account AWS, Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi AWS Management Consolecome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

   Segui le istruzioni riportate nella pagina Creazione di un set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

   Per istruzioni, consulta Aggiungere gruppi nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se vogliono interagire con l' AWS AWS Management Console esterno di. Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Determinazione dei requisiti

Il cluster di database rappresenta l'elemento di base di Aurora. Una o più istanze database possono appartenere a un cluster di database. Un cluster di database fornisce un indirizzo di rete denominato endpoint del cluster. Le applicazioni si connettono all'endpoint del cluster esposto dal cluster di database quando devono accedere ai database creati in quel cluster di database. Le informazioni

specificate al momento della creazione del cluster di database controllano elementi di configurazione come memoria, motore di database e versione, configurazione di rete, sicurezza e periodi di manutenzione.

Prima di creare un cluster DB e un gruppo di sicurezza, devi conoscere le necessità del cluster DB e della rete. Ecco alcune cose importanti da considerare:

- Requisiti delle risorse – Quali sono i requisiti di memoria e del processore per l'applicazione o il servizio? Utilizzerai queste impostazioni nella definizione della classe di istanza database da usare durante la creazione del cluster di database. Per specifiche sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora.
- VPC, sottorete e gruppo di sicurezza– Il cluster DB si trova in un Virtual Private Cloud (VPC). È necessario configurare le regole del gruppo di sicurezza per la connessione a un cluster di database. L'elenco seguente descrive le regole per ogni opzione VPC:
  - VPC predefinito: se il tuo AWS account ha un VPC predefinito nella regione AWS , quel VPC è configurato per supportare i cluster DB. Se specifichi il VPC predefinito quando crei il cluster di database:
    - Assicurati di creare un gruppo di sicurezza VPC che autorizza le connessioni dall'applicazione o dal servizio al cluster DB Aurora. Utilizza l'opzione Security Group sulla console VPC o AWS CLI per creare gruppi di sicurezza VPC. Per informazioni, consultare Fase 3: creazione di un gruppo di sicurezza VPC.
    - Devi specificare il gruppo di sottoreti del database predefinito. Se questo è il primo cluster DB creato nella AWS regione, Amazon RDS creerà il gruppo di sottoreti DB predefinito al momento della creazione del cluster DB.
  - VPC definito dall'utente: se vuoi specificare un VPC definito dall'utente quando crei un cluster di database:
    - Assicurati di creare un gruppo di sicurezza VPC che autorizza le connessioni dall'applicazione o dal servizio al cluster DB Aurora. Utilizza l'opzione Security Group sulla console VPC o AWS CLI per creare gruppi di sicurezza VPC. Per informazioni, consultare Fase 3: creazione di un gruppo di sicurezza VPC.
    - Il VPC deve soddisfare certi requisiti per ospitare cluster di database, come avere almeno due sottoreti, ognuna in una zona di disponibilità separata. Per informazioni, consulta Amazon VPC e Aurora.
    - Devi specificare un gruppo di sottoreti del database che definisce quali sottoreti in quel VPC possono essere utilizzate dal cluster di database. Per informazioni, consulta la sezione relativa al gruppo di sottoreti del database in Uso di un cluster database in un VPC.

- Elevata disponibilità: hai bisogno di supporto per il failover? In Aurora, un'implementazione Multi-AZ crea un'istanza primaria e repliche di Aurora. Puoi configurare l'istanza primaria e le repliche Aurora in modo che si trovino in zone di disponibilità diverse per il supporto per failover. Consigliamo implementazioni Multi-AZ per carichi di lavoro di produzione per mantenere alta disponibilità. Per scopi di sviluppo e test, puoi utilizzare una non-Multi-AZ distribuzione. Per ulteriori informazioni, consulta Elevata disponibilità di Amazon Aurora.
- Policy IAM: il tuo AWS account dispone di politiche che concedono le autorizzazioni necessarie per eseguire le operazioni di Amazon RDS? Se ti connetti AWS tramite credenziali IAM, il tuo account IAM deve disporre di policy IAM che concedano le autorizzazioni necessarie per eseguire le operazioni di Amazon RDS. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.
- Porte aperte: su quale TCP/IP porta sarà in ascolto il tuo database? Il firewall in alcune aziende può bloccare le connessioni alla porta predefinita del motore di database. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster database. Tieni presente che dopo avere creato un cluster di database in ascolto su una determinata porta, puoi cambiare la porta modificando il cluster di database.
- AWS Regione: in quale AWS regione vuoi inserire il tuo database? Se il database si trova in prossimità dell'applicazione o del servizio Web, la latenza di rete potrebbe ridursi. Per ulteriori informazioni, consulta Regioni e zone di disponibilità.

Quando disponi delle informazioni necessarie per creare il gruppo di sicurezza e il cluster di database, procedi alla fase successiva.

Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza

Il cluster DB verrà creato in un VPC. I gruppi di sicurezza forniscono l'accesso al cluster di database nel VPC. Fungono da firewall per il cluster di database associato, controllando il traffico sia in entrata che in uscita a livello di cluster. Per impostazione predefinita, i cluster di database vengono creati con un firewall e un gruppo di sicurezza predefinito che impedisce l'accesso al cluster di database. Pertanto, a un gruppo di sicurezza è necessario aggiungere regole che consentano di connettersi al cluster di database. Utilizza le informazione di rete e di configurazione specificate nella fase precedente per creare regole che consentano l'accesso al cluster di database.

Ad esempio, se hai un'applicazione che accederà a un database nel tuo cluster di database in un VPC, devi aggiungere una regola TCP personalizzata che specifichi l'intervallo di porte e gli indirizzi

IP che l'applicazione userà per accedere al database. Se hai un'applicazione su un' EC2 istanza Amazon, puoi utilizzare il gruppo di sicurezza VPC che hai configurato per l'istanza Amazon EC2.

Puoi configurare la connettività tra un' EC2 istanza Amazon e un cluster DB quando crei il cluster DB. Per ulteriori informazioni, consulta Configura la connettività di rete automatica con un'istanza EC2 .

> **Tip**
>
> Puoi configurare automaticamente la connettività di rete tra un' EC2 istanza Amazon e un cluster DB quando crei il cluster DB. Per ulteriori informazioni, consulta Configura la connettività di rete automatica con un'istanza EC2 .

Per informazioni su come connettere le risorse in Amazon Lightsail ai cluster DB, consulta Connettere le risorse Lightsail all'utilizzo del peering VPC. Servizi AWS

Per ulteriori informazioni sulla creazione di un VPC da utilizzare con Aurora, consulta Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 . Per informazioni sugli scenari comuni per l'accesso a un'istanza database, consult Scenari per l'accesso a un cluster di DB in un VPC.

Per creare un gruppo di sicurezza VPC

1. Accedi a AWS Management Console e apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com /vpc.

   > **Note**
   >
   > Assicurati di essere nella console VPC, non nella console RDS.

2. Nell'angolo in alto a destra di AWS Management Console, scegli la AWS regione in cui desideri creare il gruppo di sicurezza VPC e il cluster DB. Nell'elenco delle risorse Amazon VPC per quella regione AWS , dovresti vedere almeno un VPC e diverse sottoreti. In caso contrario, non hai un VPC predefinito in quella AWS regione.
3. Fare clic su Security Groups (Gruppi di sicurezza) nel pannello di navigazione.
4. Scegliere Create Security Group (Crea gruppo di sicurezza).

   Viene visualizzata la pagina Create security group (Crea gruppo di sicurezza).

5. In Basic details (Dettagli di base), immettere il Security group name (Nome del gruppo di sicurezza) e la Description (Descrizione). Per VPC, seleziona il VPC nel quale desideri creare il cluster DB.
6. Per Inbound rules (Regole in entrata), scegli Add rule (Aggiungi regola).
    a. Per Type (Tipo), scegliere Custom TCP (TCP personalizzato).
    b. Per Port Range (Intervallo porte), digita il valore della porta da utilizzare per il cluster DB.
    c. Per Source (Origine), seleziona il nome del gruppo di sicurezza o digita l'intervallo di indirizzi IP (valore CIDR) da dove accedi al cluster DB. Se scegli My IP (Il mio IP), questo consente l'accesso al cluster DB dall'indirizzo IP rilevato nel browser.
7. Se occorre aggiungere altri indirizzi IP o intervalli di porta diversi, scegliere Add rule (Aggiungi regola) e immettere le informazioni relative alla regola.
8. (Facoltativo) In Outbound Rules (Regole in uscita), aggiungi regole per il traffico in uscita. Come impostazione predefinita, tutto il traffico in uscita è permesso.
9. Scegliere Create Security Group (Crea gruppo di sicurezza).

Si può utilizzare il gruppo di sicurezza VPC appena creato come gruppo di sicurezza per il cluster di database al momento della creazione.

> **Note**
>
> Se utilizzi un VPC predefinito, viene creato un gruppo di sottoreti predefinito che include tutte le sottoreti VPC. Quando si crea un cluster di database, è possibile selezionare il VPC predefinito e utilizzare default (predefinito) per DB Subnet Group (Gruppo di sottoreti del database).

Quando hai completato i requisiti di configurazione, puoi creare un cluster DB utilizzando i requisiti e il gruppo di sicurezza seguendo le istruzioni in Creazione di un cluster database Amazon Aurora. Per informazioni su come iniziare con la creazione di un cluster DB che utilizza un motore DB specifico, consulta Nozioni di base su Amazon Aurora.

Nozioni di base su Amazon Aurora

In questa sezione, puoi scoprire come creare e connetterti a un cluster Aurora DB utilizzando Amazon. RDS

Le procedure seguenti sono dei tutorial che offrono le nozioni di base su Aurora. Le sezioni successive presentano procedure e concetti Aurora più avanzati, ad esempio i vari tipi di endpoint e le modalità di dimensionamento dei cluster Aurora.

> **Important**
>
> Devi completare le attività indicate nella sezione Configurazione dell'ambiente per Amazon Aurora prima di poter creare o connettersi ad un cluster database .

Argomenti

- Creazione e connessione a un cluster di database Aurora MySQL
- Creazione e connessione di un cluster di database Aurora PostgreSQL
- Tutorial: creazione di un server Web e un cluster database Amazon Aurora

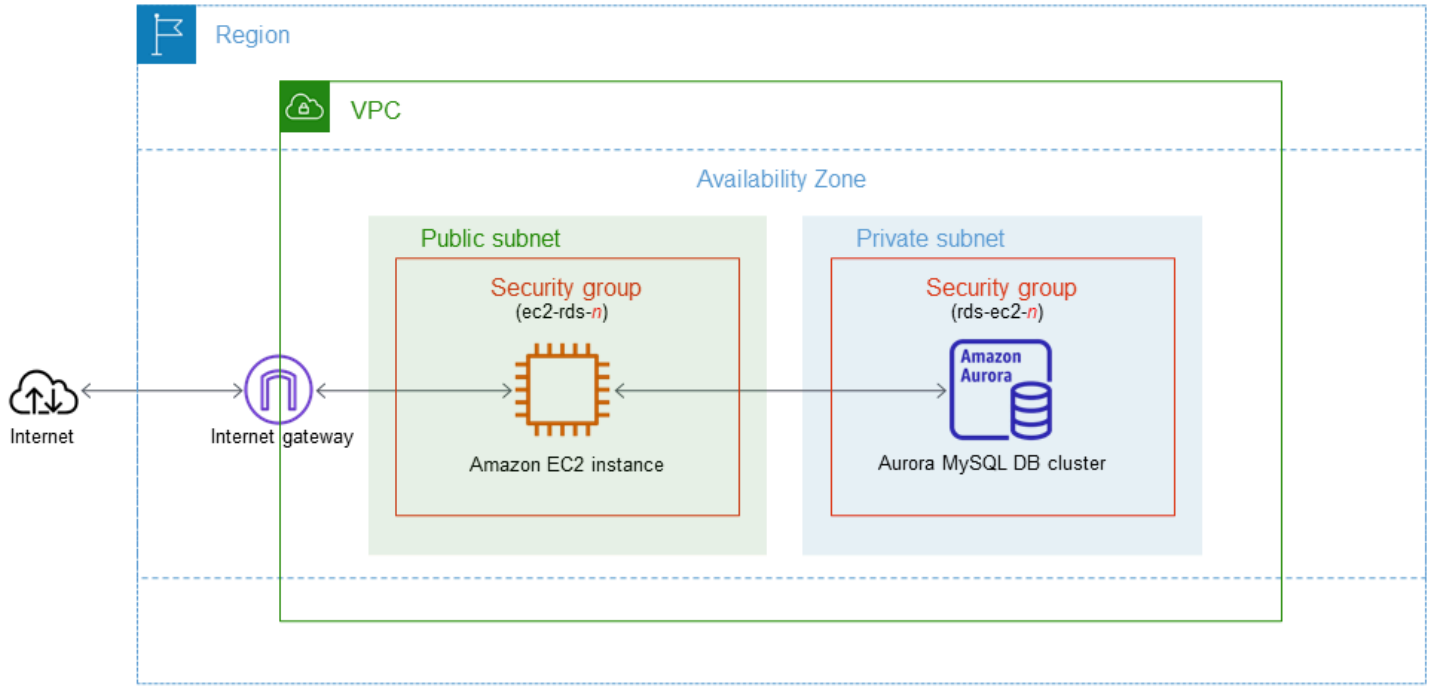
Creazione e connessione a un cluster di database Aurora MySQL

Questo tutorial illustra come creare un'istanza EC2 e un cluster di database Aurora MySQL. Il tutorial mostra come accedere al cluster di database dall'istanza EC2 utilizzando il client MySQL standard. Come best practice, questo tutorial spiega come creare un cluster di database privato in un cloud privato virtuale (VPC). Nella maggior parte dei casi, le risorse presenti nello stesso VPC, come le istanze EC2, possono accedere al cluster di database, mentre le risorse esterne al VPC non possono accedervi.

Dopo aver completato il tutorial, è presente una sottorete pubblica e una privata in ogni zona di disponibilità del VPC. In una zona di disponibilità, l'istanza EC2 si trova nella sottorete pubblica mentre l'istanza database si trova nella sottorete privata.

> **Important**
>
> Non ci sono costi per la creazione di un AWS account. Tuttavia, completando questo tutorial, potresti incorrere in costi per le AWS risorse che utilizzi. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

Il seguente diagramma illustra la configurazione al completamento del tutorial.



Questo tutorial ti consente di creare le tue risorse utilizzando uno dei seguenti metodi:

1. Usa AWS Management Console - Fase 1: creazione di un'istanza EC2 e Fase 2: creazione di un cluster di database Aurora MySQL
2. Utilizzare AWS CloudFormation per creare l'istanza del database e l'istanza EC2 - (Facoltativo) Crea VPC, istanza EC2 e cluster Aurora MySQL utilizzando AWS CloudFormation

Il primo metodo utilizza Easy create per creare un cluster Aurora MySQL DB privato con. AWS Management Console Qui, si specificano solo il tipo di motore DB, la dimensione dell'istanza DB e l'identificatore del cluster DB. Easy create (Creazione rapida) utilizza l'impostazione predefinita per altre opzioni di configurazione.

Quando si utilizza invece Standard create, è possibile specificare più opzioni di configurazione quando si crea un cluster DB. Queste opzioni includono impostazioni per la disponibilità, la sicurezza, i backup e la manutenzione. Per creare un cluster di database pubblico, è necessario utilizzare la Creazione standard. Per informazioni, consulta the section called “Creazione di un cluster di database”.

Argomenti

- Prerequisiti
- Fase 1: creazione di un'istanza EC2
- Fase 2: creazione di un cluster di database Aurora MySQL
- (Facoltativo) Crea VPC, istanza EC2 e cluster Aurora MySQL utilizzando AWS CloudFormation
- Fase 3: connessione a un cluster di database Aurora MySQL
- Fase 4: eliminazione dell'istanza EC2 e del cluster di database
- (Facoltativo) Elimina l'istanza EC2 e il cluster DB creati con CloudFormation
- (Facoltativo) Connessione del cluster database a una funzione Lambda

Prerequisiti

Prima di iniziare, completa le fasi descritte in questa sezione:

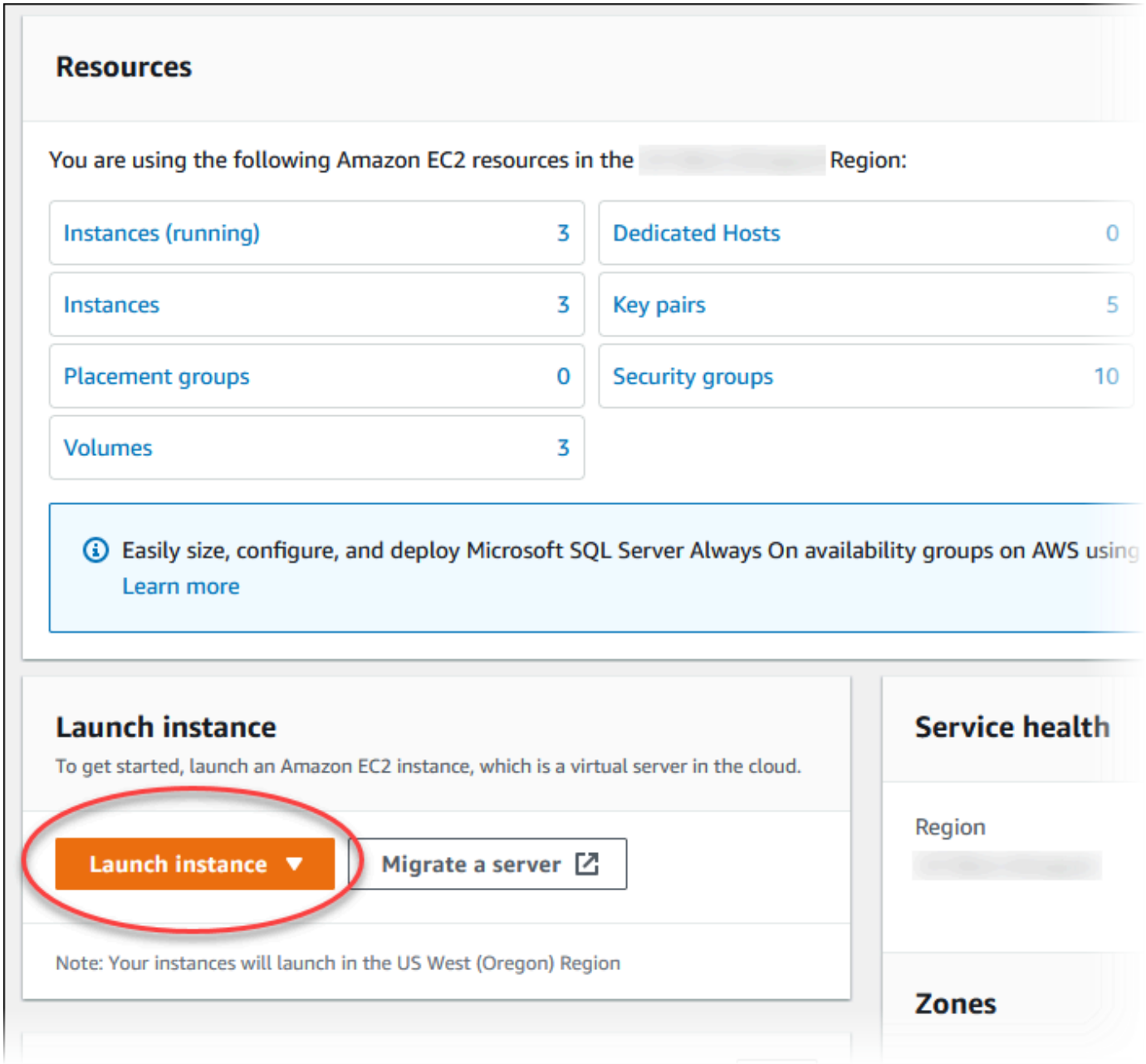
- Registrati per un Account AWS
- Crea un utente con accesso amministrativo

Fase 1: creazione di un'istanza EC2

Crea un'istanza Amazon EC2 da utilizzare per connetterti al database.

Per creare un'istanza EC2

1. Accedi AWS Management Console e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli l'istanza EC2 Regione AWS in cui desideri creare l'istanza EC2.
3. Seleziona Pannello di controllo EC2, quindi Avvia istanza, come visualizzato di seguito.

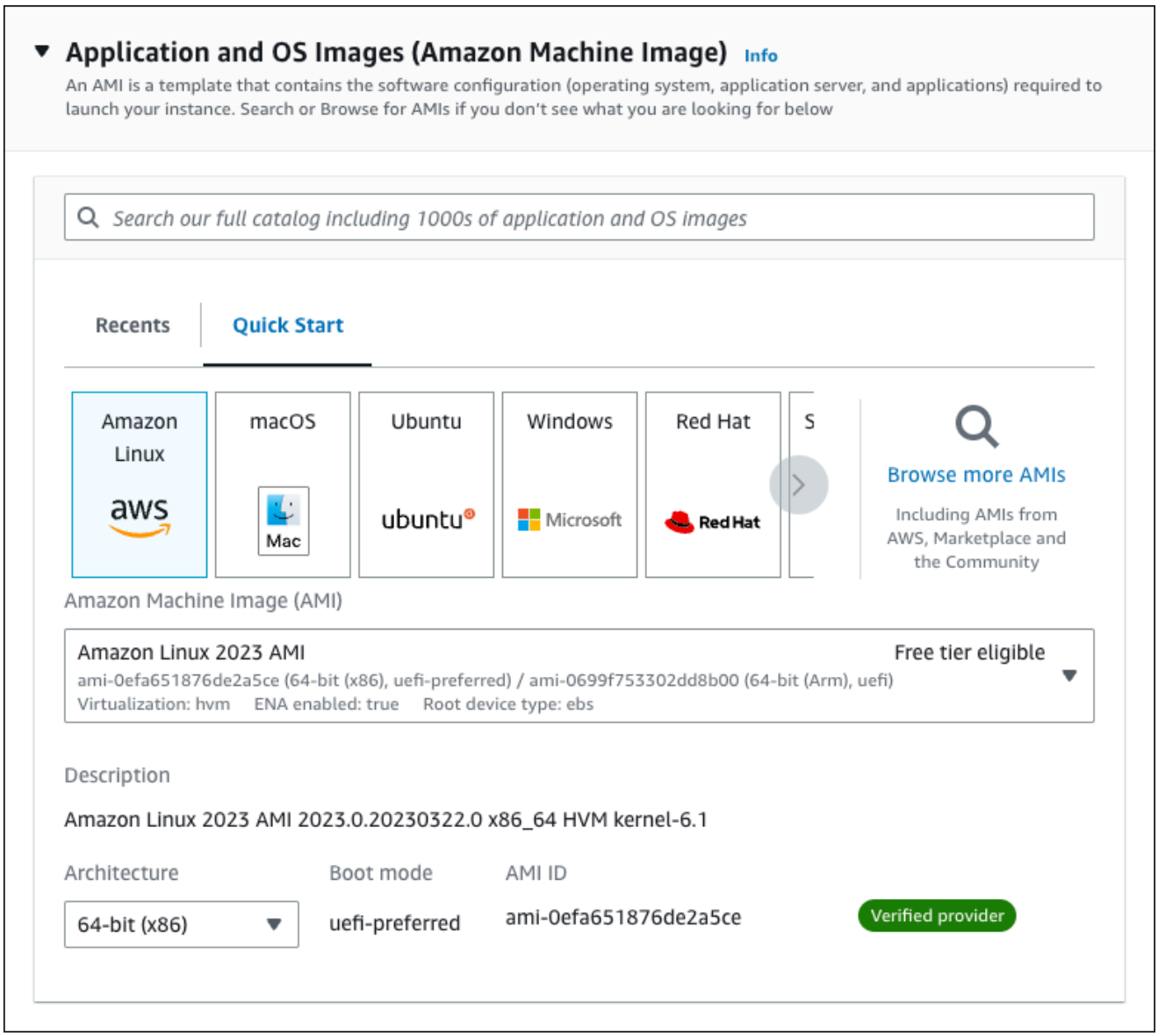


Viene visualizzata la pagina Avvia un'istanza.

4. Scegli le seguenti impostazioni nella pagina Avvia un'istanza.

   a. Nell'area Name and tags (Nome e tag), in Name (Nome) inserisci **`ec2-database-connect`**.

   b. In Immagini applicazione e sistema operativo (Amazon Machine Image), scegli Amazon Linux, quindi AMI Amazon Linux 2023. Mantieni le selezioni predefinite per le altre opzioni.



c. In Instance type (Tipo di istanza), scegli t2.micro.

d. In Key pair (login) (Coppia di chiavi (login), per Key pair name (Nome della coppia di chiavi), scegli una coppia di chiavi esistente. Per creare una nuova coppia di chiavi per l'istanza Amazon EC2, scegli Create new key pair (Crea nuova coppia di chiavi) e quindi utilizza la finestra Create key pair (Crea coppia di chiavi) per crearla.

   Per ulteriori informazioni sulla creazione di una nuova coppia di chiavi, consulta Create a key pair nella Amazon EC2 User Guide.

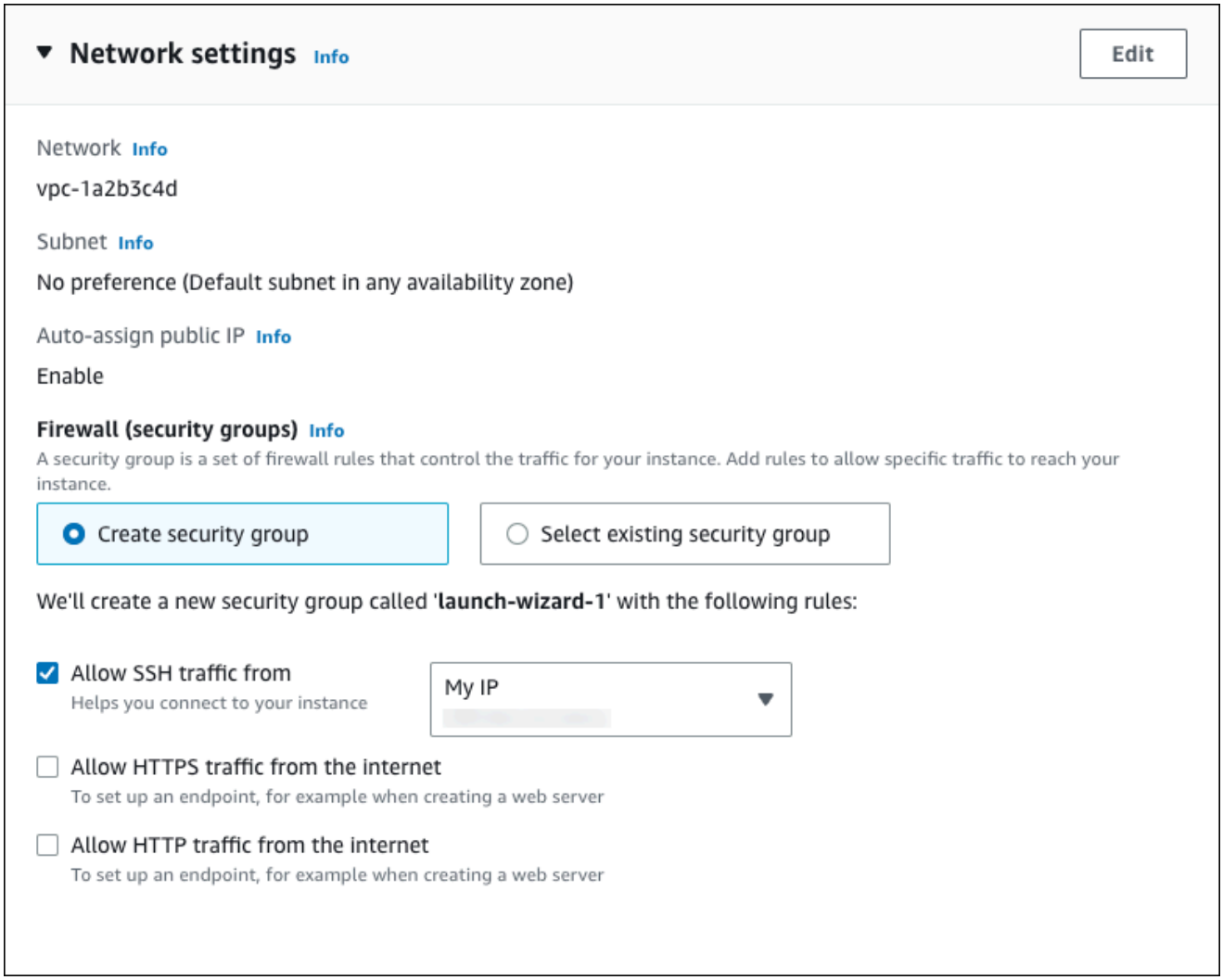
e. In Consenti traffico SSH, nell'area Impostazioni di rete scegliere l'origine delle connessioni SSH all'istanza EC2.

È possibile scegliere My IP (Il mio IP) se l'indirizzo IP visualizzato è corretto per le connessioni SSH. In caso contrario, è possibile determinare l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH). Per determinare l'indirizzo IP pubblico, in una finestra o una scheda del browser diversa, è possibile utilizzare il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è 192.0.2.1/32.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, accertati di determinare l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze EC2 pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze EC2 utilizzando SSH.

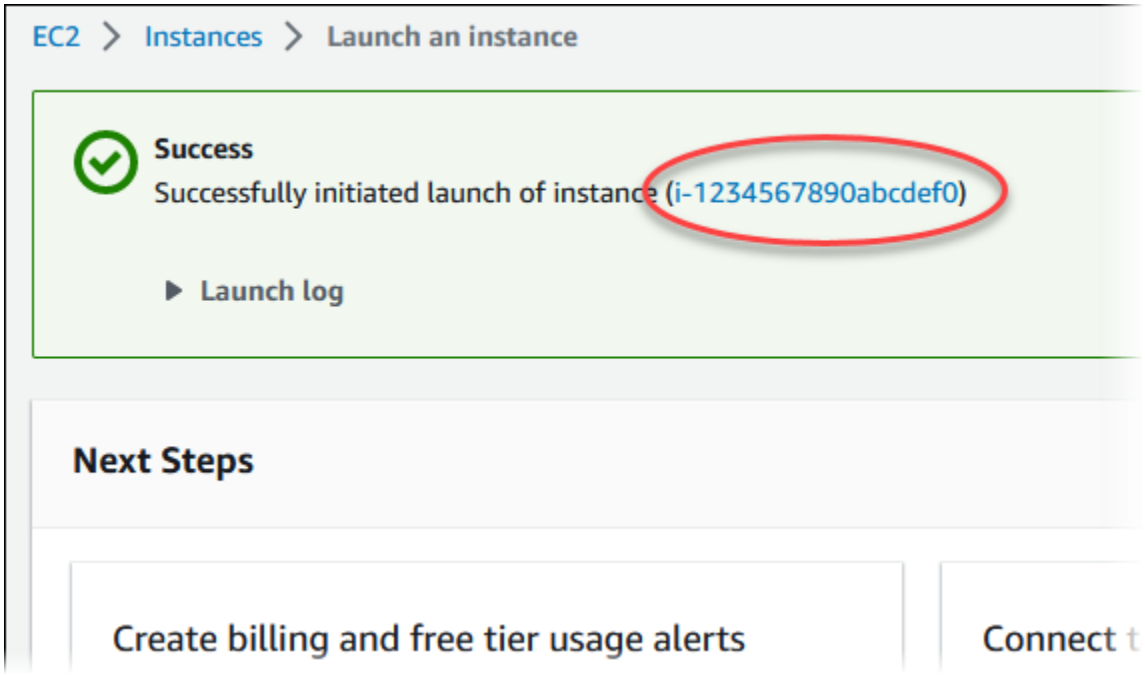
L'immagine seguente mostra un esempio della sezione Impostazioni di rete.



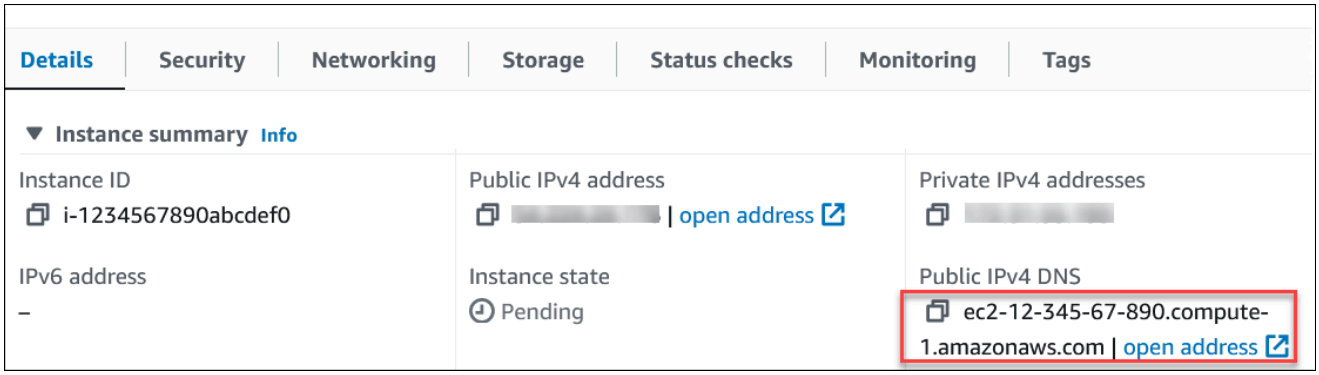
f. Lascia i valori predefiniti per le sezioni rimanenti.

g. Analizza un riepilogo della configurazione dell'istanza EC2 nel pannello Riepilogo e, quando è tutto pronto, scegli Avvia istanza.

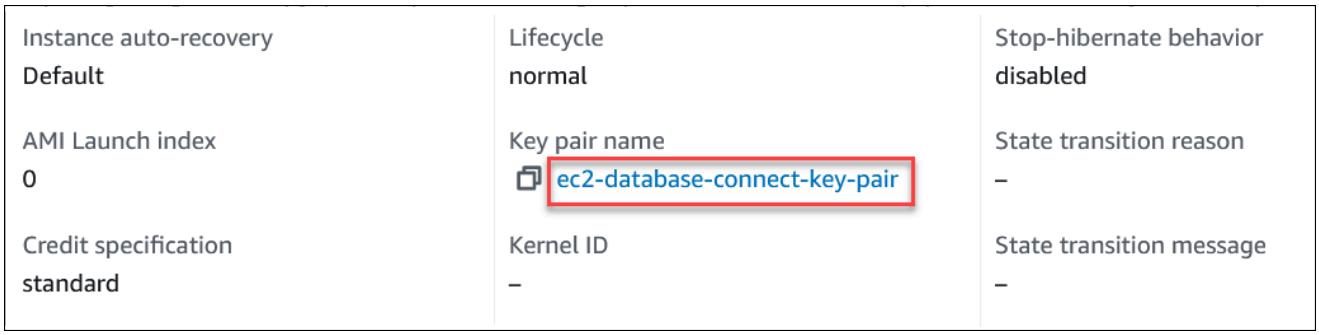
5. Nella pagina Stato avvio prendi nota dell'identificatore per la nuova istanza EC2, ad esempio: `i-1234567890abcdef0`.



6. Scegli l'identificatore dell'istanza EC2 per aprire l'elenco delle istanze EC2, quindi seleziona l'istanza EC2.
7. Nella scheda Dettagli, annota i seguenti valori, necessari quando ti connetti tramite SSH:
   a. In Riepilogo istanza, annota il valore visualizzato in DNS IPv4 pubblico.



   b. In Dettagli istanza, annota il valore visualizzato in Nome coppia di chiavi.



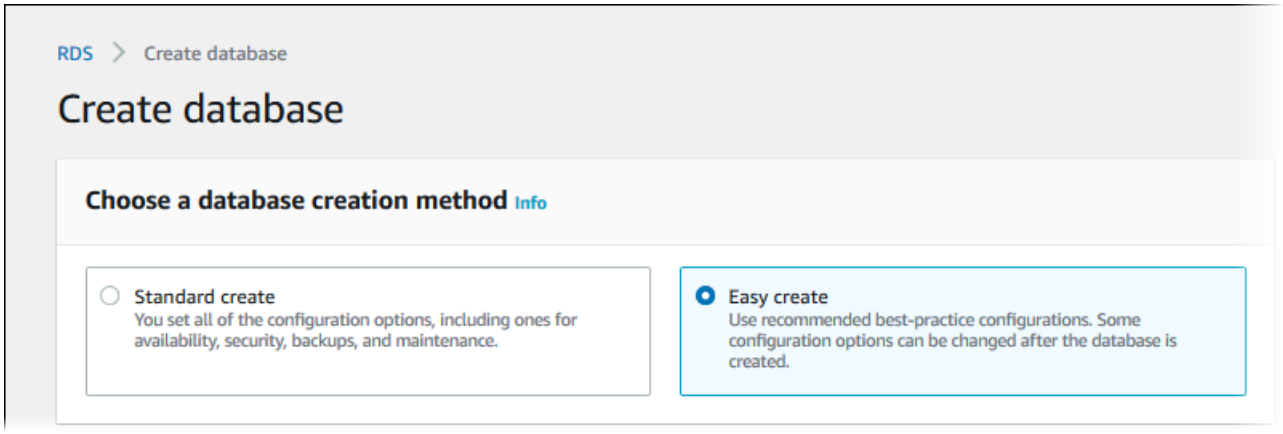
8. Attendi che Stato dell'istanza diventi In esecuzione per l'istanza EC2 prima di continuare.

Fase 2: creazione di un cluster di database Aurora MySQL

In questo esempio, utilizzi la Creazione semplice per creare un cluster di database Aurora MySQL con una classe di istanza database db.r6g.large.

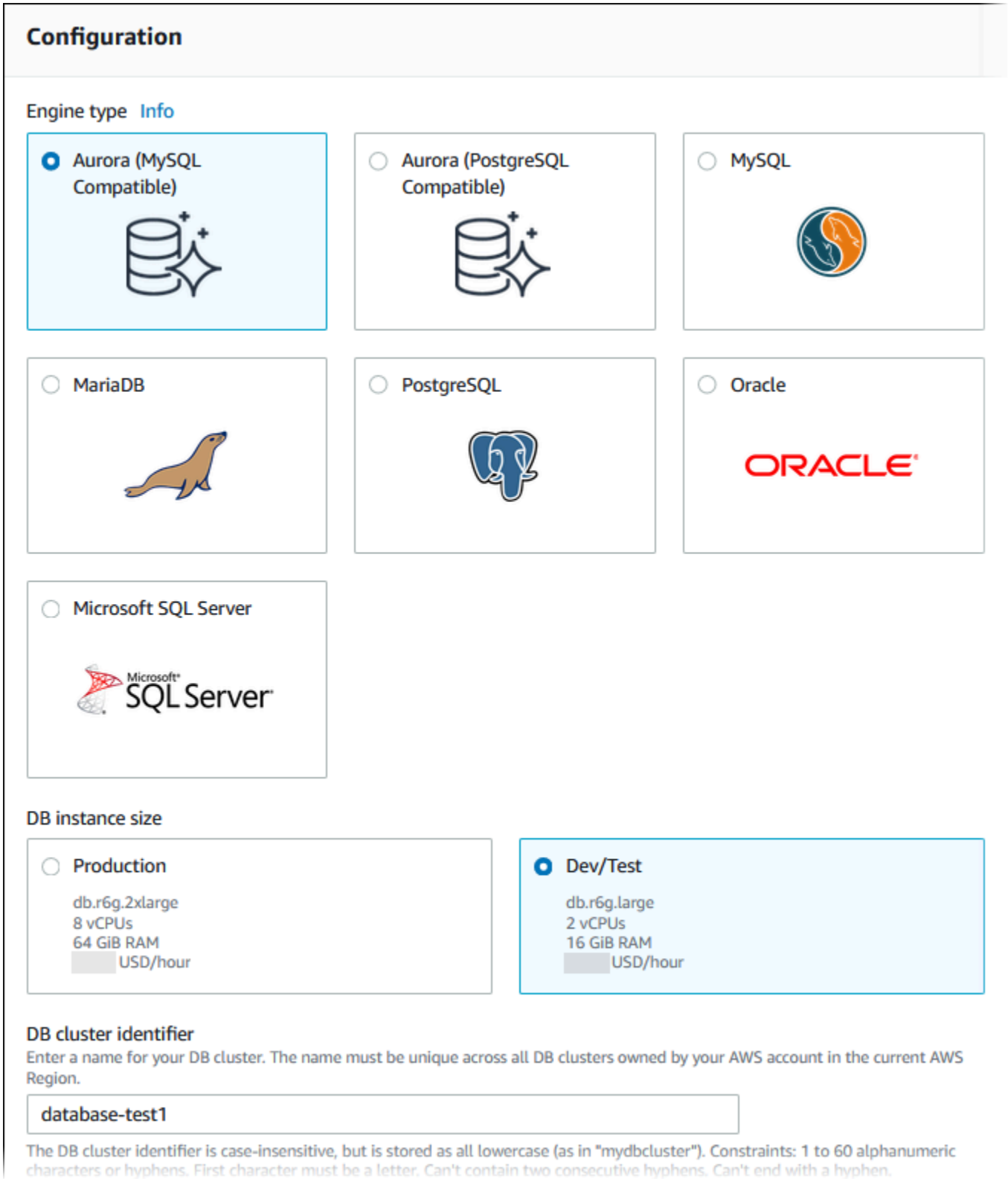
Per creare un cluster di database Aurora MySQL con Creazione semplice

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console Amazon RDS, scegli il cluster di database Regione AWS in cui desideri creare il cluster DB.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database) e verificare che l'opzione Easy Create (Creazione rapida) sia selezionata.



5. In Configurazione scegli Aurora (compatibile con MySQL) per Tipo di motore.
6. Per DB instance size (Dimensione istanza database), seleziona Dev/Test.
7. Per Identificatore cluster di database, immetti **`database-test1`**.

   La pagina Create database (Crea database) la pagina dovrebbe apparire simile alla seguente immagine.

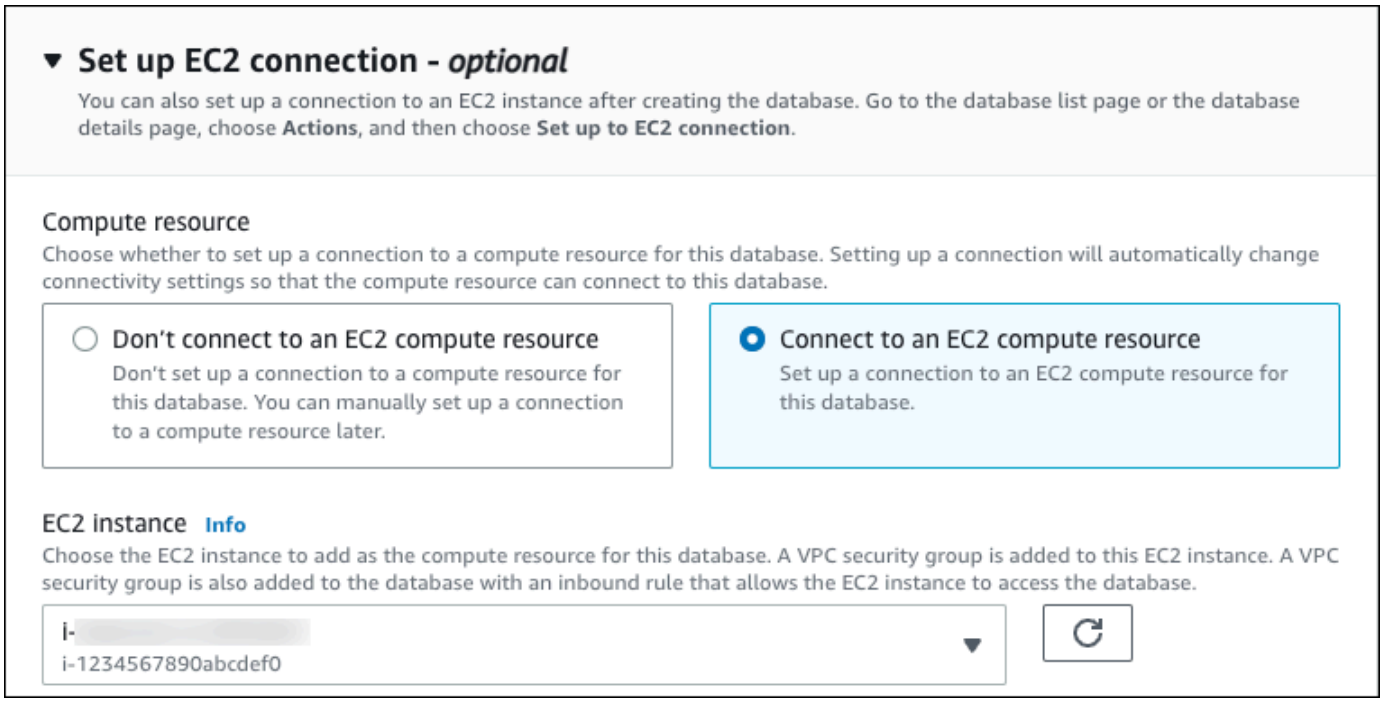


8. Per Nome utente master, inserisci un nome per l'utente master o lascia il nome predefinito.
9. Per utilizzare una password master generata automaticamente per il cluster di database, seleziona Genera automaticamente una password.

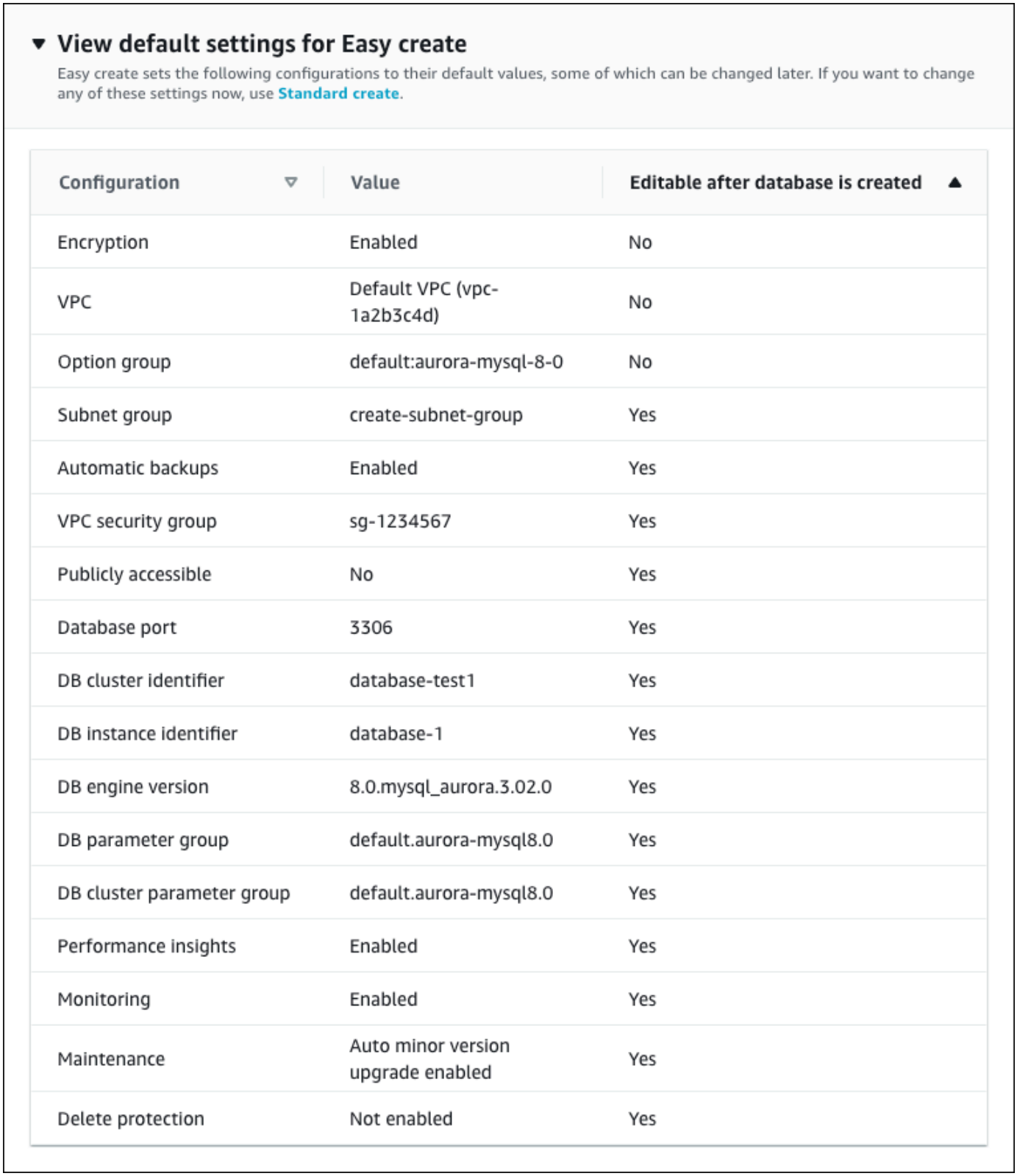
Per inserire la password master, deseleziona la casella Genera automaticamente una password e inserisci la stessa password in Password master e Conferma password.

10. Per configurare una connessione con l'istanza EC2 creata in precedenza, apri Configura connessione EC2 - opzionale.

    Seleziona Connetti a una risorsa di calcolo EC2. Scegli l'istanza EC2 creata in precedenza.



11. Apri Visualizza le impostazioni predefinite per la creazione Semplice.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-mysql-8-0	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	8.0.mysql_aurora.3.02.0	Yes
DB parameter group	default.aurora-mysql8.0	Yes
DB cluster parameter group	default.aurora-mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Puoi esaminare le impostazioni predefinite utilizzate con Easy create (Creazione rapida). La colonna Modificabile dopo la creazione del database mostra le opzioni che puoi modificare dopo aver creato il database.

- Se un'impostazione contiene No in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database.
- Se un'impostazione contiene Sì in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database o modificare il cluster di database dopo averlo creato per cambiare l'impostazione.

12. Scegliere Crea database.

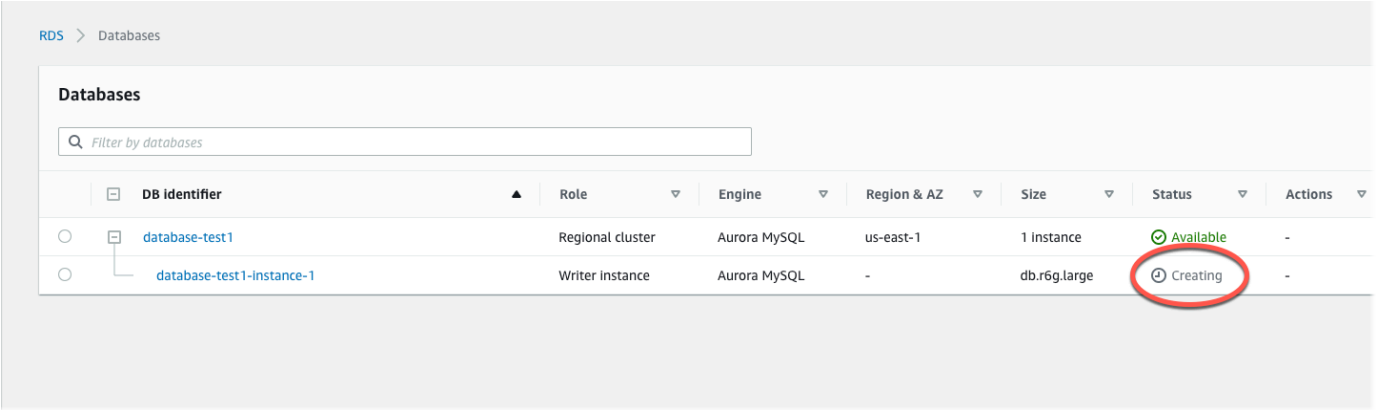
    Per vedere nome utente e password per il cluster di database, seleziona Visualizza i dettagli delle credenziali.

    Per connetterti al cluster di database come utente master, utilizza il nome utente e la password visualizzati.

    > **Important**
    >
    > Non potrai visualizzare di nuovo la password dell'utente principale. Se non la registri, potresti doverla modificare.
    > Se devi modificare la password dell'utente principale dopo che il cluster database è disponibile, puoi modificare il cluster database per eseguire tale operazione. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

13. Nell'elenco Database seleziona il nome del nuovo cluster di database Aurora MySQL per visualizzarne i dettagli.

    L'istanza di scrittura ha lo stato Creazione in corso fino a quando il cluster di database non è pronto per essere utilizzato.



Quando lo stato dell'istanza di scrittura cambia in Disponibile, puoi connetterti al cluster di database. A seconda della classe di istanza database e della quantità di storage, prima che il nuovo cluster di database sia disponibile possono trascorrere fino a 20 minuti.

(Facoltativo) Crea VPC, istanza EC2 e cluster Aurora MySQL utilizzando AWS CloudFormation

Invece di utilizzare la console per creare il VPC, l'istanza EC2 e il cluster Aurora MySQL DB, puoi AWS CloudFormation utilizzarla per fornire risorse trattando l'infrastruttura come codice. AWS Per aiutarti a organizzare AWS le tue risorse in unità più piccole e più gestibili, puoi utilizzare la funzionalità nested stack. AWS CloudFormation Per ulteriori informazioni, consulta Creare uno stack sulla AWS CloudFormation console e Lavorare con gli stack annidati.

> **Important**
>
> AWS CloudFormation è gratuito, ma le risorse che CloudFormation crea sono attive. Ti verranno addebitati i costi di utilizzo standard per queste risorse fino alla loro cessazione. L'addebito totale sarà minimo. Per informazioni su come ridurre al minimo gli addebiti, vai al AWS piano gratuito.

Per creare le tue risorse utilizzando la AWS CloudFormation console, completa i seguenti passaggi:

- Passaggio 1: scarica il CloudFormation modello
- Passaggio 2: configura le tue risorse utilizzando CloudFormation

Scarica il CloudFormation modello

Un CloudFormation modello è un file di testo JSON o YAML che contiene le informazioni di configurazione sulle risorse che desideri creare nello stack. Questo modello crea anche un VPC e un bastion host per te insieme al cluster Aurora.

Per scaricare il file modello, apri il seguente link, Modello Aurora MySQL. CloudFormation

Nella pagina Github, fai clic sul pulsante Scarica file raw per salvare il file YAML del modello.

Configura le tue risorse usando CloudFormation

> **Note**
>
> Prima di iniziare questo processo, assicurati di avere una coppia di chiavi per un'istanza EC2 nel tuo Account AWS. Per ulteriori informazioni, consulta Coppie di chiavi Amazon EC2 e istanze Linux.

Quando utilizzi il AWS CloudFormation modello, devi selezionare i parametri corretti per assicurarti che le risorse vengano create correttamente. Segui la procedura riportata di seguito:

1. Accedi AWS Management Console e apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegli Crea stack.
3. Nella sezione Specificare il modello, seleziona Carica un file modello dal computer, quindi scegli Avanti.
4. Nella pagina Specificare i dettagli dello stack, imposta i seguenti parametri:
   a. Imposta il nome dello stack su AurMy SQL. TestStack
   b. In Parametri, imposta le zone di disponibilità selezionando due zone di disponibilità.
   c. Nella configurazione Linux Bastion Host, in Key Name, seleziona una coppia di chiavi per accedere alla tua istanza EC2.
   d. Nelle impostazioni di configurazione di Linux Bastion Host, imposta l'intervallo IP consentito sul tuo indirizzo IP. Per connetterti alle istanze EC2 nel tuo VPC utilizzando Secure Shell (SSH), determina il tuo indirizzo IP pubblico utilizzando il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è 192.0.2.1/32.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze EC2 pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze EC2 utilizzando SSH.

e. Nella configurazione generale del database, imposta la classe dell'istanza del database su db.r6g.large.

f. Imposta il nome del database su. **`database-test1`**

g. Per Nome utente principale del database, inserisci un nome per l'utente principale.

h. Imposta la password utente principale di Manage DB con Secrets Manager su `false` per questo tutorial.

i. Per la password del database, imposta una password a tua scelta. Ricorda questa password per ulteriori passaggi del tutorial.

j. Imposta la distribuzione Multi-AZ su`false`.

k. Lascia tutte le altre impostazioni come valori predefiniti. Fate clic su Avanti per continuare.

5. Nella pagina Configura le opzioni dello stack, lascia tutte le opzioni predefinite. Fai clic su Avanti per continuare.
6. Nella pagina Review stack, seleziona Invia dopo aver verificato le opzioni del database e dell'host Linux bastion.

Una volta completato il processo di creazione dello stack, visualizza gli stack con nomi BastionStacke AMSNS per annotare le informazioni necessarie per connetterti al database. Per ulteriori informazioni, consulta Visualizzazione dei dati e delle risorse AWS CloudFormation dello stack su. AWS Management Console

Fase 3: connessione a un cluster di database Aurora MySQL

È possibile utilizzare qualsiasi applicazione client SQL standard per la connessione al cluster di database. In questo esempio, ti connetti a un cluster di database Aurora MySQL utilizzando il client della linea di comando mysql.

Per connettersi al cluster di database Aurora MySQL

1. Individua l'endpoint (nome DNS) e il numero di porta per l'istanza di scrittura del cluster di database.

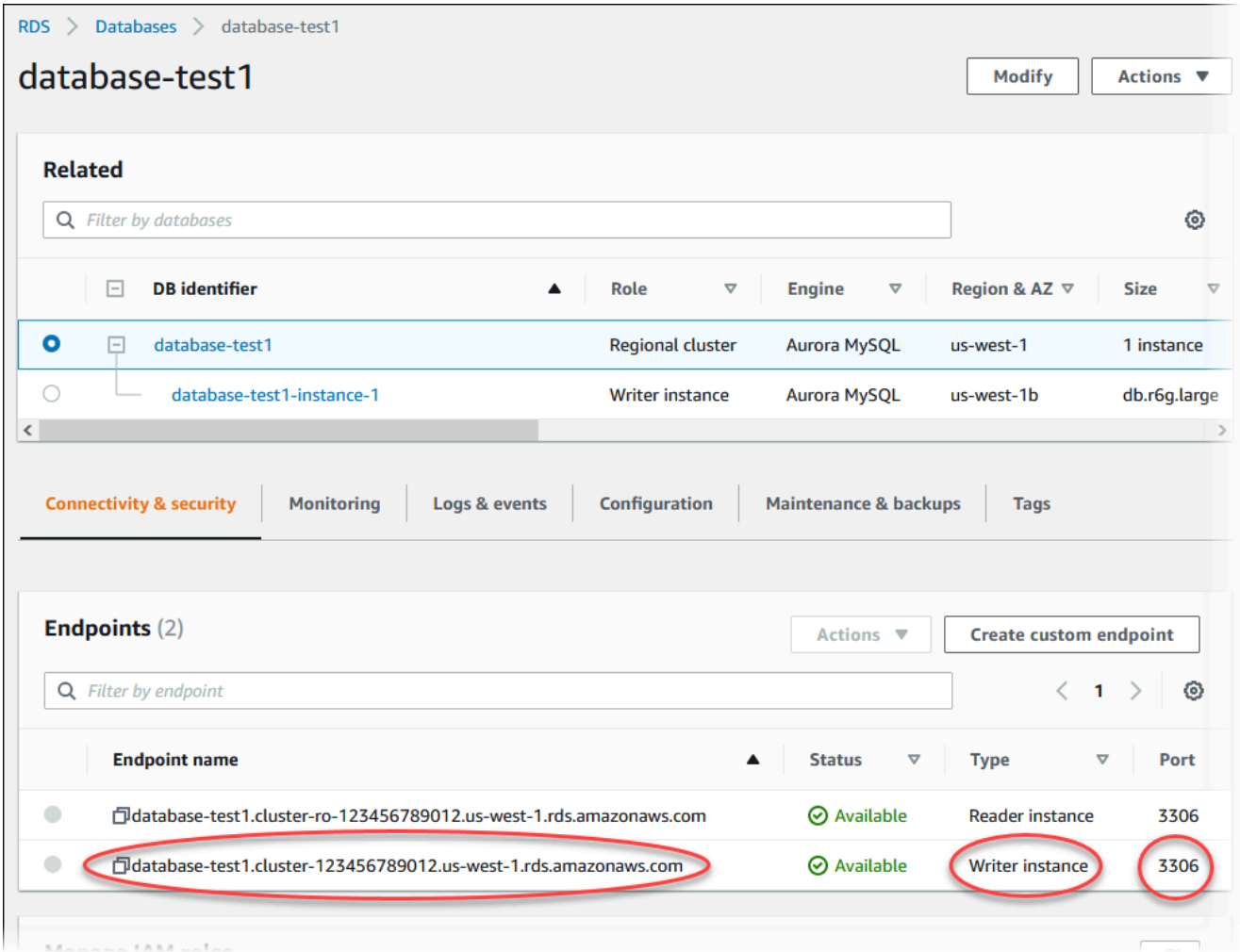
   a. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   b. Nell'angolo superiore destro della console Amazon RDS, scegli la Regione AWS del cluster di database.

   c. Nel riquadro di navigazione, scegli Databases (Database).

   d. Seleziona il nome di un cluster di database Aurora MySQL per visualizzarne i dettagli.

   e. Nella scheda Connettività e sicurezza, copia l'endpoint dell'istanza di scrittura. Annotare anche il numero di porta. L'endpoint e il numero di porta sono necessari per la connessione al cluster di database.



2. Connettiti all'istanza EC2 che hai creato in precedenza seguendo i passaggi in Connetti alla tua istanza Linux nella Amazon EC2 User Guide.

   Ti consigliamo di connetterti all'istanza EC2 tramite SSH. Se l'utilità client SSH è installata su Windows, Linux o Mac, puoi connetterti all'istanza utilizzando il comando nel seguente formato:

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Ad esempio, supponi che `ec2-database-connect-key-pair.pem` sia archiviato in `/dir1` su Linux e che il DNS IPv4 pubblico per l'istanza EC2 sia `ec2-12-345-678-90.compute-1.amazonaws.com`. Quindi, il comando SSH sarà simile al seguente:

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. Ottieni le ultime correzioni di bug e gli aggiornamenti di sicurezza aggiornando il software sulla tua istanza EC2. A tale scopo, utilizzare il comando seguente.

   > **Note**
   >
   > L'opzione `-y` installa gli aggiornamenti senza chiedere conferma. Per esaminare gli aggiornamenti prima di installarli, omettere questa opzione.

   ```
   sudo dnf update -y
   ```

4. Per installare il client della linea di comando mysql da MariaDB su Amazon Linux 2023, esegui il comando seguente:

   ```
   sudo dnf install mariadb105
   ```

5. Connessione a un cluster di database Aurora MySQL. Ad esempio, immetti il comando seguente: Questa azione consente di connetterti al cluster di database Aurora MySQL utilizzando il client MySQL.

   Sostituisci l'endpoint dell'istanza di scrittura per *`endpoint`* e sostituisci il nome utente master utilizzato per *`admin`*. Devi fornire la password master utilizzata quando viene richiesta una password.

```
mysql -h endpoint -P 3306 -u admin -p
```

Dopo aver immesso la password per l'utente, l'output dovrebbe essere analogo a quanto mostrato di seguito.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 217
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Per ulteriori informazioni sulla connessione a un cluster di database Aurora MySQL, consulta Connessione a un cluster di database Amazon Aurora MySQL. In caso di mancata connessione al cluster database, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Per motivi di sicurezza, la best practice è utilizzare connessioni crittografate. Utilizzare una connessione MySQL non crittografata solo quando il client e il server sono nello stesso VPC e la rete è attendibile. Per ulteriori informazioni sull'uso di connessioni crittografate, consulta Connessione ad Aurora MySQL tramite SSL.

6. Eseguire comandi SQL.

   Ad esempio, il seguente comando SQL mostra la data e l'ora correnti:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Fase 4: eliminazione dell'istanza EC2 e del cluster di database

Dopo la connessione e l'esplorazione dell'istanza EC2 e del cluster di database di esempio che hai creato, eliminale per evitare di ricevere l'addebito dei relativi costi.

Se in passato AWS CloudFormation creavi risorse, salta questo passaggio e vai al passaggio successivo.

Per eliminare l'istanza EC2

1. Accedi AWS Management Console e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel riquadro di navigazione, seleziona Istanze.
3. Seleziona l'istanza EC2 e scegli Stato istanza, Termina istanza.
4. Quando viene richiesta la conferma, seleziona Terminate (Interrompi).

Per ulteriori informazioni sull'eliminazione di un'istanza EC2, consulta Terminate your instance nella Amazon EC2 User Guide.

Per eliminare un cluster di database

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database), quindi selezionare l'istanza database associata con il cluster di database.
3. In Actions (Azioni), selezionare Delete (Elimina).
4. Deseleziona Creare uno snapshot finale?
5. Completa la conferma e scegli Elimina.

Dopo aver eliminato tutte le istanze database associate a un cluster di database, il cluster di database viene eliminato automaticamente.

(Facoltativo) Elimina l'istanza EC2 e il cluster DB creati con CloudFormation

Se prima creavi AWS CloudFormation risorse, elimina lo CloudFormation stack dopo esserti connesso ed esplorato l'istanza EC2 di esempio e il cluster DB, in modo che non ti vengano più addebitati costi.

Per eliminare le risorse CloudFormation

1. Apri la AWS CloudFormation console.
2. Nella pagina Stacks della CloudFormation console, seleziona lo stack principale (lo stack senza il nome VPCStack o AMSNS). BastionStack
3. Scegli Elimina.

4. Seleziona Elimina stack quando viene richiesta la conferma.

Per ulteriori informazioni sull'eliminazione di uno stack in CloudFormation, consulta Eliminazione di uno stack sulla console nella Guida per l' AWS CloudFormationutente.AWS CloudFormation

(Facoltativo) Connessione del cluster database a una funzione Lambda

È anche possibile connettere il cluster database Aurora MySQL a una risorsa di calcolo serverless Lambda. Le funzioni Lambda consentono di eseguire il codice senza il provisioning o la gestione dell'infrastruttura. Una funzione Lambda consente inoltre di rispondere automaticamente alle richieste di esecuzione del codice su qualsiasi scala, da una dozzina di eventi al giorno a centinaia al secondo. Per ulteriori informazioni, consulta Connessione automatica di una funzione Lambda e di un cluster database Aurora.

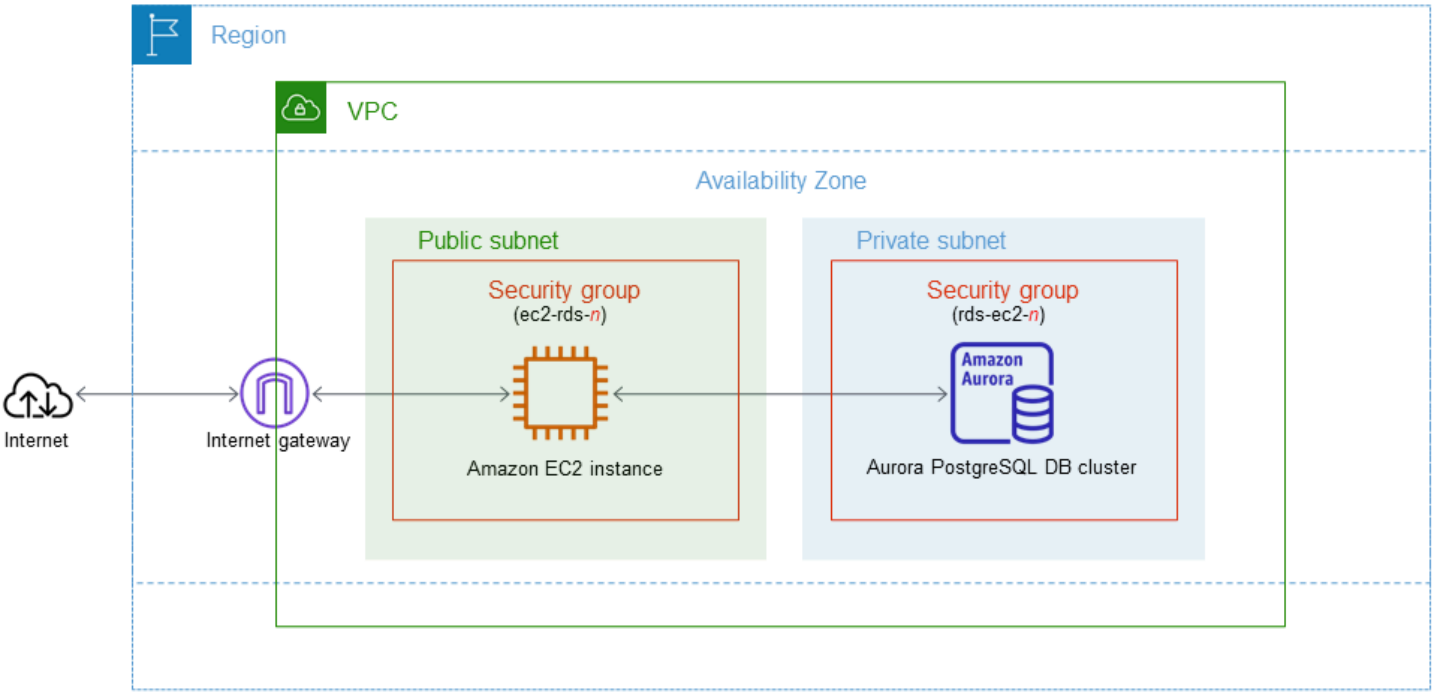
Creazione e connessione di un cluster di database Aurora PostgreSQL

Questo tutorial illustra come creare un'istanza EC2 e un cluster di database Aurora PostgreSQL. Il tutorial mostra come accedere al cluster di database dall'istanza EC2 utilizzando il client PostgreSQL standard. Come best practice, questo tutorial spiega come creare un cluster di database privato in un cloud privato virtuale (VPC). Nella maggior parte dei casi, le risorse presenti nello stesso VPC, come le istanze EC2, possono accedere al cluster di database, mentre le risorse esterne al VPC non possono accedervi.

Dopo aver completato il tutorial, è presente una sottorete pubblica e una privata in ogni zona di disponibilità del VPC. In una zona di disponibilità, l'istanza EC2 si trova nella sottorete pubblica mentre l'istanza database si trova nella sottorete privata.

> **Important**
>
> Non ci sono costi per la creazione di un AWS account. Tuttavia, completando questo tutorial, potresti incorrere in costi per le AWS risorse che utilizzi. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

Il seguente diagramma illustra la configurazione al completamento del tutorial.



Questo tutorial ti consente di creare le tue risorse utilizzando uno dei seguenti metodi:

1. Usa AWS Management Console - Fase 1: creazione di un'istanza EC2 e Fase 2: creazione di un cluster di database Aurora PostgreSQL
2. Utilizzare AWS CloudFormation per creare l'istanza del database e l'istanza EC2 - (Facoltativo) Crea VPC, istanza EC2 e cluster Aurora PostgreSQL utilizzando AWS CloudFormation

Il primo metodo utilizza Easy create per creare un cluster Aurora PostgreSQL DB privato con. AWS Management Console Qui, si specificano solo il tipo di motore DB, la dimensione dell'istanza DB e l'identificatore del cluster DB. Easy create (Creazione rapida) utilizza l'impostazione predefinita per altre opzioni di configurazione.

Quando si utilizza invece Standard create, è possibile specificare più opzioni di configurazione quando si crea un cluster DB. Queste opzioni includono impostazioni per la disponibilità, la sicurezza, i backup e la manutenzione. Per creare un cluster di database pubblico, è necessario utilizzare la Creazione standard. Per informazioni, consulta the section called “Creazione di un cluster di database”.

Argomenti

- Prerequisiti
- Fase 1: creazione di un'istanza EC2

- Fase 2: creazione di un cluster di database Aurora PostgreSQL
- (Facoltativo) Crea VPC, istanza EC2 e cluster Aurora PostgreSQL utilizzando AWS CloudFormation
- Fase 3: connessione a un cluster di database Aurora PostgreSQL
- Fase 4: eliminazione dell'istanza EC2 e del cluster di database
- (Facoltativo) Elimina l'istanza EC2 e il cluster DB creati con CloudFormation
- (Facoltativo) Connessione del cluster database a una funzione Lambda

Prerequisiti

Prima di iniziare, completa le fasi descritte in questa sezione:

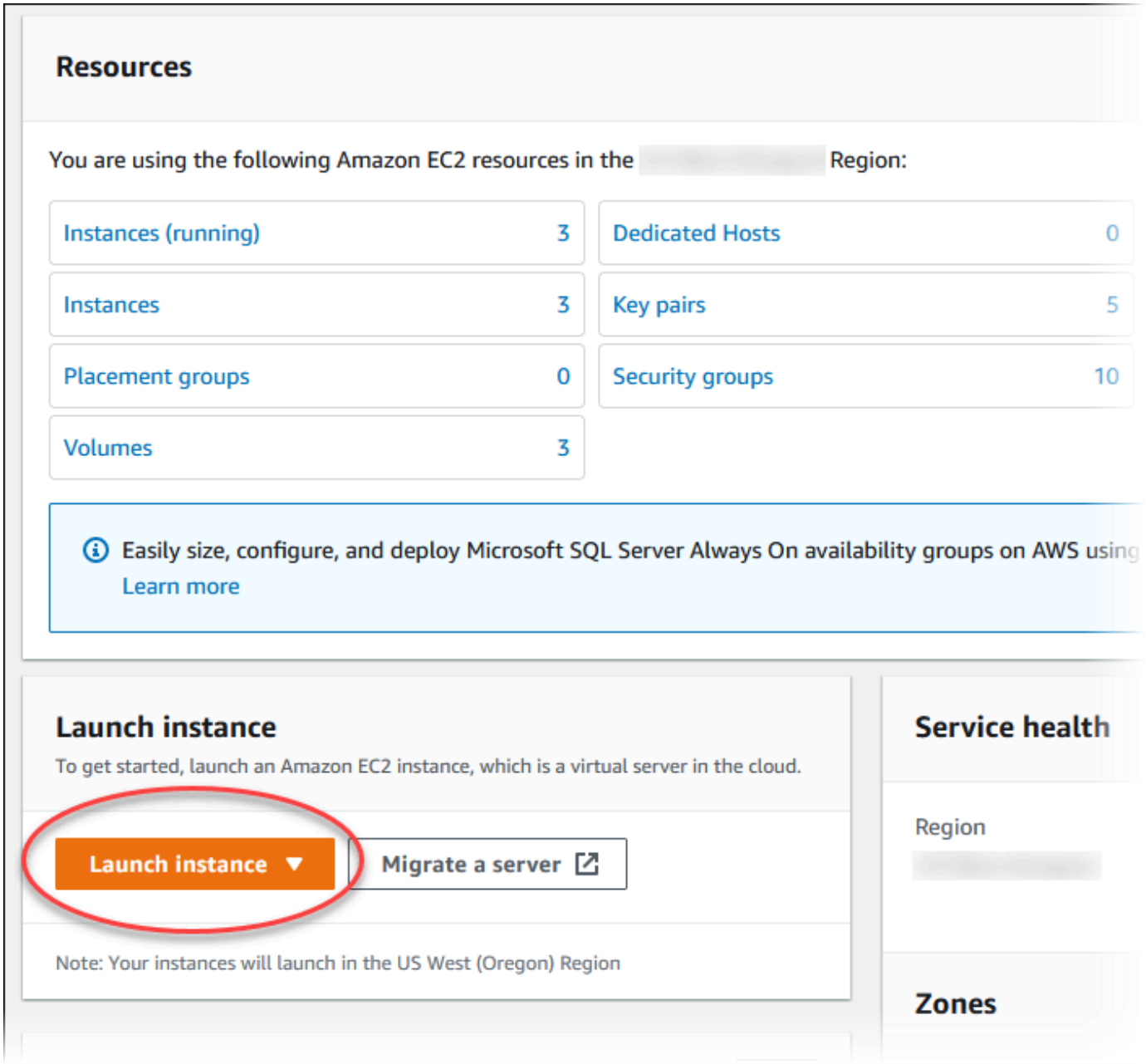
- Registrati per un Account AWS
- Crea un utente con accesso amministrativo

Fase 1: creazione di un'istanza EC2

Crea un'istanza Amazon EC2 da utilizzare per connetterti al database.

Per creare un'istanza EC2

1. Accedi AWS Management Console e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli l'istanza EC2 Regione AWS in cui desideri creare l'istanza EC2.
3. Seleziona Pannello di controllo EC2, quindi Avvia istanza, come visualizzato di seguito.

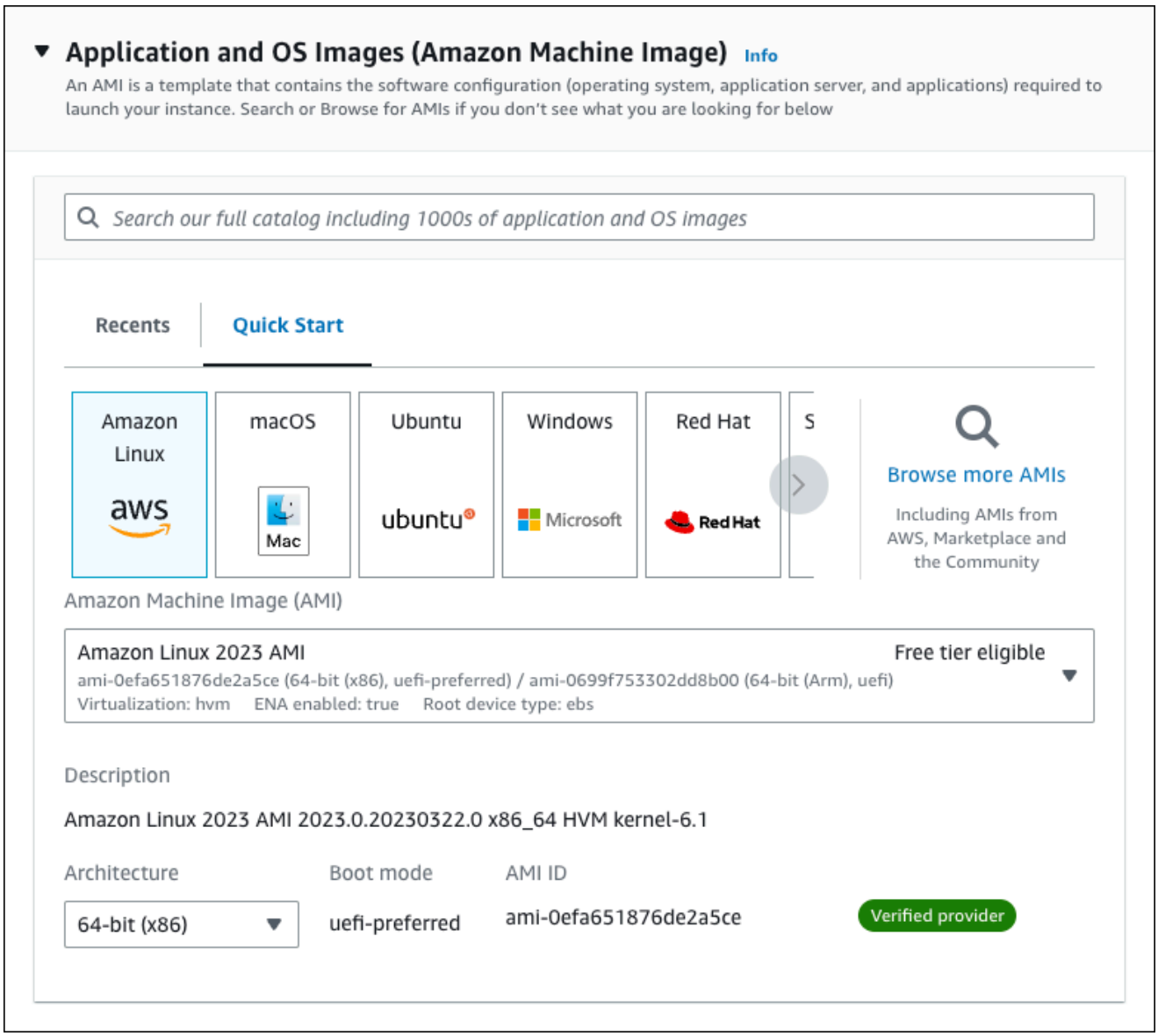


Viene visualizzata la pagina Avvia un'istanza.

4. Scegli le seguenti impostazioni nella pagina Avvia un'istanza.

   a. Nell'area Name and tags (Nome e tag), in Name (Nome) inserisci **`ec2-database-connect`**.

   b. In Immagini applicazione e sistema operativo (Amazon Machine Image), scegli Amazon Linux, quindi AMI Amazon Linux 2023. Mantieni le selezioni predefinite per le altre opzioni.



c. In Instance type (Tipo di istanza), scegli t2.micro.

d. In Key pair (login) (Coppia di chiavi (login), per Key pair name (Nome della coppia di chiavi), scegli una coppia di chiavi esistente. Per creare una nuova coppia di chiavi per l'istanza Amazon EC2, scegli Create new key pair (Crea nuova coppia di chiavi) e quindi utilizza la finestra Create key pair (Crea coppia di chiavi) per crearla.

   Per ulteriori informazioni sulla creazione di una nuova coppia di chiavi, consulta Create a key pair nella Amazon EC2 User Guide.

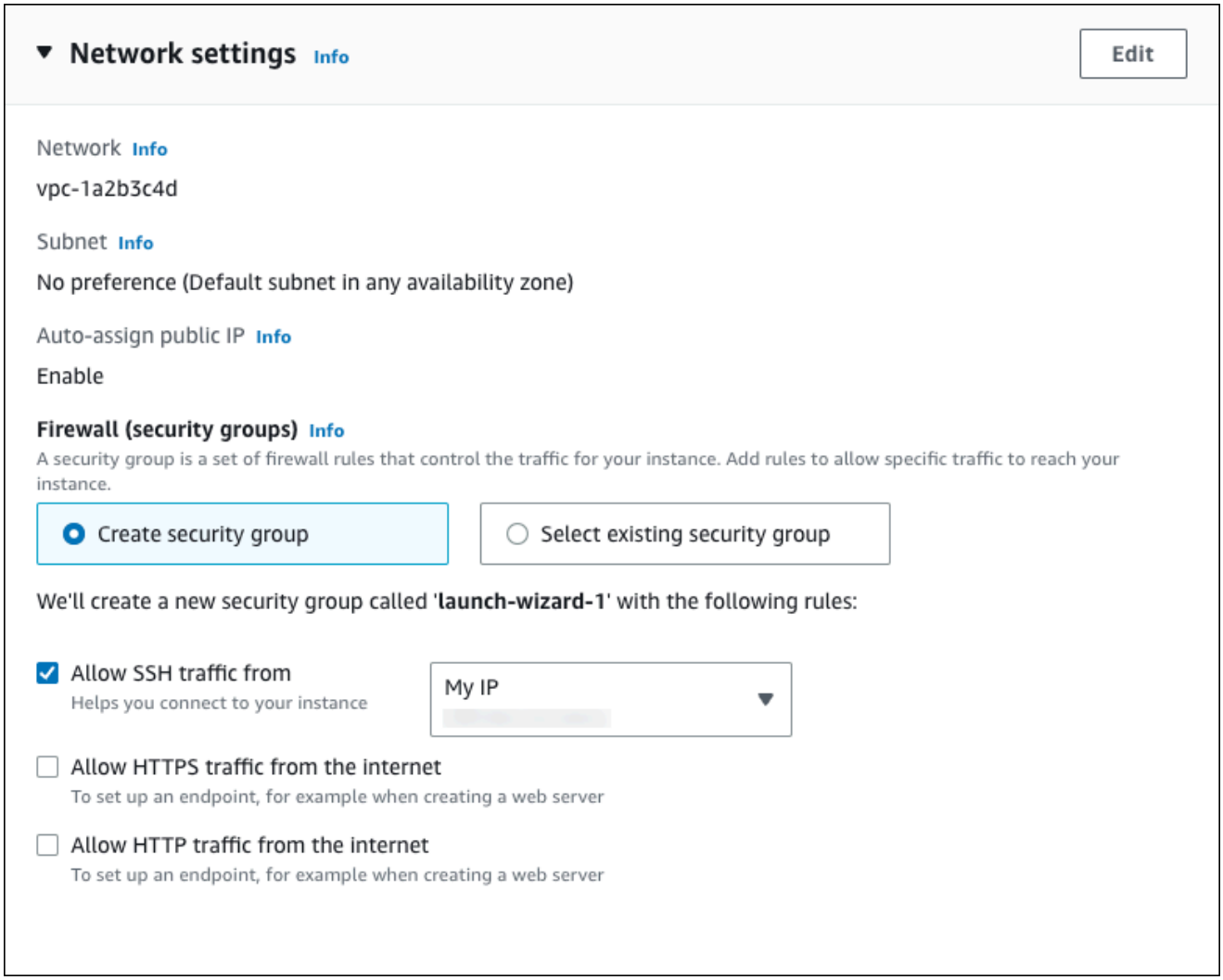
e. In Consenti traffico SSH, nell'area Impostazioni di rete scegliere l'origine delle connessioni SSH all'istanza EC2.

È possibile scegliere My IP (Il mio IP) se l'indirizzo IP visualizzato è corretto per le connessioni SSH. In caso contrario, è possibile determinare l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH). Per determinare l'indirizzo IP pubblico, in una finestra o una scheda del browser diversa, è possibile utilizzare il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è 192.0.2.1/32.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, accertati di determinare l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze EC2 pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze EC2 utilizzando SSH.

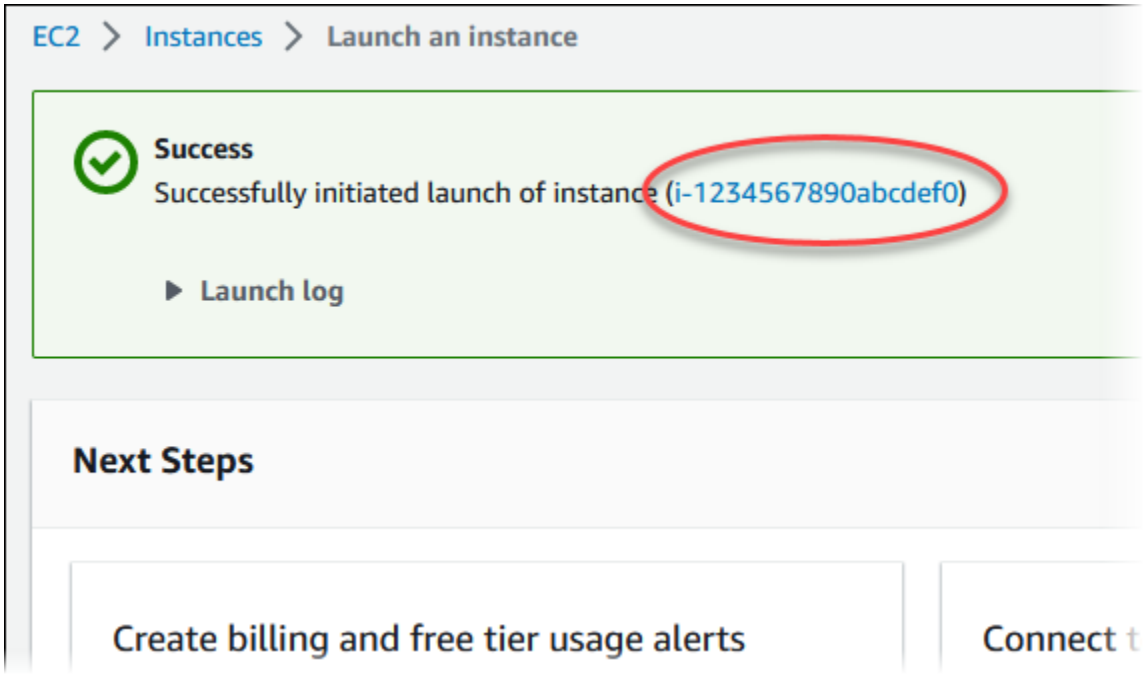
L'immagine seguente mostra un esempio della sezione Impostazioni di rete.



f. Lascia i valori predefiniti per le sezioni rimanenti.

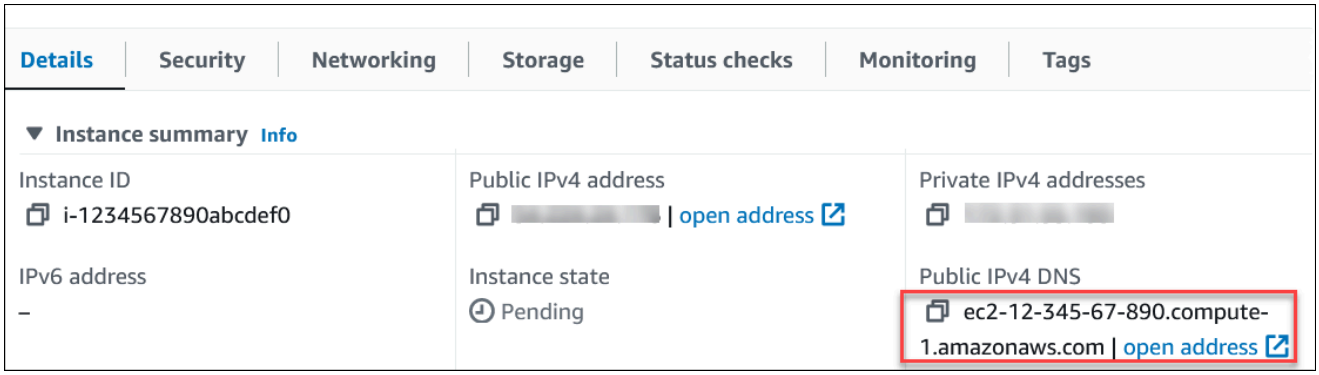
g. Analizza un riepilogo della configurazione dell'istanza EC2 nel pannello Riepilogo e, quando è tutto pronto, scegli Avvia istanza.

5. Nella pagina Stato avvio prendi nota dell'identificatore per la nuova istanza EC2, ad esempio: `i-1234567890abcdef0`.

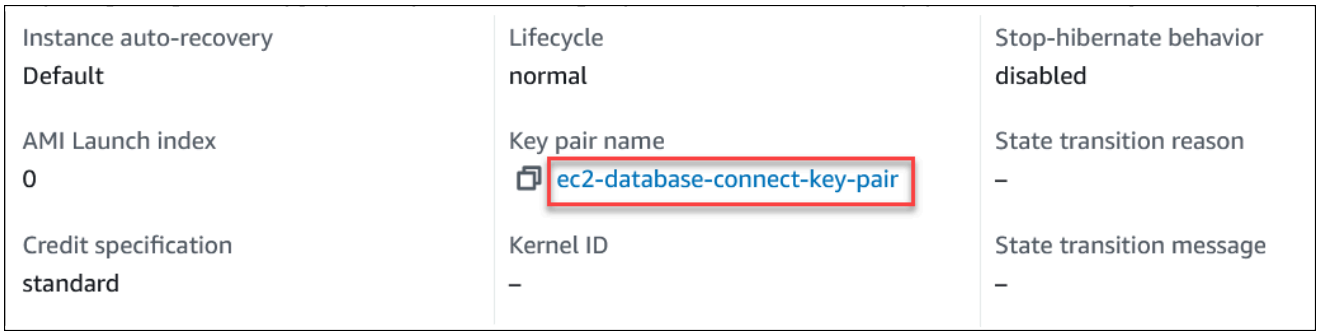


6. Scegli l'identificatore dell'istanza EC2 per aprire l'elenco delle istanze EC2, quindi seleziona l'istanza EC2.
7. Nella scheda Dettagli, annota i seguenti valori, necessari quando ti connetti tramite SSH:

   a. In Riepilogo istanza, annota il valore visualizzato in DNS IPv4 pubblico.



   b. In Dettagli istanza, annota il valore visualizzato in Nome coppia di chiavi.



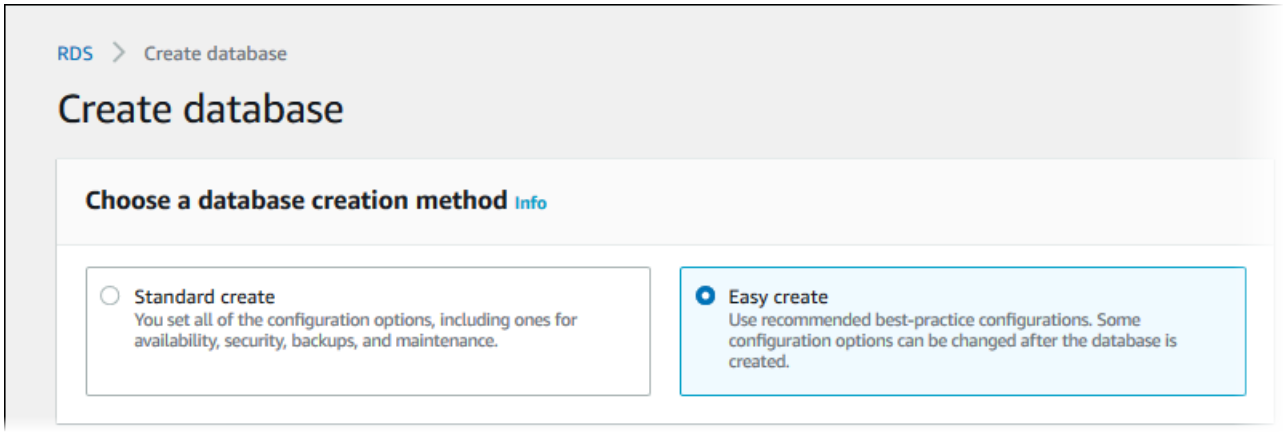
8. Attendi che Stato dell'istanza diventi In esecuzione per l'istanza EC2 prima di continuare.

Fase 2: creazione di un cluster di database Aurora PostgreSQL

In questo esempio, utilizzi la Creazione semplice per creare un cluster di database Aurora PostgreSQL con una classe di istanza database db.t4g.large.

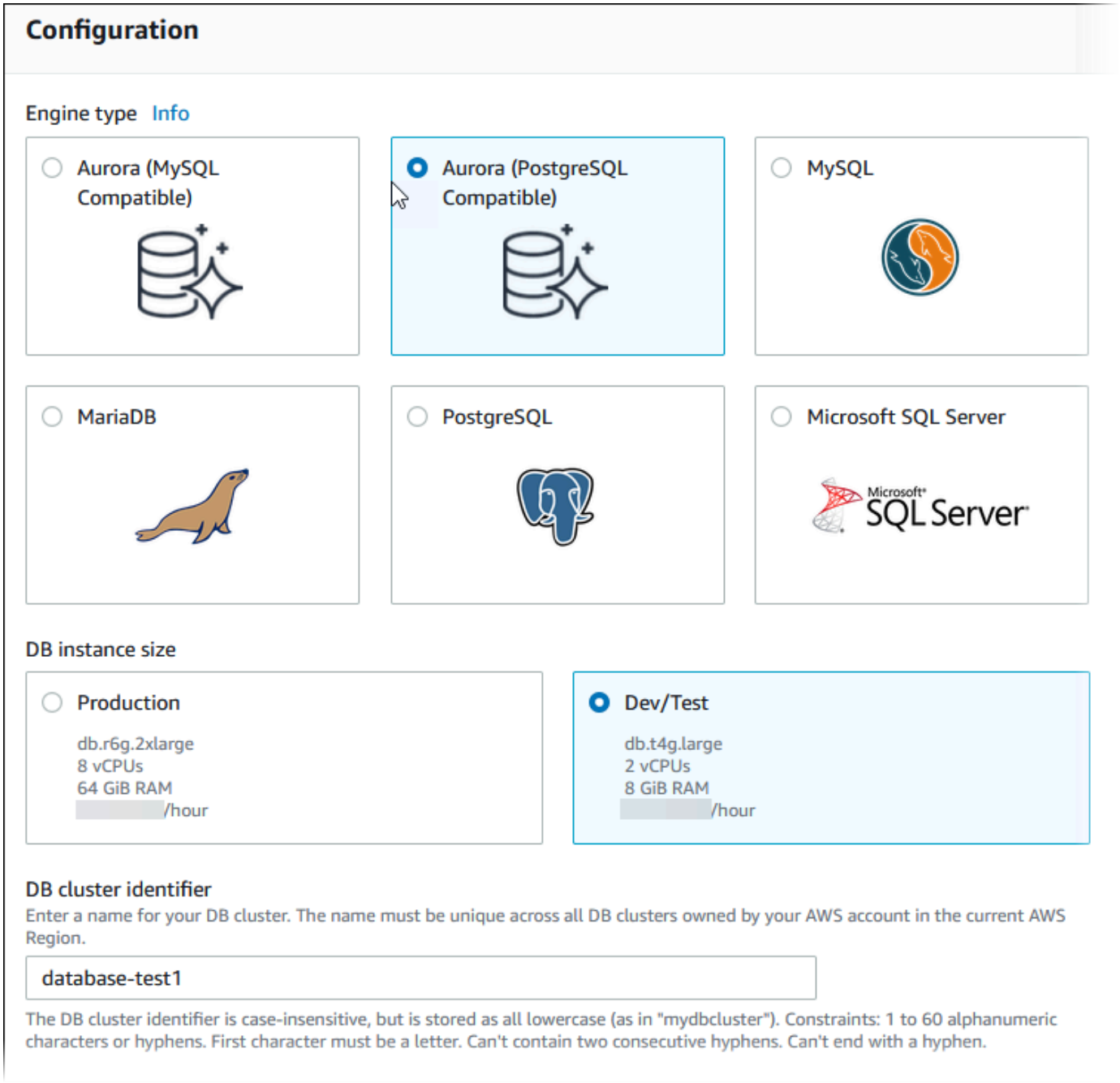
Per creare un cluster di database Aurora PostgreSQL con Creazione semplice

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console Amazon RDS, scegli la Regione AWS in cui desideri creare il cluster di database.
3. Nel riquadro di navigazione, scegli Databases (Database).
4. Scegli Crea database e verifica che l'opzione Creazione semplice sia selezionata.



5. In Configurazione scegli Aurora (compatibile con PostgreSQL) per Tipo di motore.
6. Per DB instance size (Dimensione istanza database), seleziona Dev/Test.
7. Per Identificatore cluster di database, immetti **`database-test1`**.

   La pagina Create database (Crea database) la pagina dovrebbe apparire simile alla seguente immagine.

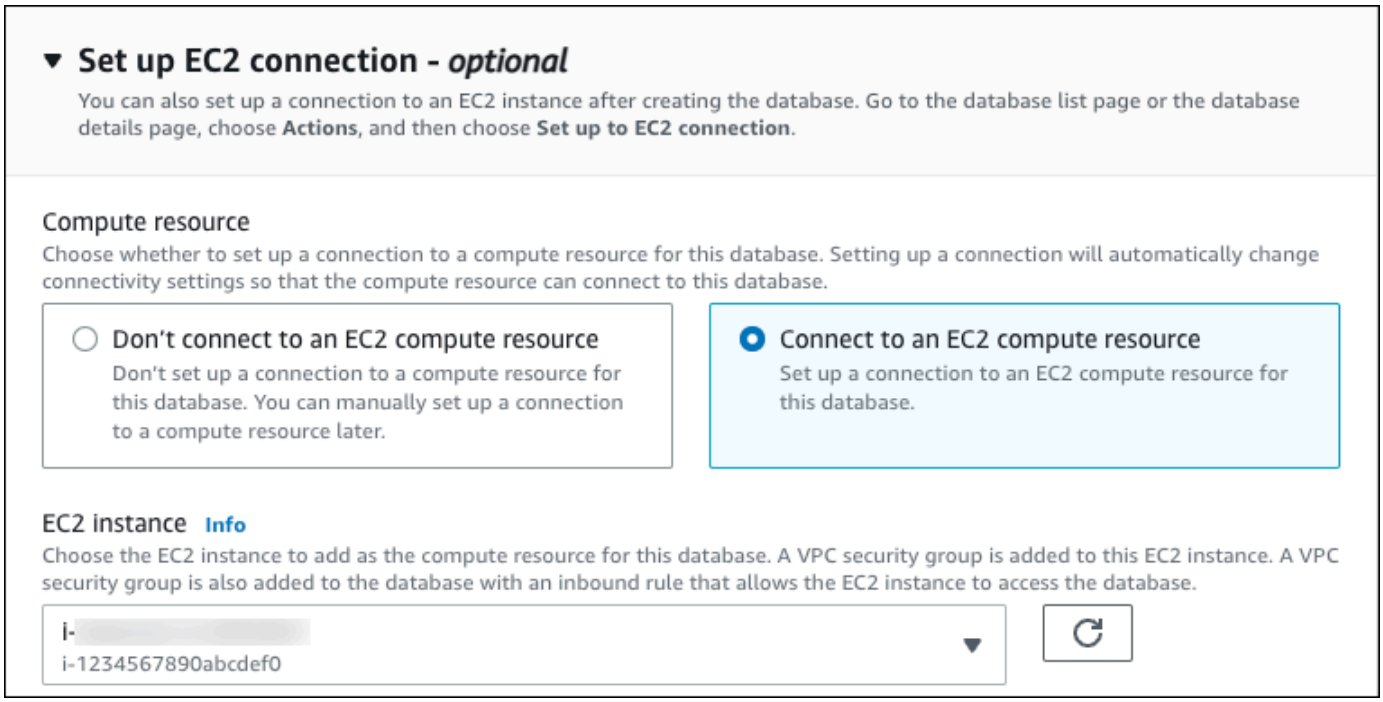


8. In Nome utente master, inserisci un nome per l'utente o lascia invariato il nome predefinito (**`postgres`**).
9. Per utilizzare una password master generata automaticamente per il cluster di database, seleziona Genera automaticamente una password.

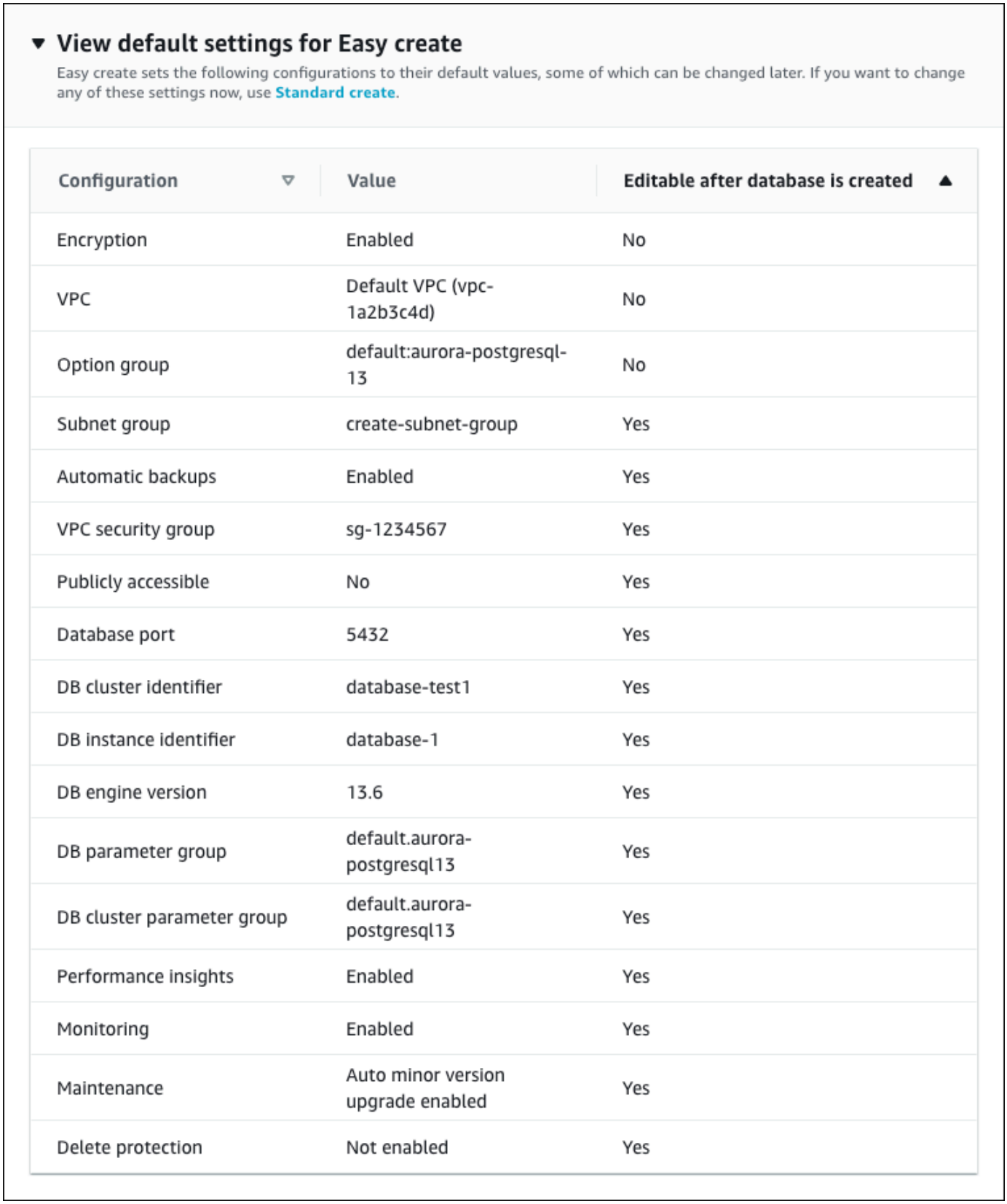
   Per inserire la password master, deseleziona la casella Genera automaticamente una password e inserisci la stessa password in Password master e Conferma password.

10. Per configurare una connessione con l'istanza EC2 creata in precedenza, apri Configura connessione EC2 - opzionale.

    Seleziona Connetti a una risorsa di calcolo EC2. Scegli l'istanza EC2 creata in precedenza.



11. Apri Visualizza le impostazioni predefinite per la creazione Semplice.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-postgresql-13	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	5432	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	13.6	Yes
DB parameter group	default.aurora-postgresql13	Yes
DB cluster parameter group	default.aurora-postgresql13	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Puoi esaminare le impostazioni predefinite utilizzate con Easy create (Creazione rapida). La colonna Modificabile dopo la creazione del database mostra le opzioni che puoi modificare dopo aver creato il database.

- Se un'impostazione contiene No in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database.
- Se un'impostazione contiene Sì in quella colonna e desideri cambiarla, puoi utilizzare la Creazione standard per creare il cluster di database o modificare il cluster di database dopo averlo creato per cambiare l'impostazione.

12. Scegliere Crea database.

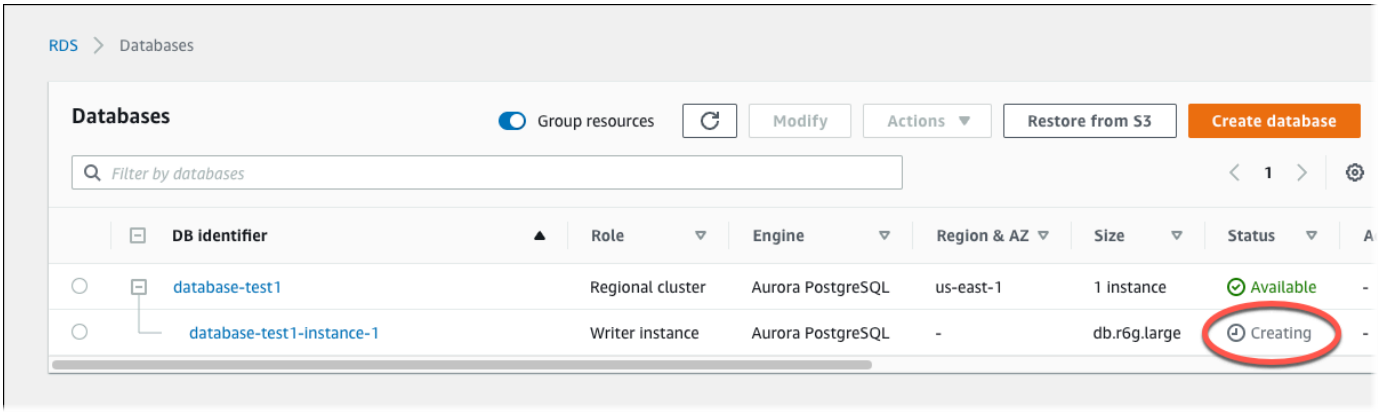
    Per vedere nome utente e password per il cluster di database, seleziona Visualizza i dettagli delle credenziali.

    Per connetterti al cluster di database come utente master, utilizza il nome utente e la password visualizzati.

    > **Important**
    >
    > Non potrai visualizzare di nuovo la password dell'utente principale. Se non la registri, potresti doverla modificare.
    > Se devi modificare la password dell'utente principale dopo che il cluster database è disponibile, puoi modificare il cluster database per eseguire tale operazione. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

13. Nell'elenco Database seleziona il nome del nuovo cluster di database Aurora PostgreSQL per visualizzarne i dettagli.

    L'istanza di scrittura ha lo stato Creazione in corso fino a quando il cluster di database non è pronto per essere utilizzato.



Quando lo stato dell'istanza di scrittura cambia in Disponibile, puoi connetterti al cluster di database. A seconda della classe di istanza database e della quantità di storage, prima che il nuovo cluster di database sia disponibile possono trascorrere fino a 20 minuti.

(Facoltativo) Crea VPC, istanza EC2 e cluster Aurora PostgreSQL utilizzando AWS CloudFormation

Invece di utilizzare la console per creare il VPC, l'istanza EC2 e il cluster DB Aurora PostgreSQL, puoi utilizzarla per fornire risorse trattando l'infrastruttura come codice. AWS CloudFormation AWS Per aiutarti a organizzare AWS le tue risorse in unità più piccole e più gestibili, puoi utilizzare la funzionalità nested stack. AWS CloudFormation Per ulteriori informazioni, consulta Creare uno stack sulla AWS CloudFormation console e Lavorare con gli stack annidati.

> **Important**
>
> AWS CloudFormation è gratuito, ma le risorse che CloudFormation crea sono attive. Ti verranno addebitati i costi di utilizzo standard per queste risorse fino alla loro cessazione. L'addebito totale sarà minimo. Per informazioni su come ridurre al minimo gli addebiti, vai al AWS piano gratuito.

Per creare le tue risorse utilizzando la AWS CloudFormation console, completa i seguenti passaggi:

- Passaggio 1: scarica il CloudFormation modello
- Passaggio 2: configura le tue risorse utilizzando CloudFormation

Scarica il CloudFormation modello

Un CloudFormation modello è un file di testo JSON o YAML che contiene le informazioni di configurazione sulle risorse che desideri creare nello stack. Questo modello crea anche un VPC e un bastion host per te insieme al cluster Aurora.

Per scaricare il file modello, apri il seguente link, modello Aurora PostgreSQL. CloudFormation

Nella pagina Github, fai clic sul pulsante Scarica file raw per salvare il file YAML del modello.

Configura le tue risorse usando CloudFormation

> **Note**
>
> Prima di iniziare questo processo, assicurati di avere una coppia di chiavi per un'istanza EC2 nel tuo Account AWS. Per ulteriori informazioni, consulta Coppie di chiavi Amazon EC2 e istanze Linux.

Quando utilizzi il AWS CloudFormation modello, devi selezionare i parametri corretti per assicurarti che le risorse vengano create correttamente. Segui la procedura riportata di seguito:

1. Accedi AWS Management Console e apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegli Crea stack.
3. Nella sezione Specificare il modello, seleziona Carica un file modello dal computer, quindi scegli Avanti.
4. Nella pagina Specificare i dettagli dello stack, imposta i seguenti parametri:
   a. Imposta il nome dello stack su AurPostgre SQL. TestStack
   b. In Parametri, imposta le zone di disponibilità selezionando due zone di disponibilità.
   c. Nella configurazione Linux Bastion Host, in Key Name, seleziona una coppia di chiavi per accedere alla tua istanza EC2.
   d. Nelle impostazioni di configurazione di Linux Bastion Host, imposta l'intervallo IP consentito sul tuo indirizzo IP. Per connetterti alle istanze EC2 nel tuo VPC utilizzando Secure Shell (SSH), determina il tuo indirizzo IP pubblico utilizzando il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è 192.0.2.1/32.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze EC2 pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze EC2 utilizzando SSH.

e. Nella configurazione generale del database, imposta la classe dell'istanza del database su db.t4g.large.

f. Imposta il nome del database su. **`database-test1`**

g. Per Nome utente principale del database, inserisci un nome per l'utente principale.

h. Imposta la password utente principale di Manage DB con Secrets Manager su `false` per questo tutorial.

i. Per la password del database, imposta una password a tua scelta. Ricorda questa password per ulteriori passaggi del tutorial.

j. Imposta la distribuzione Multi-AZ su`false`.

k. Lascia tutte le altre impostazioni come valori predefiniti. Fate clic su Avanti per continuare.

5. Nella pagina Configura le opzioni dello stack, lascia tutte le opzioni predefinite. Fai clic su Avanti per continuare.
6. Nella pagina Review stack, seleziona Invia dopo aver verificato le opzioni del database e dell'host Linux bastion.

Una volta completato il processo di creazione dello stack, visualizza gli stack con i nomi BastionStacke APGNS per annotare le informazioni necessarie per connetterti al database. Per ulteriori informazioni, vedere Visualizzazione dei dati e delle risorse AWS CloudFormation dello stack su. AWS Management Console

Fase 3: connessione a un cluster di database Aurora PostgreSQL

È possibile utilizzare qualsiasi applicazione client PostgreSQL standard per la connessione al cluster di database. In questo esempio, ti connetti a un cluster di database Aurora PostgreSQL utilizzando il client della linea di comando psql.

Per effettuare la connessione a un cluster di database Aurora PostgreSQL

1. Individua l'endpoint (nome DNS) e il numero di porta per l'istanza di scrittura del cluster di database.

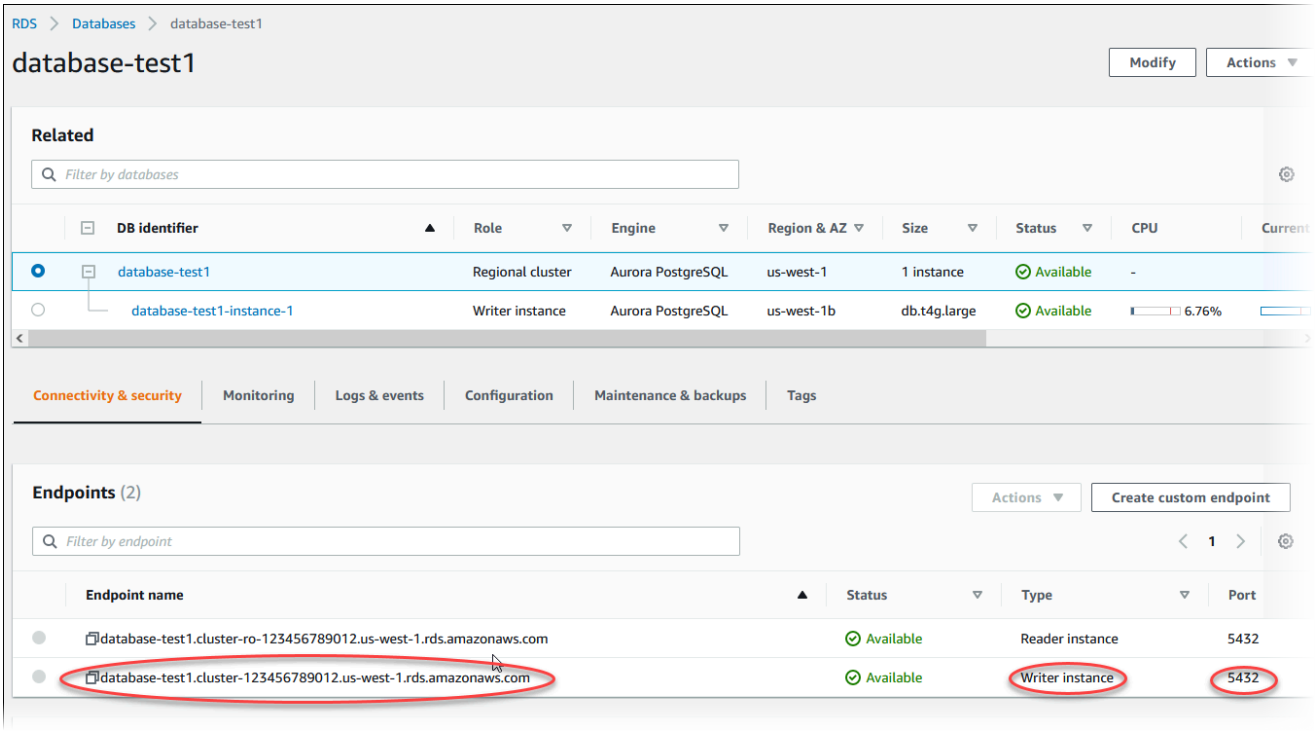
   a. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   b. Nell'angolo in alto a destra della console Amazon RDS, scegli Regione AWS per il cluster DB.

   c. Nel riquadro di navigazione, scegli Databases (Database).

   d. Scegli il nome del cluster di database Aurora PostgreSQL per visualizzarne i dettagli.

   e. Nella scheda Connettività e sicurezza, copia l'endpoint dell'istanza di scrittura. Annotare anche il numero di porta. L'endpoint e il numero di porta sono necessari per la connessione al cluster di database.



2. Connettiti all'istanza EC2 che hai creato in precedenza seguendo i passaggi in Connetti alla tua istanza Linux nella Amazon EC2 User Guide.

   Ti consigliamo di connetterti all'istanza EC2 tramite SSH. Se l'utilità client SSH è installata su Windows, Linux o Mac, puoi connetterti all'istanza utilizzando il comando nel seguente formato:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Ad esempio, supponi che `ec2-database-connect-key-pair.pem` sia archiviato in `/dir1` su Linux e che il DNS IPv4 pubblico per l'istanza EC2 sia `ec2-12-345-678-90.compute-1.amazonaws.com`. Il comando SSH sarà simile al seguente:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

3. Ottieni le ultime correzioni di bug e gli aggiornamenti di sicurezza aggiornando il software sulla tua istanza EC2. A tale scopo, utilizzare il comando seguente.

> **Note**
>
> L'opzione `-y` installa gli aggiornamenti senza chiedere conferma. Per esaminare gli aggiornamenti prima di installarli, omettere questa opzione.

```
sudo dnf update -y
```

4. Installa il client della linea di comando psql da PostgreSQL su Amazon Linux 2023 utilizzando il seguente comando:

```
sudo dnf install postgresql15
```

5. Effettua la connessione al cluster di database Aurora PostgreSQL. Ad esempio, immetti il comando seguente: Questa azione ti consente di effettuare la connessione al cluster di database Aurora PostgreSQL utilizzando il client psql.

   Sostituisci l'endpoint dell'istanza di scrittura per *`endpoint`*, sostituisci il nome database `--dbname` a cui vuoi connetterti per *`postgres`* e sostituisci il nome utente master utilizzato per *`postgres`*. Devi fornire la password master utilizzata quando viene richiesta una password.

```
psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
```

   Dopo aver immesso la password per l'utente, l'output dovrebbe essere analogo a quanto mostrato di seguito.

```
psql (14.3, server 14.6)
```

```
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

Per ulteriori informazioni sulla connessione a un cluster di database Aurora PostgreSQL, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL. In caso di mancata connessione al cluster database, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Per motivi di sicurezza, la best practice è utilizzare connessioni crittografate. Utilizza una connessione PostgreSQL non crittografata solo quando il client e il server sono nello stesso VPC e la rete è attendibile. Per ulteriori informazioni sull'uso di connessioni crittografate, consulta Sicurezza dei dati Aurora PostgreSQL con SSL/TLS.

6. Eseguire comandi SQL.

   Ad esempio, il seguente comando SQL mostra la data e l'ora correnti:

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Fase 4: eliminazione dell'istanza EC2 e del cluster di database

Dopo la connessione e l'esplorazione dell'istanza EC2 e del cluster di database di esempio che hai creato, eliminale per evitare di ricevere l'addebito dei relativi costi.

Se in passato AWS CloudFormation creavi risorse, salta questo passaggio e vai al passaggio successivo.

Per eliminare l'istanza EC2

1. Accedi AWS Management Console e apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel riquadro di navigazione, seleziona Istanze.
3. Seleziona l'istanza EC2 e scegli Stato istanza, Termina istanza.
4. Quando viene richiesta la conferma, seleziona Terminate (Interrompi).

Per ulteriori informazioni sull'eliminazione di un'istanza EC2, consulta Terminate your instance nella Amazon EC2 User Guide.

Per eliminare un cluster database

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database), quindi selezionare l'istanza database associata con il cluster di database.
3. In Actions (Azioni), selezionare Delete (Elimina).
4. Scegliere Delete (Elimina).

   Dopo aver eliminato tutte le istanze database associate a un cluster di database, il cluster di database viene eliminato automaticamente.

(Facoltativo) Elimina l'istanza EC2 e il cluster DB creati con CloudFormation

Se prima creavi AWS CloudFormation risorse, elimina lo CloudFormation stack dopo esserti connesso ed esplorato l'istanza EC2 di esempio e il cluster DB, in modo che non ti vengano più addebitati costi.

Per eliminare le risorse CloudFormation

1. Apri la AWS CloudFormation console.
2. Nella pagina Stacks della CloudFormation console, seleziona lo stack principale (lo stack senza il nome VPCStack o APGNS). BastionStack
3. Scegli Elimina.
4. Seleziona Elimina stack quando viene richiesta la conferma.

Per ulteriori informazioni sull'eliminazione di uno stack in CloudFormation, consulta Eliminazione di uno stack sulla console nella Guida per l' AWS CloudFormationutente.AWS CloudFormation

(Facoltativo) Connessione del cluster database a una funzione Lambda

Puoi anche connettere il tuo cluster Aurora PostgreSQL DB a una risorsa di elaborazione serverless Lambda. Le funzioni Lambda consentono di eseguire il codice senza il provisioning o la gestione dell'infrastruttura. Una funzione Lambda consente inoltre di rispondere automaticamente alle richieste

di esecuzione del codice su qualsiasi scala, da una dozzina di eventi al giorno a centinaia al secondo. Per ulteriori informazioni, consulta Connessione automatica di una funzione Lambda e di un cluster database Aurora.

Tutorial: creazione di un server Web e un cluster database Amazon Aurora

Questo tutorial descrive come installare un server Web Apache con PHP e creare un database MariaDB, MySQL o PostgreSQL. Il server Web viene eseguito in un'istanza Amazon EC2 utilizzando Amazon Linux 2023 e puoi scegliere tra un cluster database Aurora MySQL o Aurora PostgreSQL. Sia l'istanza Amazon EC2 che l'Cluster DB vengono eseguite in un virtual private cloud (VPC) basato sul servizio Amazon VPC.

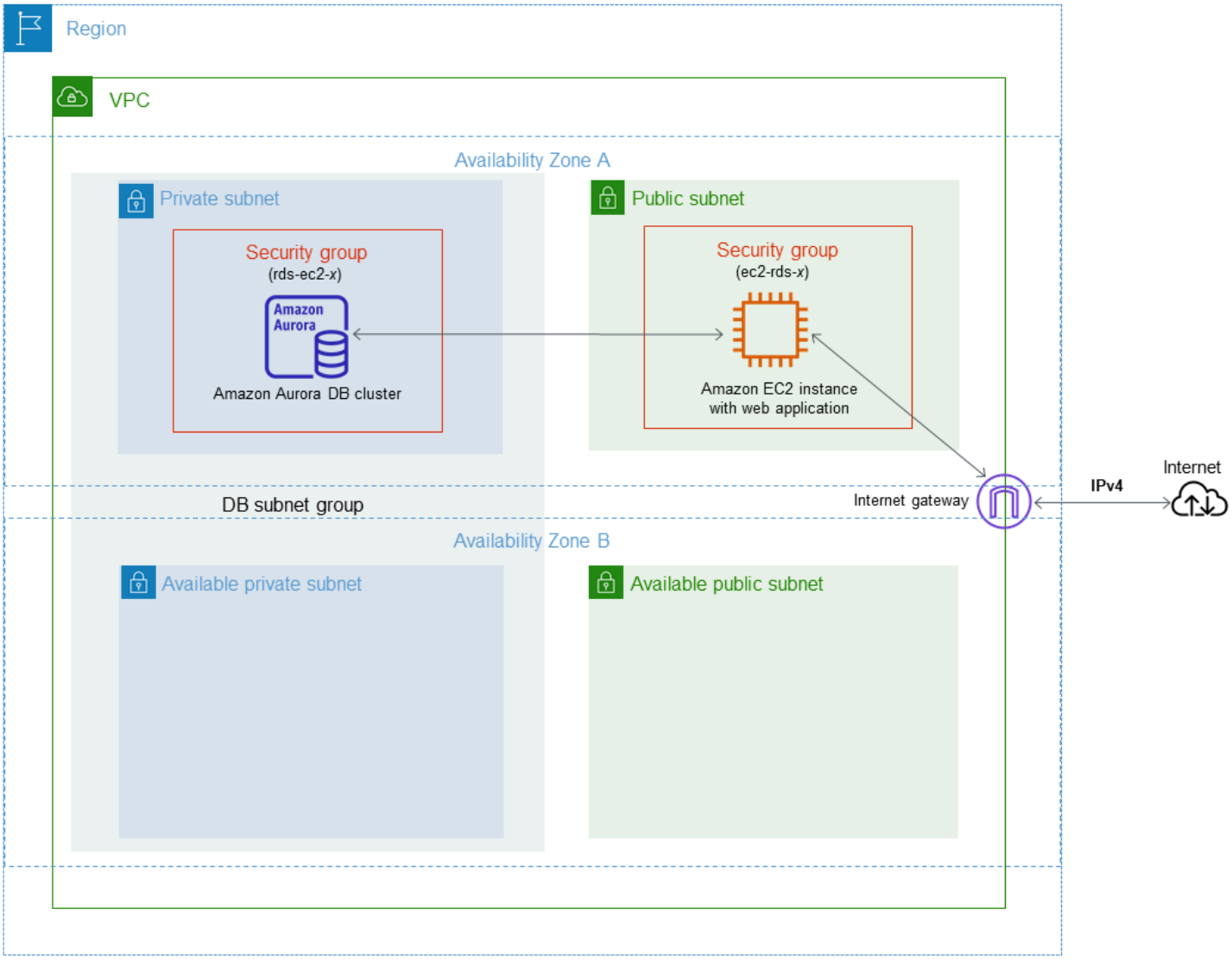
> **Important**
>
> Non vi sono costi aggiuntivi per la creazione di un account AWS. Tuttavia, completando l'esercitazione, potresti incorrere in costi per le risorse AWS utilizzate. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

> **Note**
>
> Questo tutorial funziona con Amazon Linux 2023 e potrebbe non funzionare per altre versioni di Linux.

Nel tutorial che segue, viene creata un'istanza EC2 che utilizza VPC, sottoreti e gruppo di sicurezza predefiniti per Account AWS. In questo tutorial viene illustrato come creare il cluster database e configurare automaticamente la connettività con l'istanza EC2 creata. Nel tutorial viene quindi mostrato come installare il server Web sull'istanza EC2. Il server Web viene connesso al cluster database nel VPC utilizzando l'endpoint del cluster di scrittura del database.

1. Avvia un'EC2istanza per connetterti con l'
2. Creazione di un cluster di database Amazon Aurora
3. Installazione di un server Web nell'istanza EC2

Il seguente diagramma illustra la configurazione al completamento del tutorial.



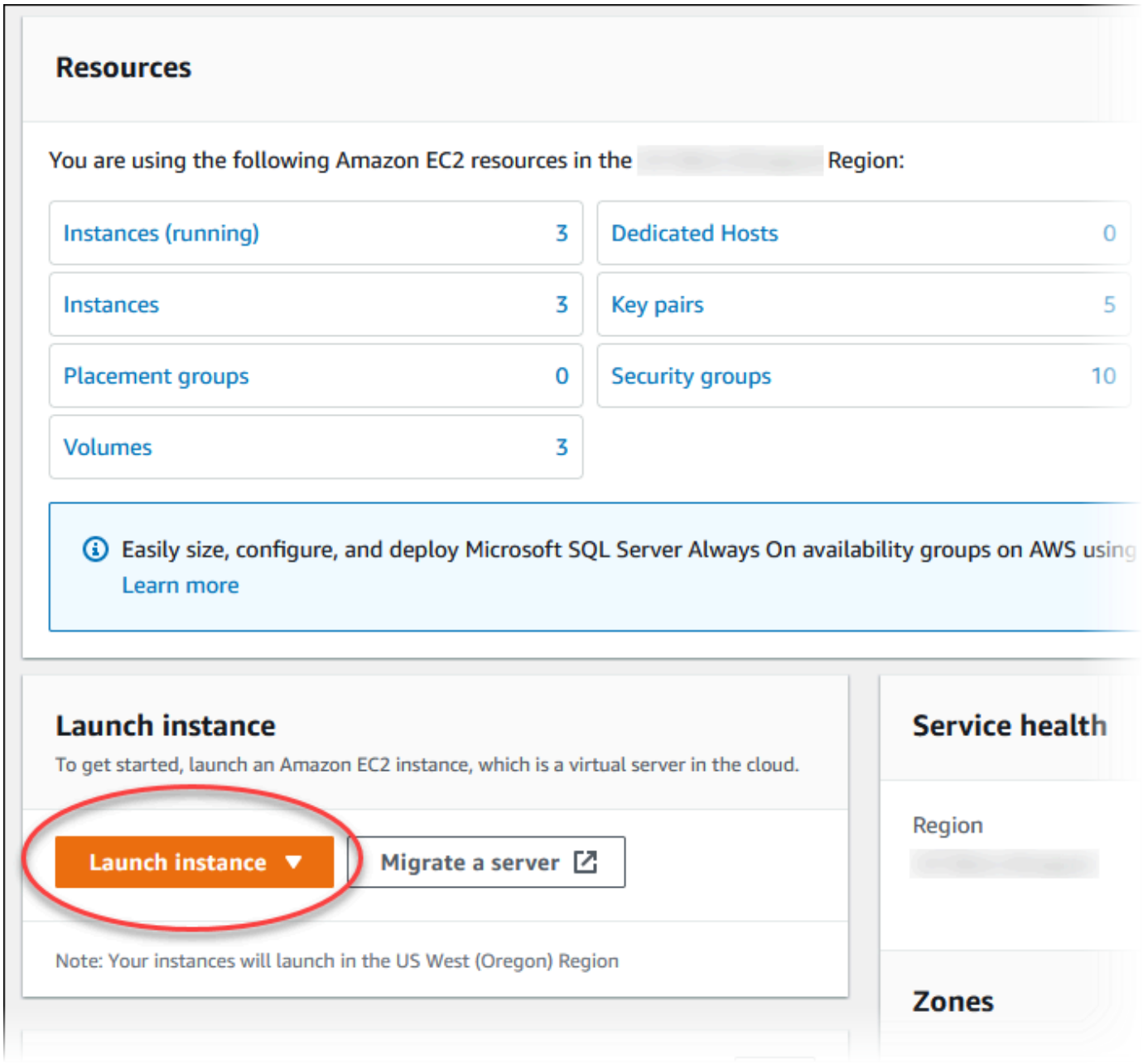
> **Note**
>
> Dopo aver completato il tutorial, è presente una sottorete pubblica e una privata in ogni zona di disponibilità del VPC. Questo tutorial utilizza il VPC predefinito per Account AWS e configura automaticamente la connettività tra l'istanza EC2 e il cluster database. Se preferisci invece configurare un nuovo VPC per questo scenario, completa le attività in Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 .

Avvia un'EC2istanza per connetterti con l'

Crea un'EC2istanza Amazon nella sottorete pubblica del tuoVPC.

Per avviare un'istanza EC2

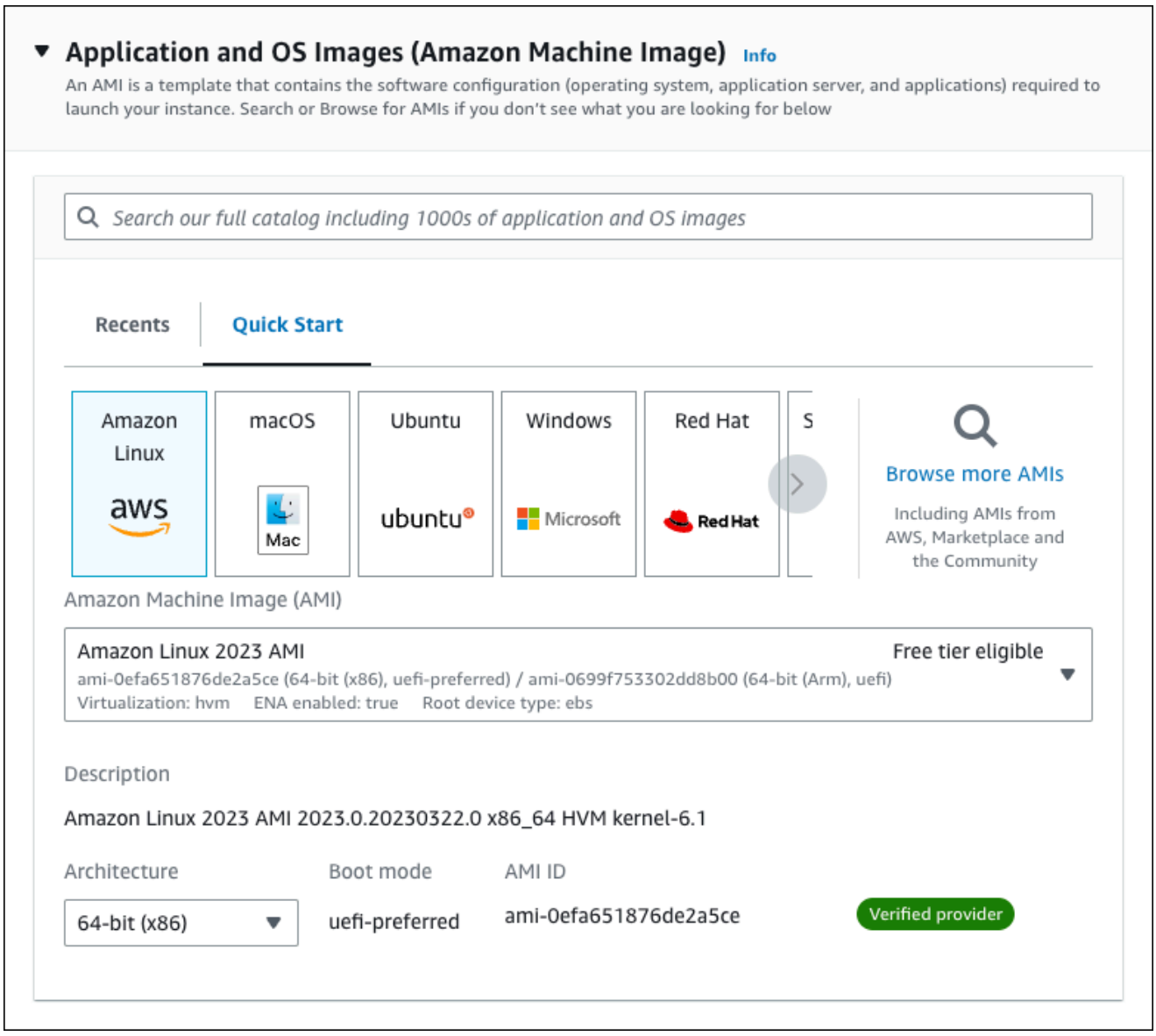
1. Accedi a AWS Management Console e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli Regione AWS dove vuoi creare l'istanza. EC2
3. Scegli EC2Dashboard, quindi scegli Launch instance, come mostrato di seguito.



4. Scegli le seguenti impostazioni nella pagina Avvia un'istanza.

a. Nell'area Name and tags (Nome e tag), in Name (Nome) inserisci **`tutorial-ec2-instance-web-server`**.

b. In Immagini di applicazioni e sistemi operativi (Amazon Machine Image), scegli Amazon Linux, quindi scegli Amazon Linux 2023 AMI. Manteni i valori predefiniti per le altre opzioni.



c. In Instance type (Tipo di istanza), scegli t2.micro.

d. In Key pair (login) (Coppia di chiavi (login), per Key pair name (Nome della coppia di chiavi), scegli una coppia di chiavi esistente. Per creare una nuova coppia di chiavi per l'EC2istanza Amazon, scegli Crea nuova coppia di chiavi, quindi utilizza la finestra Crea key pair per crearla.

Per ulteriori informazioni sulla creazione di una nuova coppia di chiavi, consulta Create a key pair nella Amazon EC2 User Guide.

e. In Network settings (Impostazioni di rete), imposta questi valori e conserva le impostazioni predefinite degli altri valori:

- In Consenti SSH traffico da, scegli la fonte delle SSH connessioni all'EC2istanza.

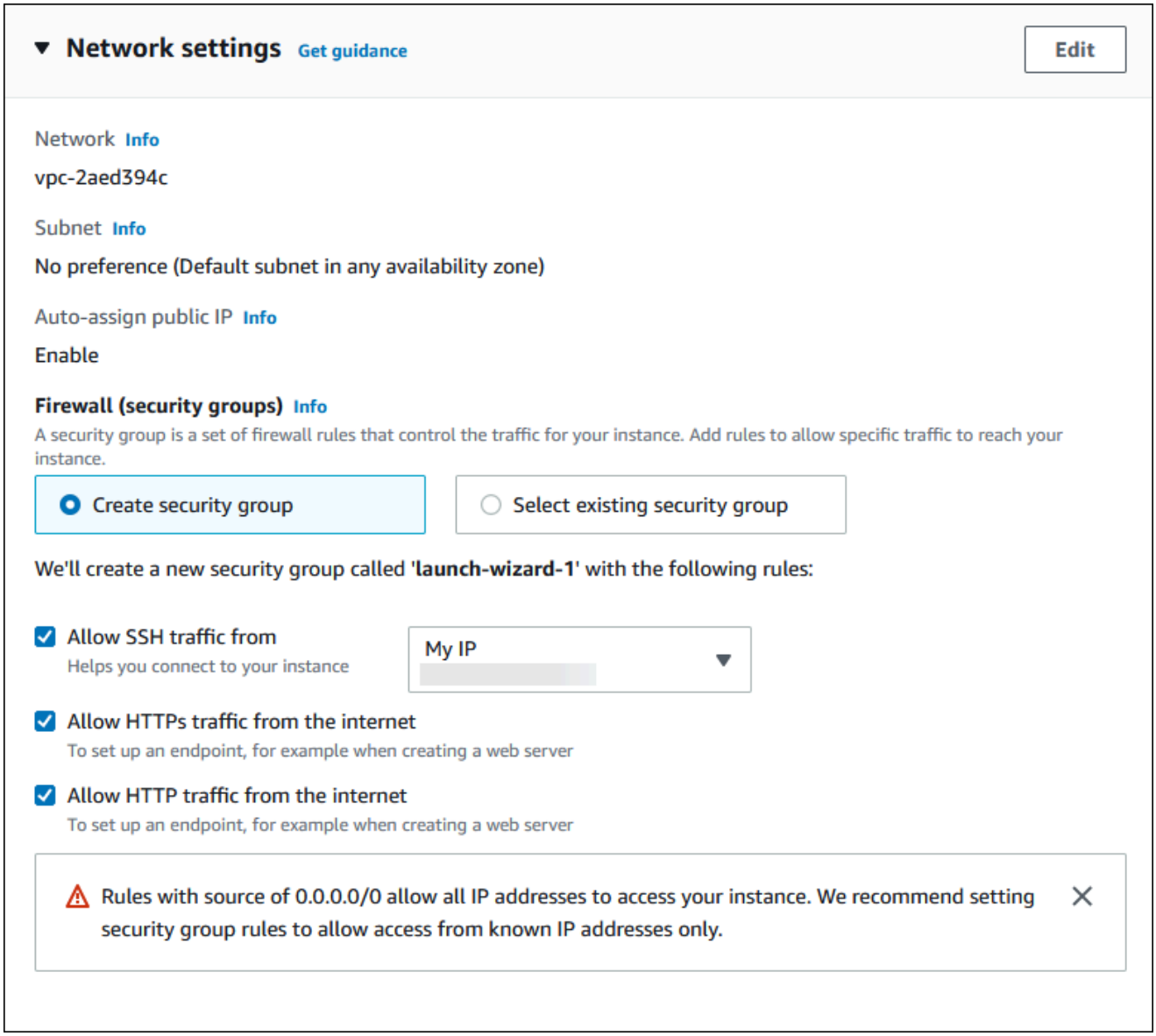
  Puoi scegliere My IP se l'indirizzo IP visualizzato è corretto per SSH le connessioni.

  Altrimenti, puoi determinare l'indirizzo IP da utilizzare per connetterti alle EC2 istanze VPC utilizzando Secure Shell (SSH). Per determinare l'indirizzo IP pubblico, in una finestra o una scheda del browser diversa, è possibile utilizzare il servizio all'indirizzo https://checkip.amazonaws.com. Un esempio di indirizzo IP è `203.0.113.25/32`.

  In molti casi, è possibile connettersi tramite un provider di servizi Internet (ISP) o da una protezione protetta dal firewall senza un indirizzo IP statico. In tal caso, accertati di determinare l'intervallo di indirizzi IP utilizzati dai computer client.

  > **Warning**
  >
  > Se lo utilizzi `0.0.0.0/0` per SSH l'accesso, consenti a tutti gli indirizzi IP di accedere alle tue istanze pubbliche utilizzandoSSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo uno specifico indirizzo IP o un intervallo di indirizzi da utilizzare per accedere alle istanze. SSH

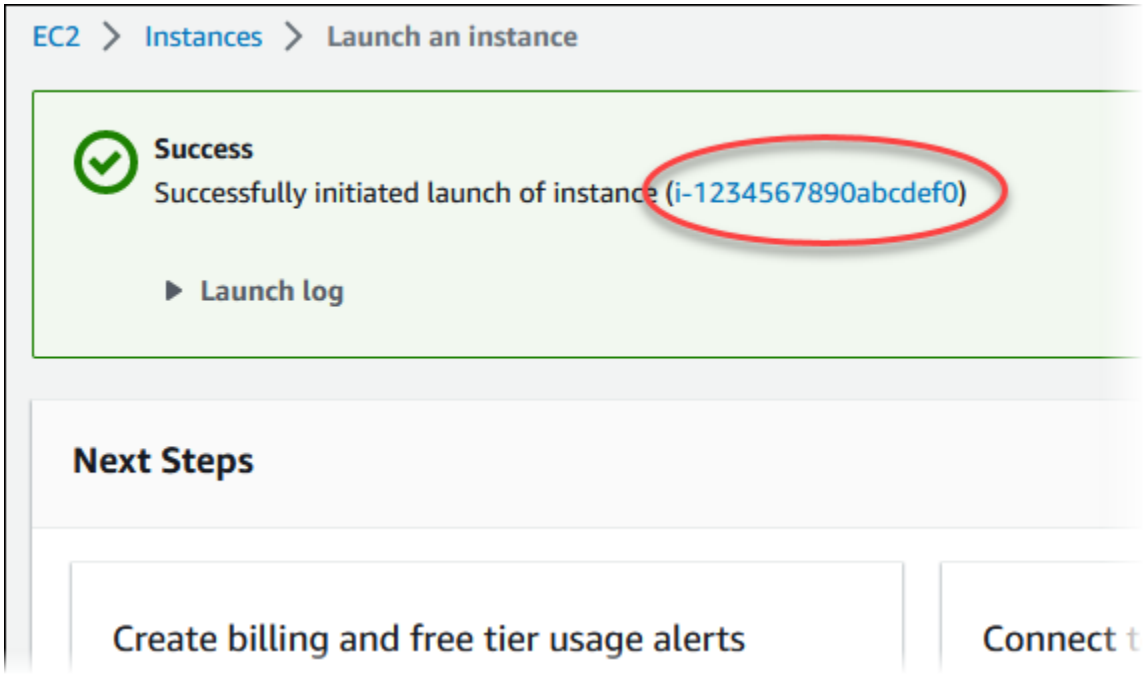
- Attiva Consenti HTTPs il traffico da Internet.
- Attiva Consenti il HTTP traffico da Internet.



f. Lascia i valori predefiniti per le sezioni rimanenti.

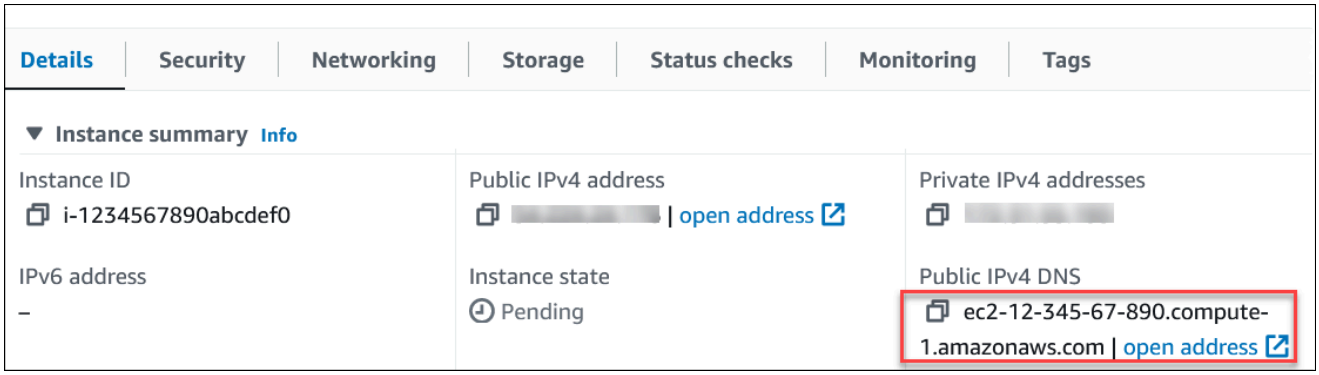
g. Analizza un riepilogo della configurazione dell'istanza nel pannello Summary (Riepilogo) e, quando è tutto pronto, scegli Launch instance (Avvia istanza).

5. Nella pagina Launch Status, annota l'identificatore della nuova EC2 istanza, ad esempio:`i-1234567890abcdef0`.

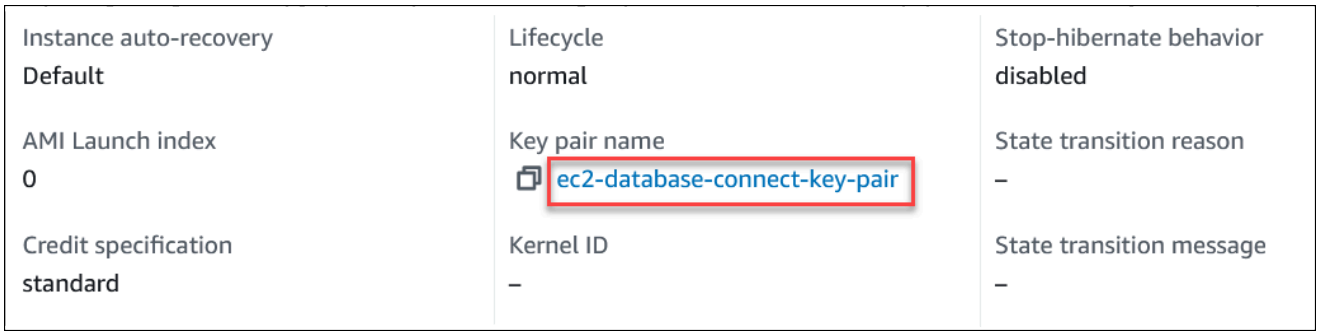


6. Scegli l'identificatore dell'EC2istanza per aprire l'elenco delle EC2 istanze, quindi seleziona l'istanza. EC2
7. Nella scheda Dettagli, annota i seguenti valori, che ti servono quando ti connetti tramite: SSH

   a. Nel riepilogo dell'istanza, annota il valore di Public IPv4 DNS.



   b. In Dettagli istanza, annota il valore visualizzato in Nome coppia di chiavi.



8. Attendi che Instance state (Stato istanza) per l'istanza sia Running (In esecuzione) prima di continuare.
9. Completo Creazione di un cluster di database Amazon Aurora.

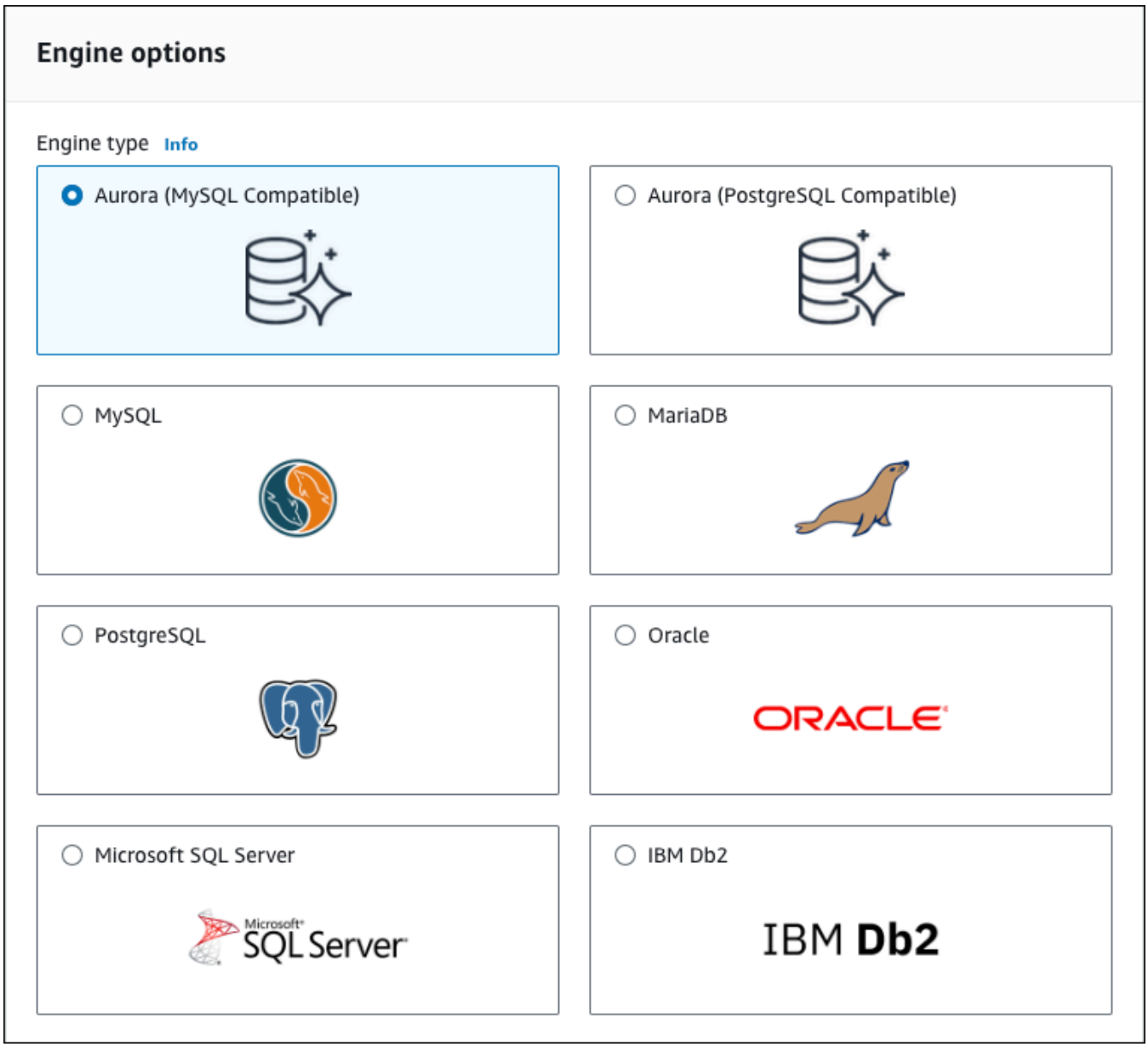
Creazione di un cluster di database Amazon Aurora

Crea un cluster database Amazon Aurora MySQL o Aurora PostgreSQL per conservare i dati utilizzati da un'applicazione web.

Aurora MySQL

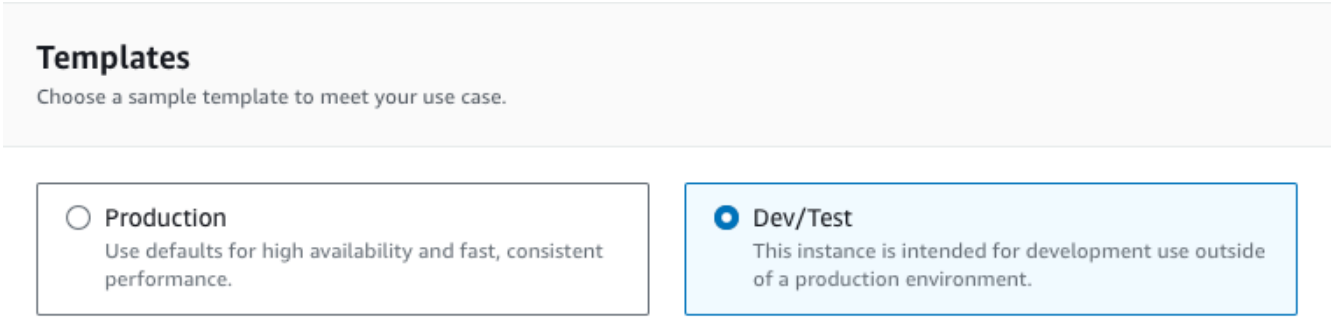
Creazione di un cluster di database Aurora MySQL

1. Accedi alla AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della AWS Management Console, assicurati che la Regione AWS sia identica a quella in cui hai creato l'istanza EC2.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).
5. Nella pagina Crea database scegli Creazione standard.
6. In Opzioni motore scegli Aurora (compatibile con MySQL).



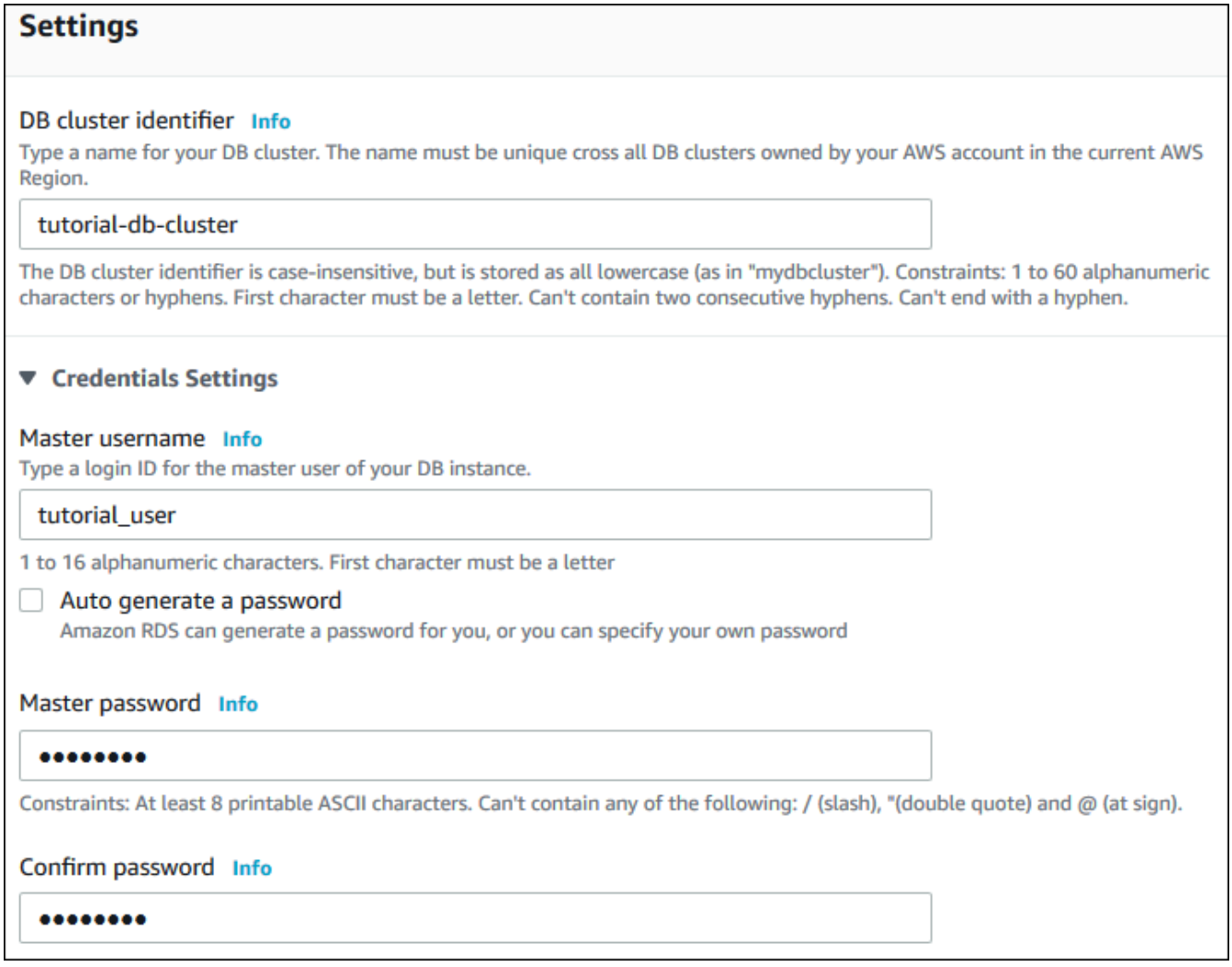
Mantenere i valori predefiniti per la versione e le altre opzioni del motore.

7. Nella sezione Templates (Modelli), selezionare Dev/Test.



8. Nella sezione Settings (Impostazioni) impostare questi valori:

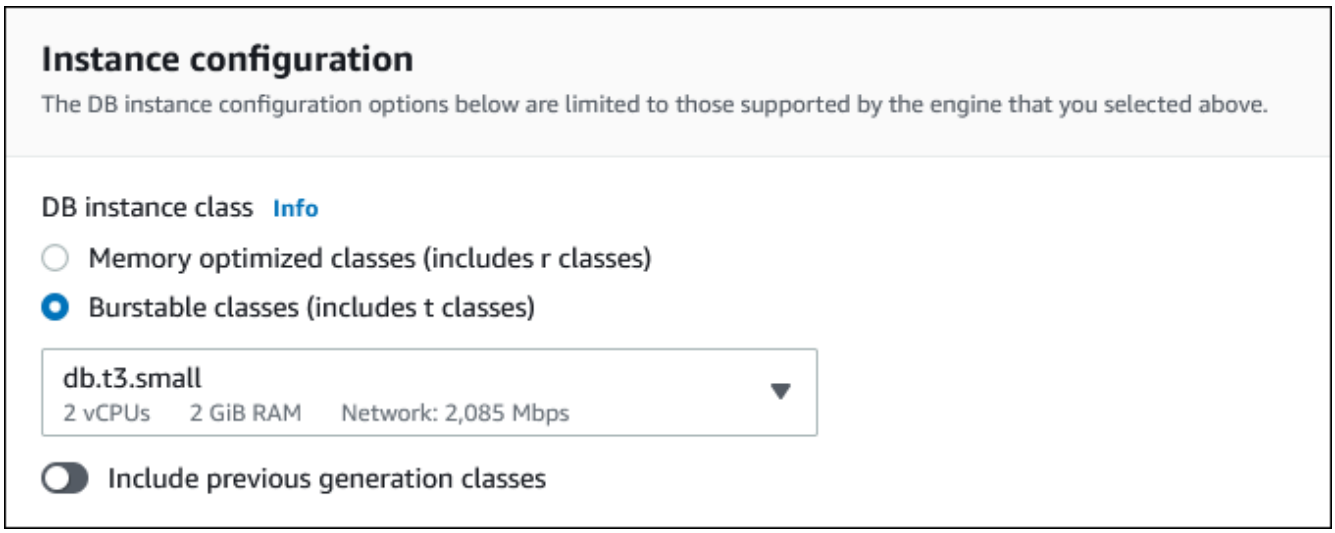
   - DB cluster identifier (Identificatore cluster DB): digita **`tutorial-db-cluster`**.
   - Master username (Nome utente master): digita **`tutorial_user`**.
   - Auto generate a password (Genera automaticamente una password): lascia l'opzione disattivata.
   - Master password (Password master): scegli una password.
   - Confirm password (Conferma la password): –digitare di nuovo la password.



9. Nella sezione Instance configuration (Configurazione dell'istanza), imposta i seguenti valori:

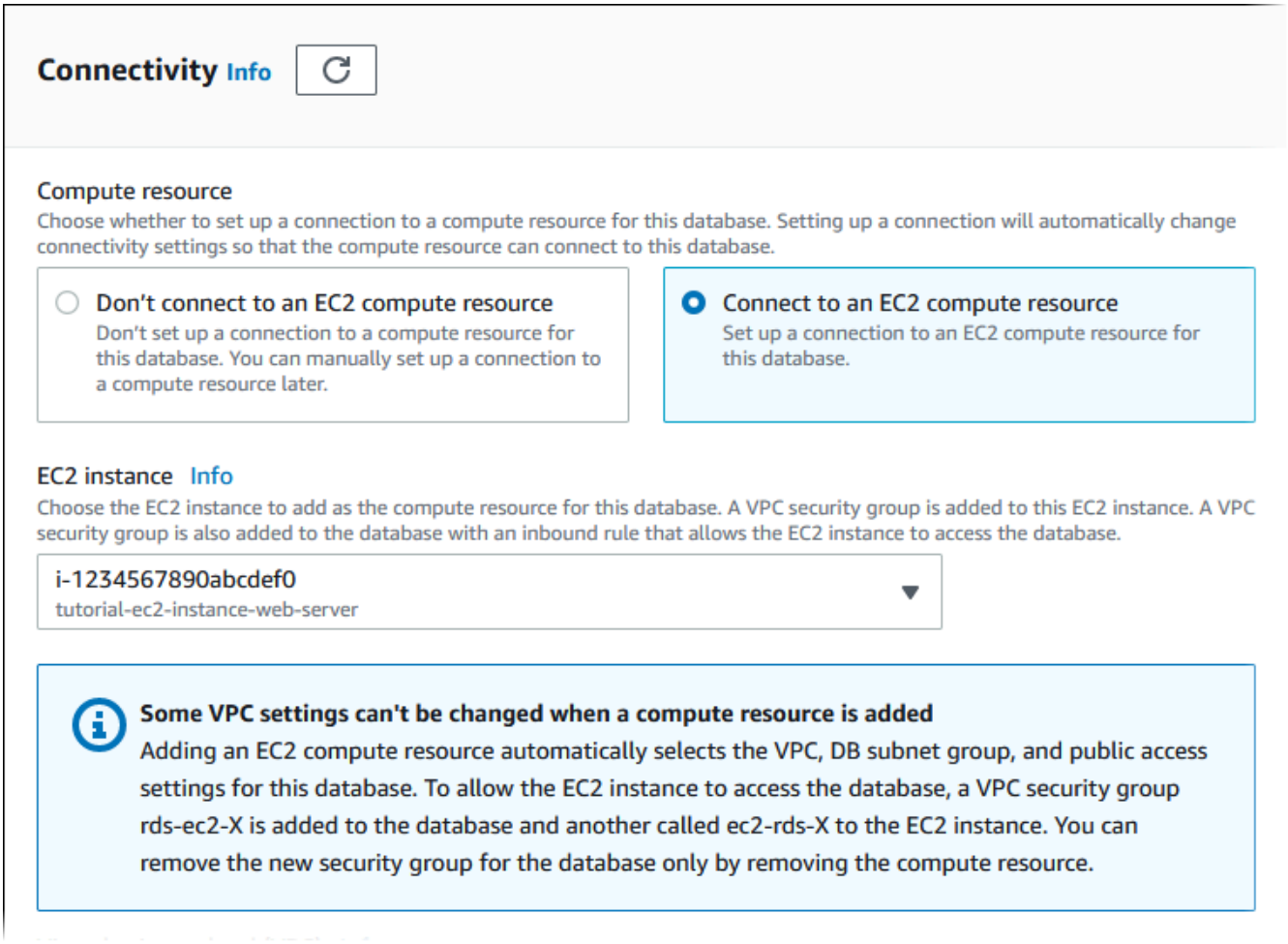
   - Classi espandibili (include le classi t)
   - db.t3.small o db.t3.medium

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.



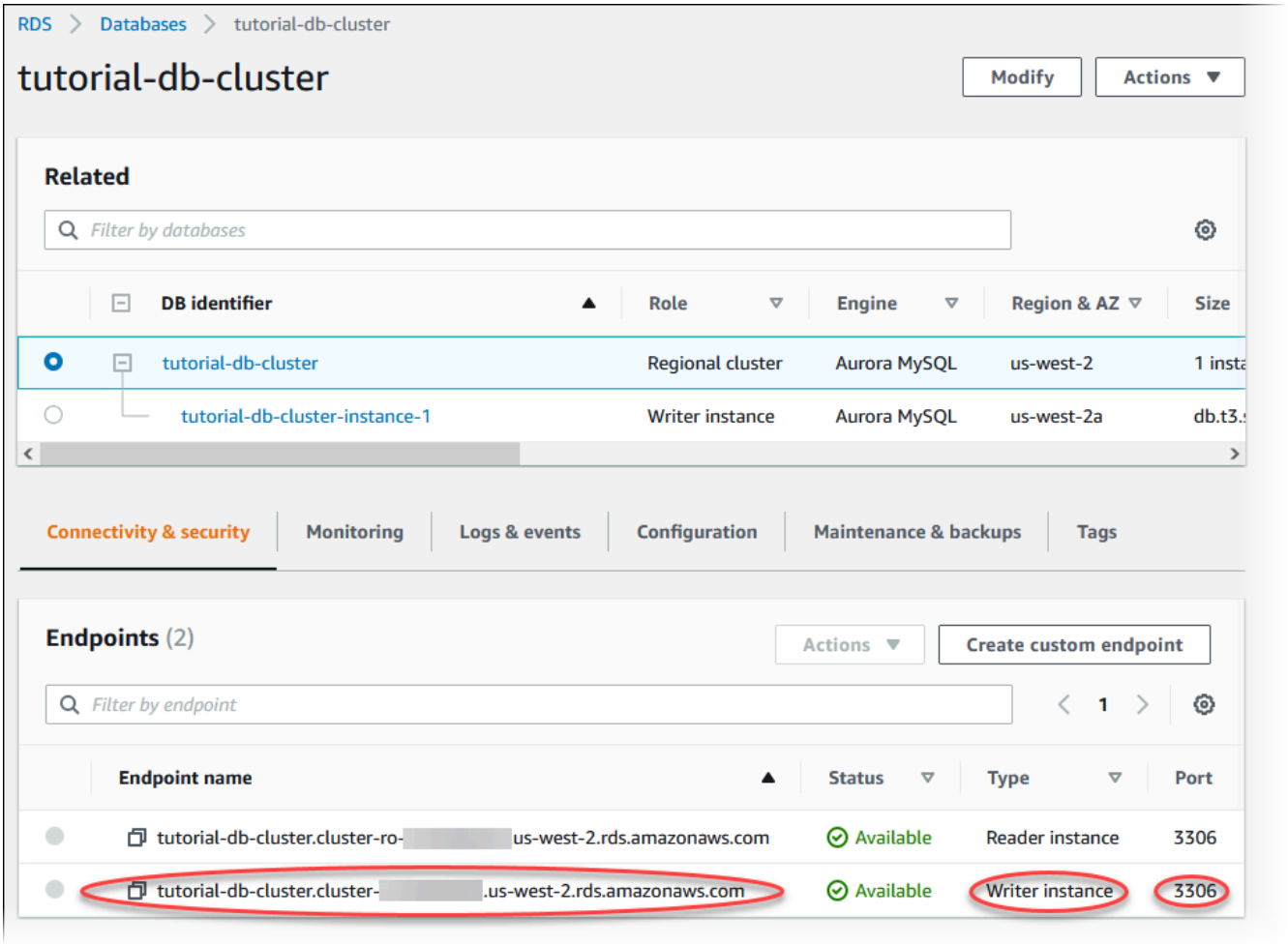
10. Nella sezione Availability and durability (Disponibilità e durata), utilizza i valori predefiniti.
11. Nella sezione Connectivity (Connettività), imposta i seguenti valori e lascia gli altri valori come predefiniti:

    - In Compute resource (Risorse di calcolo), seleziona Connect to an EC2 compute resource (Connetti a una risorsa di calcolo EC2).
    - Per l'istanza EC2, scegli l'istanza EC2 che hai creato in precedenza, ad esempio tutorial-ec2 -. instance-web-server



12. Aprire la sezione Additional configuration (Configurazione aggiuntiva) e specificare **`sample`** per Initial database name (Nome database iniziale). Lasciare le impostazioni predefinite per le altre opzioni.
13. Per creare il cluster Aurora MySQL DB, scegliere Crea database.

    Il nuovo cluster di database apparirà nell'elenco Database con lo stato Creazione in corso.
14. Attendere che lo Stato del nuovo cluster del database appaia come Disponibile. Quindi scegliere il nome del cluster del database per visualizzarne i dettagli.
15. Nella sezione Connettività e sicurezza, visualizzare Endpoint e Porta dell'istanza di scrittura del database.



Prendere nota dell'endpoint e della porta dell'istanza database writer. Queste informazioni verranno utilizzate per effettuare la connessione del server Web al cluster DB.

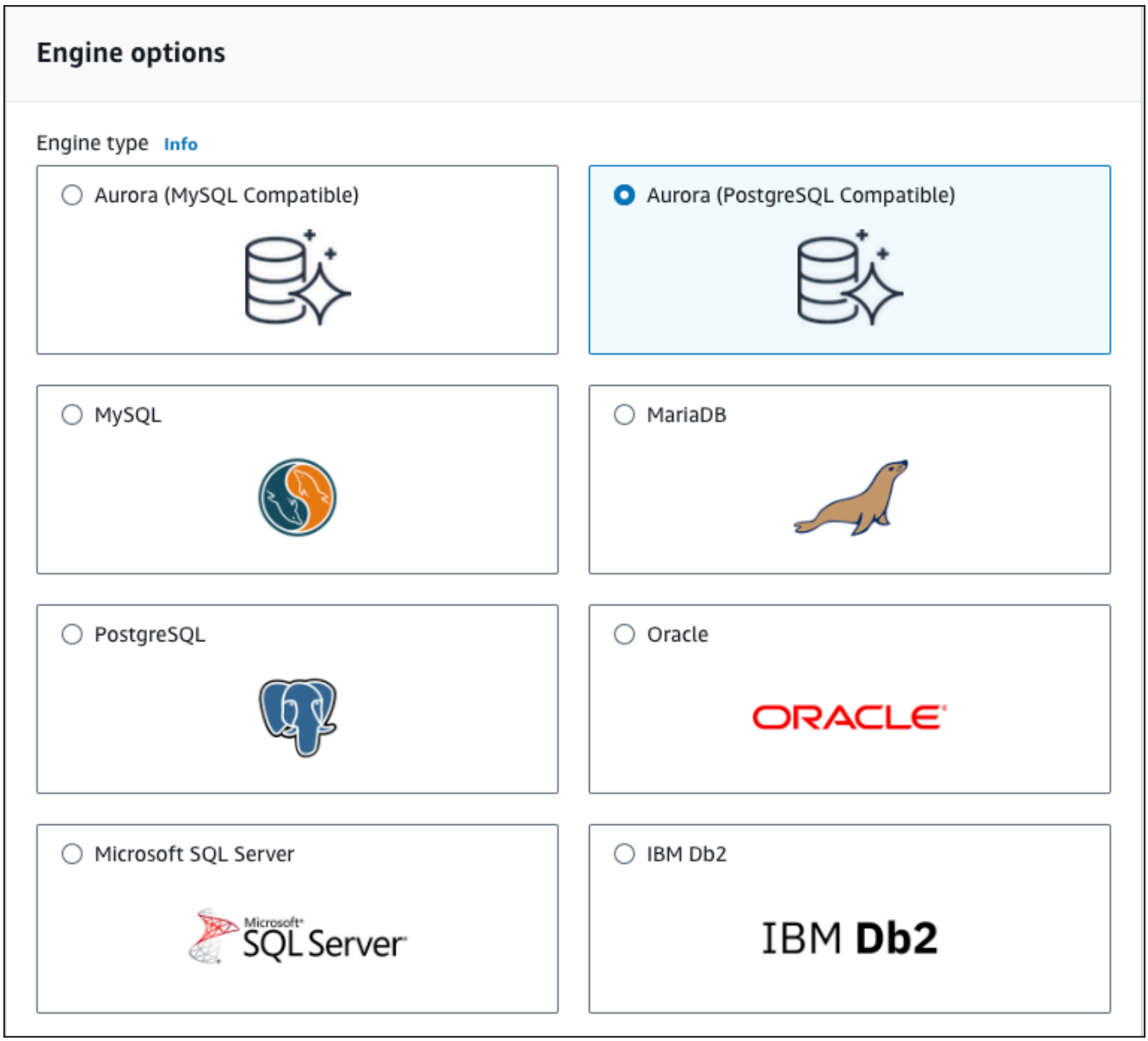
16. Completo Installazione di un server Web nell'istanza EC2.

Aurora PostgreSQL

Creazione di un cluster di database Aurora PostgreSQL

1. Accedi alla AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della AWS Management Console, assicurati che la Regione AWS sia identica a quella in cui hai creato l'istanza EC2.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).

5. Nella pagina Crea database scegli Creazione standard.
6. In Opzioni motore scegli Aurora (compatibile con PostgreSQL).

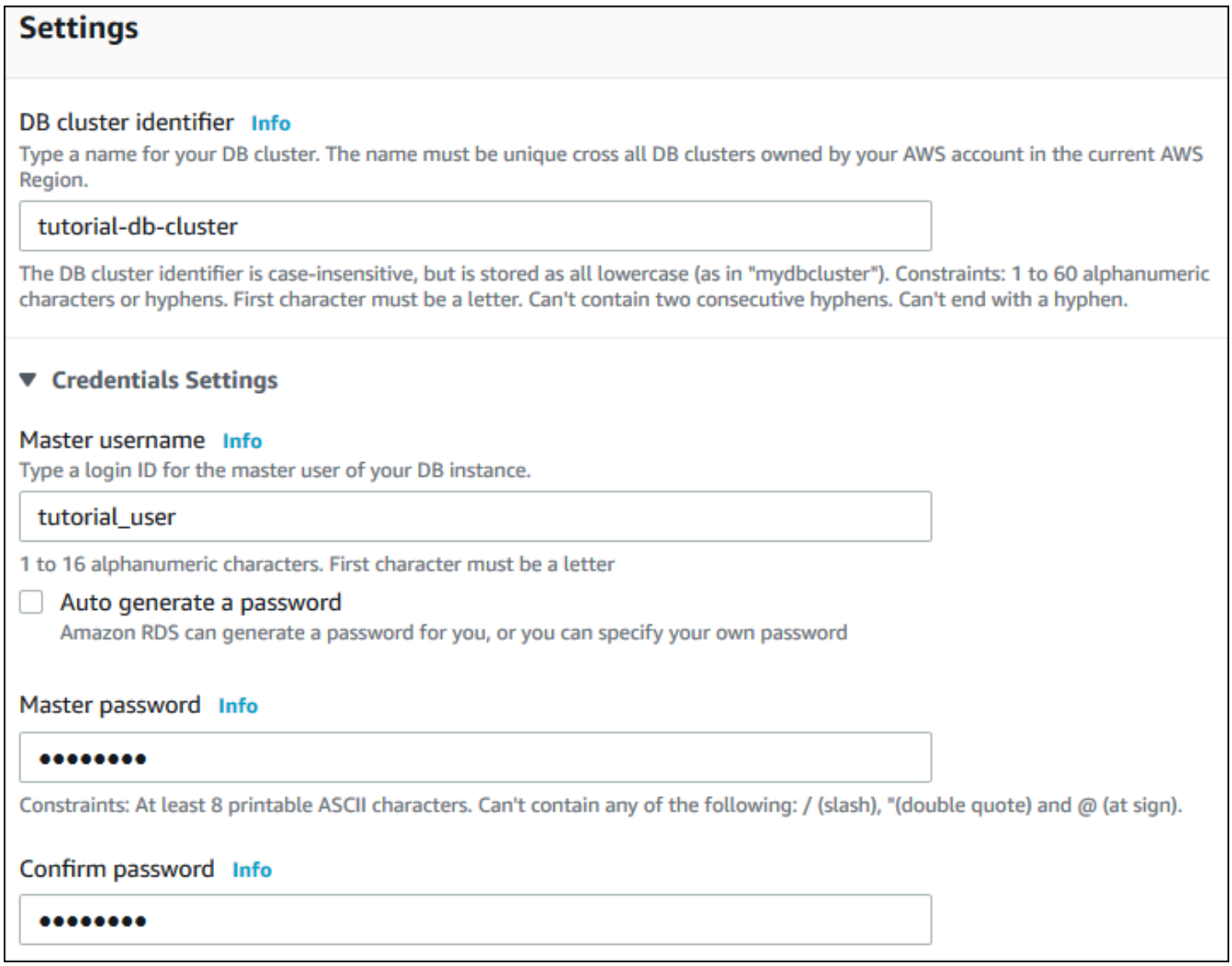


Mantenere i valori predefiniti per la versione e le altre opzioni del motore.

7. Nella sezione Templates (Modelli), selezionare Dev/Test.

8. Nella sezione Settings (Impostazioni) impostare questi valori:

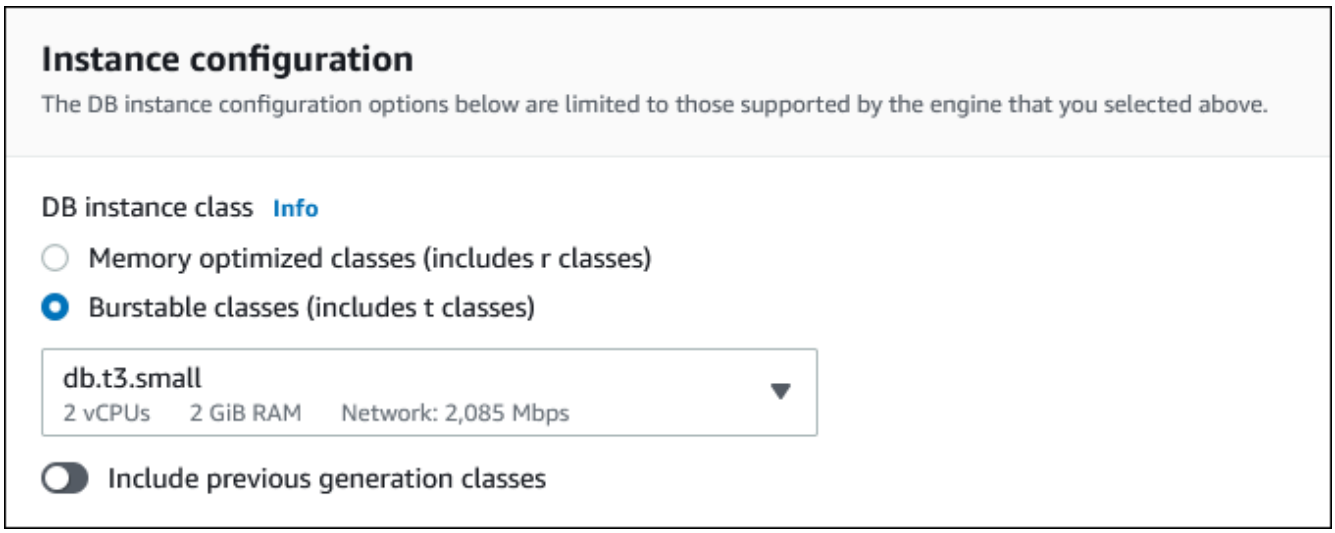
   - DB cluster identifier (Identificatore cluster DB): digita **`tutorial-db-cluster`**.
   - Master username (Nome utente master): digita **`tutorial_user`**.
   - Auto generate a password (Genera automaticamente una password): lascia l'opzione disattivata.
   - Master password (Password master): scegli una password.
   - Confirm password (Conferma la password): –digitare di nuovo la password.



9. Nella sezione Instance configuration (Configurazione dell'istanza), imposta i seguenti valori:

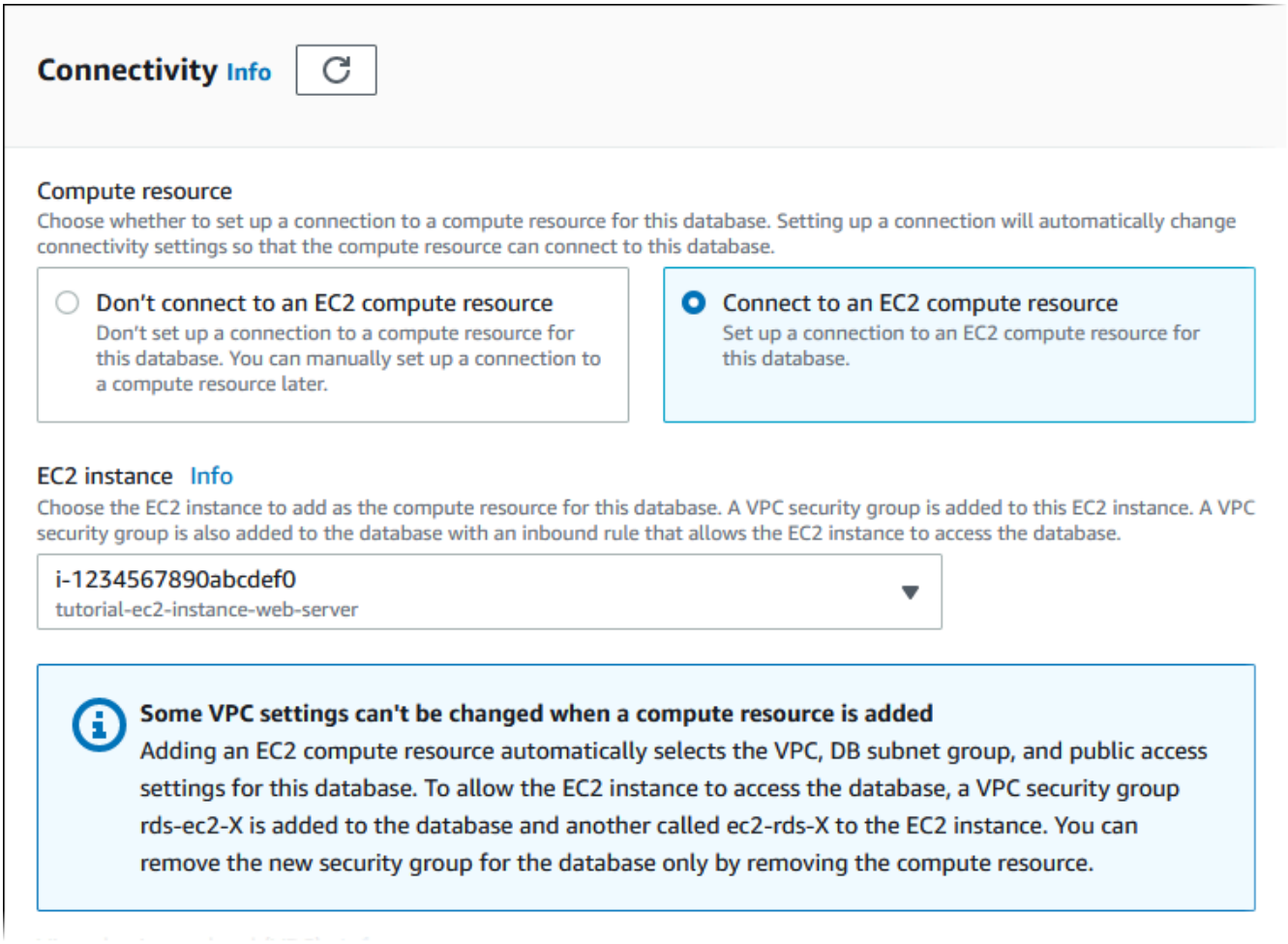
   - Classi espandibili (include le classi t)
   - db.t3.small o db.t3.medium

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.



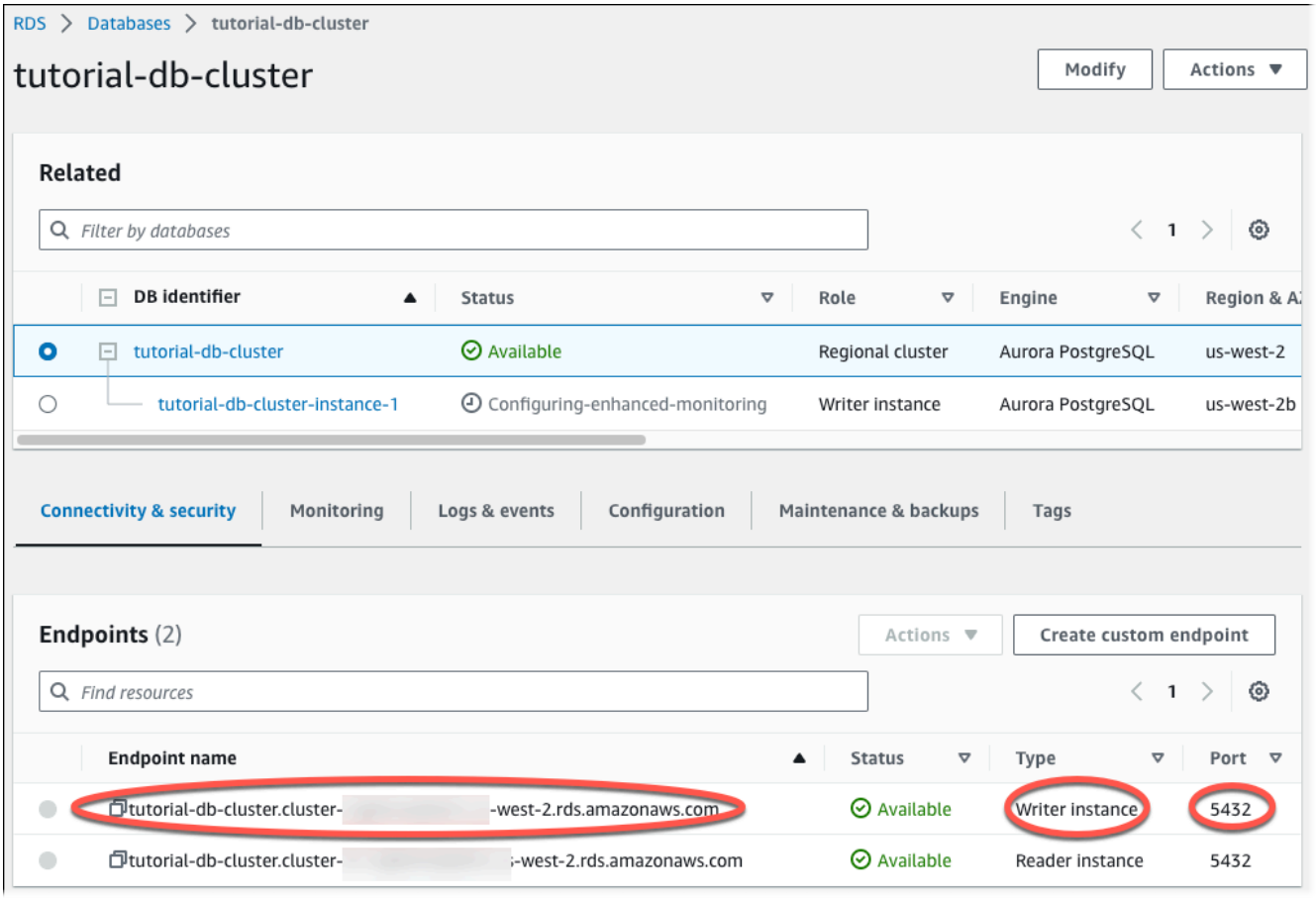
10. Nella sezione Availability and durability (Disponibilità e durata), utilizza i valori predefiniti.
11. Nella sezione Connectivity (Connettività), imposta i seguenti valori e lascia gli altri valori come predefiniti:

    - In Compute resource (Risorse di calcolo), seleziona Connect to an EC2 compute resource (Connetti a una risorsa di calcolo EC2).
    - Per l'istanza EC2, scegli l'istanza EC2 che hai creato in precedenza, ad esempio tutorial-ec2 -. instance-web-server



12. Aprire la sezione Additional configuration (Configurazione aggiuntiva) e specificare **`sample`** per Initial database name (Nome database iniziale). Lasciare le impostazioni predefinite per le altre opzioni.
13. Per creare il cluster database Aurora MySQL, scegli Crea database.

    Il nuovo cluster di database apparirà nell'elenco Database con lo stato Creazione in corso.
14. Attendere che lo Stato del nuovo cluster del database appaia come Disponibile. Quindi scegliere il nome del cluster del database per visualizzarne i dettagli.
15. Nella sezione Connettività e sicurezza, visualizzare Endpoint e Porta dell'istanza di scrittura del database.



Prendere nota dell'endpoint e della porta dell'istanza database writer. Queste informazioni verranno utilizzate per effettuare la connessione del server Web al cluster DB.

16. Completo Installazione di un server Web nell'istanza EC2.

Installazione di un server Web nell'istanza EC2

Installa un server Web in un'istanza EC2 creata in Avvia un'EC2istanza per connetterti con l'. Il server web si connette al cluster di database Amazon Aurora creato in Creazione di un cluster di database Amazon Aurora.

Installazione di un server Web Apache con PHP e MariaDB

Esegui la connessione all'istanza EC2 e installa il server Web Apache.

Per effettuare la connessione all'istanza EC2 e installare il server Web Apache con PHP.

1. Connettiti all'istanza EC2 che hai creato in precedenza seguendo i passaggi in Connetti alla tua istanza Linux nella Amazon EC2 User Guide.

Ti consigliamo di connetterti all'istanza EC2 tramite SSH. Se l'utilità client SSH è installata su Windows, Linux o Mac, puoi connetterti all'istanza utilizzando il comando nel seguente formato:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

Ad esempio, supponi che `ec2-database-connect-key-pair.pem` sia archiviato in `/dir1` su Linux e che il DNS IPv4 pubblico per l'istanza EC2 sia `ec2-12-345-678-90.compute-1.amazonaws.com`. Il comando SSH sarà simile al seguente:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
user@ec2-12-345-678-90.compute-1.amazonaws.com
```

2. Ottieni le ultime correzioni di bug e gli aggiornamenti di sicurezza aggiornando il software sulla tua istanza EC2. A questo scopo, eseguire il comando seguente.

> **Note**
>
> L'opzione `-y` installa gli aggiornamenti senza chiedere conferma. Per esaminare gli aggiornamenti prima di installarli, omettere questa opzione.

```
sudo dnf update -y
```

3. Al completamento degli aggiornamenti, installa il server Web Apache, PHP e il software MariaDB utilizzando i comandi seguenti. Con questo comando vengono installati contemporaneamente più pacchetti software e dipendenze correlate.

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

   PostgreSQL

   ```
   sudo dnf install -y httpd php php-pgsql postgresql15
   ```

Se si verifica un errore, è possibile che l'istanza non sia stata lanciata con un'AMI Amazon Linux 2023. Puoi invece utilizzare l'AMI Amazon Linux 2. È possibile visualizzare la versione di Amazon Linux con il comando seguente.

```
cat /etc/system-release
```

Per ulteriori informazioni, consulta la pagina relativa all'aggiornamento del software dell'istanza.

4. Avviare il server Web con il comando visualizzato di seguito.

```
sudo systemctl start httpd
```

È possibile verificare che il server Web sia installato e avviato correttamente. A tale scopo, immettere il nome DNS (Domain Name System) pubblico dell'istanza EC2 nella barra degli indirizzi di un browser Web, ad esempio: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. Se il server Web è in esecuzione, verrà visualizzata la pagina di test di Apache.

Se la pagina di test Apache non viene visualizzata, controllare le regole in entrata per il gruppo di sicurezza VPC creato in Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 . Assicurati che le regole in entrata ne includano una che consenta l'accesso HTTP (porta 80) per l'indirizzo IP utilizzato per connettersi al server Web.

> **Note**
>
> La pagina di test di Apache viene visualizzata solo quando la directory principale dei documenti è vuota, `/var/www/html`. Dopo aver aggiunto contenuti alla directory root dei documenti, i contenuti vengono visualizzati all'indirizzo DNS pubblico dell'istanza EC2. Prima di questo punto, vengono visualizzati nella pagina di test di Apache.

5. Configurare il server Web affinché si avvii a ogni avvio del sistema tramite il comando `systemctl`.

```
sudo systemctl enable httpd
```

Per permettere a ec2-user di gestire file nella directory principale predefinita del server Web Apache, è necessario modificare la proprietà e le autorizzazioni della directory /var/www. Sono disponibili molti modi per completare questa attività. In questo tutorial, aggiungi l'utente ec2-user al gruppo apache per assegnare la proprietà del gruppo apache della directory /var/www e assegnare autorizzazioni di scrittura al gruppo.

Per impostare le autorizzazioni dei file sul server Web Apache

1. Aggiungere l'utente ec2-user al gruppo apache.

   ```
   sudo usermod -a -G apache ec2-user
   ```

2. Per aggiornare le autorizzazioni e includere il nuovo gruppo apache, eseguire la disconnessione.

   ```
   exit
   ```

3. Effettuare nuovamente l'accesso e verificare che il gruppo apache esista mediante il comando groups.

   ```
   groups
   ```

   L'output avrà un aspetto simile al seguente:

   ```
   ec2-user adm wheel apache systemd-journal
   ```

4. Cambiare la proprietà del gruppo della directory /var/www e dei suoi contenuti sul gruppo apache.

   ```
   sudo chown -R ec2-user:apache /var/www
   ```

5. Cambiare le autorizzazioni della directory /var/www e delle sue sottodirectory per aggiungere le autorizzazioni di scrittura di gruppo e impostare l'ID di gruppo per le sottodirectory create in futuro.

   ```
   sudo chmod 2775 /var/www
   find /var/www -type d -exec sudo chmod 2775 {} \;
   ```

6. Cambiare le autorizzazioni in modo ricorsivo per i file nella directory /var/www e nelle sue sottodirectory per aggiungere le autorizzazioni di scrittura di gruppo.

```
find /var/www -type f -exec sudo chmod 0664 {} \;
```

Ora, `ec2-user` (e qualsiasi membro futuro del gruppo `apache`) può aggiungere, eliminare e modificare i file nella root del documento di Apache. Questo consente di aggiungere contenuti, ad esempio un sito Web statico o un'applicazione PHP.

> **Note**
>
> Un server Web che esegue il protocollo HTTP non offre alcuna sicurezza di trasporto per i dati inviati e ricevuti. Quando ti connetti a un server HTTP tramite un browser Web, la molte informazioni sono visibili a persone non autorizzate in qualsiasi punto del percorso di rete. Queste informazioni includono gli URL visitati, il contenuto delle pagine Web ricevute e i contenuti (incluse le password) di eventuali moduli HTML.
> La best practice per la protezione del tuo server Web prevede l'installazione del supporto per HTTPS (HTTP Secure). Questo protocollo protegge i dati con la crittografia SSL/TLS. Per ulteriori informazioni, consulta Esercitazione: Configurare SSL/TLS con l'AMI di Amazon Linux nella Guida per l'utente di Amazon EC2 .

Connessione del server web Apache al cluster di database

Successivamente, aggiungere contenuti al server Web Apache che effettua la connessione all'Amazon Aurora Cluster DB.

Per aggiungere contenuti al server Web Apache che effettua la connessione all'Cluster DB.

1. Mentre è ancora in corso la connessione all'istanza EC2, modificare la directory in `/var/www` e creare una nuova sottodirectory denominata `inc`.

   ```
   cd /var/www
   mkdir inc
   cd inc
   ```

2. Creare un nuovo file nella directory `inc` denominato `dbinfo.inc` e poi modificarlo con nano (o un altro editor a scelta).

   ```
   >dbinfo.inc
   ```

```
nano dbinfo.inc
```

3. Aggiungi i seguenti contenuti al file `dbinfo.inc`. Qui, *`db_instance_endpoint`* è l'endpoint dell'istanza di scrittura del cluster database, senza la porta, per il cluster database.

> **Note**
>
> Si consiglia di inserire le informazioni relative al nome utente e alla password in una cartella che non fa parte della directory principale del documento per il server Web. In questo modo si riduce la possibilità che le informazioni di sicurezza vengano esposte. Assicurati di modificare `master password` in una password adatta per la tua applicazione.

```
<?php

define('DB_SERVER', 'db_cluster_writer_endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');
?>
```

4. Salvare e chiudere il file `dbinfo.inc`. Se stai usando nano, salva e chiudi il file usando Ctrl+S e Ctrl+X.
5. Cambiare la directory in `/var/www/html`.

```
cd /var/www/html
```

6. Creare un nuovo file nella directory `html` denominato `SamplePage.php` e poi modificarlo con nano (o un altro editor a scelta).

```
>SamplePage.php
nano SamplePage.php
```

7. Aggiungere i seguenti contenuti al file `SamplePage.php`:

   MariaDB & MySQL

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
```

```
<body>
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
```

```
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";
```

```
   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
```

```
<?php

/* Connect to PostgreSQL and select the database. */
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
```

```
</form>
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
?>
</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
```

```
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;

}
?>
```

8. Salvare e chiudere il file `SamplePage.php`.
9. Verificare che il server Web effettui correttamente la connessione all'cluster DB aprendo un browser web e navigando fino a `http://`*`EC2 instance endpoint`*`/SamplePage.php`, ad esempio: `http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php`.

È possibile utilizzare `SamplePage.php` per aggiungere dati all'cluster DB. I dati aggiunti verranno visualizzati nella pagina. Per verificare che i dati siano stati inseriti nella tabella, installa il client MySQL nell'istanza Amazon EC2. Esegui quindi la connessione al cluster di database ed esegui la query sulla tabella.

Per ulteriori informazioni sulla connessione a un cluster DB, consultare Connessione a un cluster database Amazon Aurora.

Per assicurarsi che l'il cluster DB sia il più possibile sicura, verificare che le fonti esterne al VPC non possano connettersi all'Cluster DB.

Dopo aver terminato il test del server Web e del database, è necessario eliminare l'istanza il DB e l'istanza Amazon EC2.

- Per eliminare un cluster di database, segui le istruzioni in Eliminazione di cluster e istanze database di Aurora. Non è necessario creare uno snapshot finale.
- Per terminare un'istanza Amazon EC2, segui le istruzioni riportate in Termina istanza nella Guida per l'utente di Amazon EC2.

Tutorial di Amazon Aurora e codice di esempio

La AWS documentazione include diversi tutorial che ti guidano attraverso i casi d'uso più comuni di . Molti di questi tutorial mostrano come usare Amazon con altri servizi. AWS Inoltre, puoi accedere al codice di esempio in. GitHub

> **Note**
>
> Puoi trovare altri tutorial nel Blog di AWS Database. Per ulteriori informazioni sulla formazione, consulta AWS Training and Certification.

Argomenti

- Tutorial in questa guida
- Tutorial in altre guide AWS
- AWS portale di contenuti per workshop e laboratori per PostgreSQL
- AWS portale di contenuti per workshop e laboratori per MySQL
- Tutorial ed esempi di codice in GitHub
- Utilizzo di questo servizio con un AWS SDK

Tutorial in questa guida

I seguenti tutorial mostrano come eseguire le attività comuni con Amazon Aurora:

- Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4

  Scopri come includere un cluster di database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. In questo caso, il VPC condivide i dati con un server Web in esecuzione su un' EC2 istanza Amazon nello stesso VPC.
- Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

  Scopri come includere un cluster di database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. In questo caso, il VPC condivide i dati con EC2 un'istanza Amazon nello stesso VPC. In questo tutorial crei il VPC per questo scenario che funziona con un database in esecuzione in modalità dual-stack.

- Tutorial: creazione di un server Web e un cluster database Amazon Aurora

  Questo tutorial consente di installare un server Web Apache con PHP e creare un database MySQL. Il server Web viene eseguito su un' EC2 istanza Amazon che utilizza Amazon Linux e il database MySQL è . Sia l' EC2 istanza Amazon che il cluster di DB vengono eseguiti in un Amazon VPC.

- Tutorial: Ripristino di un cluster database Amazon Autora da uno snapshot cluster DB

  Scopri come ripristinare un cluster di database da uno snapshot di cluster di database.

- Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare

  Scopri come utilizzare i tag per specificare i cluster di database di Aurora da arrestare.

- Tutorial: registra le modifiche allo stato delle istanze DB utilizzando Amazon EventBridge

  Scopri come registrare una modifica dello stato di un'istanza DB utilizzando Amazon EventBridge e AWS Lambda.

Tutorial in altre guide AWS

I seguenti tutorial in altre AWS guide mostrano come eseguire attività comuni con Amazon :

> **Note**
>
> Alcuni tutorial utilizzano istanze database Amazon RDS, ma possono essere adattate per utilizzare cluster di database Aurora.

- Tutorial: Aurora Serverless nella Guida per gli sviluppatori di AWS AppSync

  Scopri come utilizzare per AWS AppSync fornire una fonte di dati per l'esecuzione di comandi SQL su cluster Aurora Serverless DB con l'API Data abilitata. È possibile utilizzare i resolver di AWS AppSync per eseguire le istruzioni SQL sull'API dati tramite query GraphQL, mutazioni e sottoscrizioni.

- Tutorial: Rotazione di un segreto per un AWS database nella guida per l'utente AWS Secrets Manager

Scopri come creare un segreto per un AWS database e configurare il segreto in modo che ruoti in base a una pianificazione. Attivi una rotazione manualmente e confermi che la nuova versione del segreto continua a fornire l'accesso.

- Tutorial ed esempi nella Guida per gli sviluppatori di AWS Elastic Beanstalk

  Scopri come distribuire applicazioni che utilizzano database Amazon RDS con. AWS Elastic Beanstalk

- Utilizzo dei dati da un database Amazon RDS per creare un'origine dati Amazon ML nella Amazon Machine Learning Developer Guide

  Scopri come creare un oggetto dell'origine dati Amazon Machine Learning (Amazon ML) dai dati memorizzati in un'istanza database MySQL.

- Abilitazione manuale dell'accesso a un'istanza Amazon RDS in un VPC nella Amazon QuickSight User Guide

  Scopri come abilitare QuickSight l'accesso a un'istanza database Amazon RDS in un VPC.

AWS portale di contenuti per workshop e laboratori per PostgreSQL

La seguente raccolta di workshop e altri contenuti pratici ti aiuta a comprendere le caratteristiche e le funzionalità di Amazon Aurora PostgreSQL:

- Creazione manuale di un nuovo cluster Aurora

  Informazioni su come creare manualmente un cluster Amazon Aurora PostgreSQL.

- Configurare Cloud9 e inizializzare il database

  Informazioni su come configurare Cloud9 e inizializzare il database PostgreSQL.

- Clonazione rapida

  Informazioni su come creare un clone rapido Aurora.

- Gestione del piano di query

  Informazioni su come controllare i piani di esecuzione per una serie di istruzioni tramite l gestione dei piani di query.

- Gestione della cache del cluster

Informazioni sulla funzionalità di gestione della cache del cluster in Aurora PostgreSQL.

- Streaming delle attività di database

  Informazioni su come monitorare e controllare l'attività del database con questa funzione.

- Uso di Approfondimenti sulle prestazioni

  Informazioni su come monitorare e ottimizzare la tua istanza DB tramite Approfondimenti sulle prestazioni.

- Monitoraggio delle prestazioni con strumenti RDS

  Scopri come utilizzare AWS gli strumenti Postgres (Cloudwatch, Enhanced Monitoring, Slow Query Logs, Performance Insights, PostgreSQL Catalog Views) per comprendere i problemi di prestazioni e identificare modi per migliorare le prestazioni del tuo database.

- Repliche di lettura con dimensionamento automatico

  Informazioni su come funziona il dimensionamento automatico delle repliche di lettura Aurora utilizzando uno script per la generazione di carichi.

- Test della tolleranza ai guasti

  Informazioni su come un cluster di database può gestire un errore.

- Database globale Aurora

  Informazioni su Database globale Amazon Aurora.

- Utilizzo del machine learning

  Informazioni sul machine learning Aurora.

- Aurora Serverless v2

  Informazioni su Aurora Serverless v2.

- Trusted Language Extensions per Aurora PostgreSQL

  Informazioni su come creare estensioni ad alte prestazioni che funzionano in sicurezza su Aurora PostgreSQL.

AWS portale di contenuti per workshop e laboratori per MySQL

La seguente raccolta di workshop e altri contenuti pratici ti aiuta a comprendere le caratteristiche e le funzionalità di Amazon Aurora MySQL:

- Creazione di un cluster Aurora

  Informazioni su come creare manualmente un cluster Amazon Aurora MySQL.
- Creazione di un ambiente IDE basato su Cloud9 per connettersi al database

  Informazioni su come configurare Cloud9 e inizializzare il database MySQL.
- Clonazione rapida

  Informazioni su come creare un clone rapido Aurora.
- Esecuzione del backtrack di un cluster

  Informazioni su come eseguire il backtrack di un cluster di database.
- Uso di Approfondimenti sulle prestazioni

  Informazioni su come monitorare e ottimizzare la tua istanza DB tramite Approfondimenti sulle prestazioni.
- Monitoraggio delle prestazioni con strumenti RDS

  Scopri come utilizzare AWS gli strumenti SQL per comprendere i problemi di prestazioni e identificare modi per migliorare le prestazioni del tuo database.
- Analisi delle prestazioni delle query

  Informazioni su come risolvere i problemi relativi alle prestazioni SQL utilizzando diversi strumenti.
- Repliche di lettura con dimensionamento automatico

  Informazioni su come funzionano le repliche di lettura con dimensionamento automatico.
- Test della tolleranza ai guasti

  Informazioni sulle funzionalità relative al disponibilità elevata e tolleranza ai guasti in Aurora MySQL.
- Database globale Aurora

  Informazioni su Database globale Amazon Aurora.

- Aurora Serverless v2

  Informazioni su Aurora Serverless v2.

- Utilizzo del machine learning

  Informazioni sul machine learning Aurora.

Tutorial ed esempi di codice in GitHub

I seguenti tutorial e codice di esempio GitHub mostrano come eseguire attività comuni con Amazon :

- Creazione di un libreria di prestiti Aurora Serverless v2

  Scopri come creare un'applicazione libreria di prestiti in cui i clienti possono prendere in prestito e restituire libri. L'esempio utilizza e. Aurora Serverless v2 AWS SDK per Python (Boto3)

- Creazione di un'applicazione di tracciamento degli articoli Amazon Aurora con un REST API Spring che esegue query dei dati Aurora Serverless v2 tramite SDK per Java 2.x

  Scopri come creare una REST API Spring che esegue query sui dati di Aurora Serverless v2. È destinato all'utilizzo da parte di un'applicazione React che usa SDK per Java 2.x.

- Creazione di un'applicazione di tracciamento degli articoli di Amazon Aurora che interroga i dati utilizzando Aurora Serverless v2AWS SDK per PHP

  Scopri come creare un'applicazione che utilizza `RdsDataClient` dell'API dati e Aurora Serverless v2 per monitorare e creare report degli elementi di lavoro. L'esempio utilizza. AWS SDK per PHP

- Creazione di un'applicazione di tracciamento degli articoli Amazon Aurora che esegue query dei dati Aurora Serverless v2 tramite AWS SDK per Python (Boto3)

  Scopri come creare un'applicazione che utilizza `RdsDataClient` dell'API dati e Aurora Serverless v2 per monitorare e creare report degli elementi di lavoro. L'esempio utilizza AWS SDK per Python (Boto3).

Utilizzo di questo servizio con un AWS SDK

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice, e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	Strumenti per esempi di PowerShell codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK for Rust	AWS SDK for Rust esempi di codice
SDK AWS per SAP ABAP	SDK AWS per SAP ABAP esempi di codice
SDK AWS per Swift	SDK AWS per Swift esempi di codice

Per esempi specifici del servizio, consulta Esempi di codice per l'utilizzo di Aurora AWS SDKs.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Provide feedback (Fornisci un feedback) nella parte inferiore di questa pagina.

Accesso programmatico ad

Amazon RDS fornisce i seguenti strumenti per gestire le risorse di Aurora in modo programmatico.

AWS Command Line Interface (AWS CLI)

Puoi creare e gestire le tue risorse Amazon Aurora utilizzando la shell AWS CLI a riga di comando. AWS CLI Fornisce l'accesso diretto APIs ai AWS quattro servizi, come AmazonRDS. Per la sintassi e gli esempi dei comandi per AmazonRDS, consulta rdsnel AWS CLI Command Reference.

AWS CloudFormation

Con questo strumento AWS Infrastructure as Code (IaC), puoi creare modelli che descrivono tutte le risorse Aurora che desideri e fornisce AWS CloudFormation e configura tali risorse per te. Per ulteriori informazioni, consulta the section called “Creazione di risorse con AWS CloudFormation”.

AWS kit di sviluppo software () SDKs

AWS fornisce SDKs molte tecnologie e linguaggi di programmazione popolari. Semplificano la chiamata ai AWS servizi dall'interno delle applicazioni in quel linguaggio o tecnologia. Per ulteriori informazioni al riguardoSDKs, consulta Strumenti per lo sviluppo e la gestione di applicazioni su AWS.

Amazon RDS API

Questa API è l'interfaccia a livello di protocollo per Amazon. RDS Quando si utilizza questa API opzione, è necessario formattare correttamente ogni HTTPS richiesta e aggiungere una firma digitale valida a ogni richiesta. Per ulteriori informazioni, consulta Documentazione di riferimento dell'API Amazon RDS.

Console-to-Code

Con questo strumento, puoi generare codice per le azioni che esegui nella RDS console Amazon e utilizzarlo in altri strumenti come AWS CloudFormation. Per ulteriori informazioni, consulta the section called “Console-to-Code”.

RDSDati API

RDSData API (DataAPI) fornisce l'accesso programmatico ai dati nei tuoi database Serverless Amazon Aurora. Con DataAPI, puoi utilizzare un HTTP endpoint sicuro per eseguire le tue SQL dichiarazioni senza gestire le connessioni. Per ulteriori informazioni, consulta Utilizzo dell'API dati Amazon RDS.

Console-to-CodeUtilizzalo per generare codice per le azioni della RDS console Amazon Aurora

La console fornisce un percorso guidato per creare risorse e testare prototipi. Se desideri creare le stesse risorse su larga scala, avrai bisogno del codice di automazione. Console-to-Codeè una funzionalità di Amazon Q Developer che può aiutarti a iniziare a usare il tuo codice di automazione. Console-to-Coderegistra le azioni della console, inclusi i valori predefiniti e i valori dei parametri che fornisci. Quindi utilizza l'intelligenza artificiale generativa per suggerire il codice nella lingua e nel formato preferiti per le azioni scelte. Poiché il flusso di lavoro della console assicura che i valori dei parametri specificati siano validi insieme, il codice generato utilizzando Console-to-Code ha valori di parametro compatibili. Puoi usare il codice come punto di partenza, personalizzandolo in seguito al fine di renderlo pronto per la produzione per il tuo caso d'uso specifico.

Ad esempio, con Console-to-Code, è possibile registrare la creazione di un cluster Aurora DB e scegliere di generare codice in AWS CloudFormation JSON formato. Quindi, puoi copiare quel codice e personalizzarlo per utilizzarlo nel tuo AWS CloudFormation modello.

Console-to-Code può attualmente generare infrastructure-as-code (iAc) nelle seguenti lingue e formati:

- CDKJava
- CDKPython
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

Per ulteriori informazioni e istruzioni sull'uso Console-to-Code, consulta Automating AWS services with Amazon Q Developer Console-to-Code nella Amazon Q Developer User Guide.

Configurazione di un cluster Amazon Aurora DB

Questa sezione illustra come configurare il cluster di database Aurora. Prima di creare un cluster di database Aurora, scegli la classe di istanze database che eseguirà il cluster di database. Inoltre, decidi dove verrà eseguito il cluster DB scegliendo una AWS regione. Quindi, crea il cluster di database. Se hai dei dati al di fuori di Aurora, puoi eseguirne la migrazione in un cluster di database Aurora.

Argomenti

- Creazione di un cluster database Amazon Aurora
- Creazione di RDS risorse Aurora con AWS CloudFormation
- Connessione a un cluster database Amazon Aurora
-
- Migrazione di dati a un cluster di database Amazon Aurora
- Creazione di una ElastiCache cache Amazon utilizzando le impostazioni dell'istanza Amazon DB del cluster Aurora RDS
- Migrazione automatica EC2 dei database su Amazon Aurora AmazonAWS Database Migration Service
- Tutorial: creazione di un cluster con un personalizzato e un nuovo gruppo di opzioni

Creazione di un cluster database Amazon Aurora

Un cluster di database Amazon Aurora è costituito da un'istanza database, compatibile con MySQL o PostgreSQL, e un volume cluster che conserva i dati per il cluster di database, di cui viene eseguita la copia su tre zone di disponibilità some singolo volume virtuale. Per impostazione predefinita, un cluster di database Aurora contiene un'istanza database primaria che esegue letture e scritture e, facoltativamente, fino a 15 repliche di Aurora (istanze database di scrittura). Per ulteriori informazioni sui cluster di database Aurora, consulta Cluster database Amazon Aurora.

Aurora dispone di due tipi principali di cluster database:

- Aurora con provisioning: la classe di istanza database per le istanze di scrittura e lettura viene scelta in base al carico di lavoro previsto. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora. Aurora con provisioning dispone di diverse opzioni, inclusi i database globali Aurora. Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.
- Aurora Serverless— Aurora Serverless v2 è una configurazione di ridimensionamento automatico su richiesta per Aurora. La capacità viene regolata automaticamente in base alle necessità dall'applicazione. L'addebito viene calcolato in base alle risorse utilizzate dal cluster database. Questa automazione è particolarmente utile per ambienti con carichi di lavoro estremamente variabili e imprevedibili. Per ulteriori informazioni, consulta Utilizzo Aurora Serverless v2.

Di seguito, troverai informazioni su come creare un cluster di database Aurora. Per iniziare, consulta innanzitutto Prerequisiti per i cluster di database.

Per istruzioni sulla connessione al cluster database Aurora, consulta Connessione a un cluster database Amazon Aurora.

Indice

- Prerequisiti per i cluster di database
  - Configurazione della rete per il cluster database
    - Configura la connettività di rete automatica con un'istanza EC2
    - Configurazione manuale della rete
  - Prerequisiti aggiuntivi
- Creazione di un cluster di database
  - Creazione di un'istanza DB primaria (writer)
- Impostazioni per cluster di database Aurora

- Impostazioni che non si applicano ad Amazon Aurora per i cluster di database
- Impostazioni che non si applicano alle istanze database Amazon Aurora

Prerequisiti per i cluster di database

> **Important**
>
> Prima di poter creare un cluster di database Aurora, devi completare le attività descritte in Configurazione dell'ambiente per Amazon Aurora.

Di seguito sono indicati i prerequisiti da completare prima di creare un cluster database.

Argomenti

- Configurazione della rete per il cluster database
- Prerequisiti aggiuntivi

Configurazione della rete per il cluster database

Puoi creare un cluster Amazon Aurora DB solo in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC, in un ambiente con almeno due zone di Regione AWS disponibilità. Il gruppo di sottoreti di database scelto per il cluster di database deve coprire almeno due zone di disponibilità. Questa configurazione assicura che nel cluster di database sia sempre disponibile almeno un'istanza database per il failover, nell'improbabile caso di errore in una zona.

Se prevedi di configurare la connettività tra il tuo nuovo cluster DB e un' EC2 istanza nello stesso VPC, puoi farlo durante la creazione del cluster DB. Se prevedi di connetterti al tuo cluster DB da risorse diverse dalle EC2 istanze nello stesso VPC, puoi configurare le connessioni di rete manualmente.

Argomenti

- Configura la connettività di rete automatica con un'istanza EC2
- Configurazione manuale della rete

Configura la connettività di rete automatica con un'istanza EC2

Quando crei un cluster Aurora DB, puoi utilizzarlo AWS Management Console per configurare la connettività tra un' EC2 istanza Amazon e il nuovo cluster DB. In questo caso, RDS configura automaticamente il VPC e le impostazioni di rete. Il cluster DB viene creato nello stesso VPC dell'istanza in modo che l' EC2 EC2 istanza possa accedere al cluster DB.

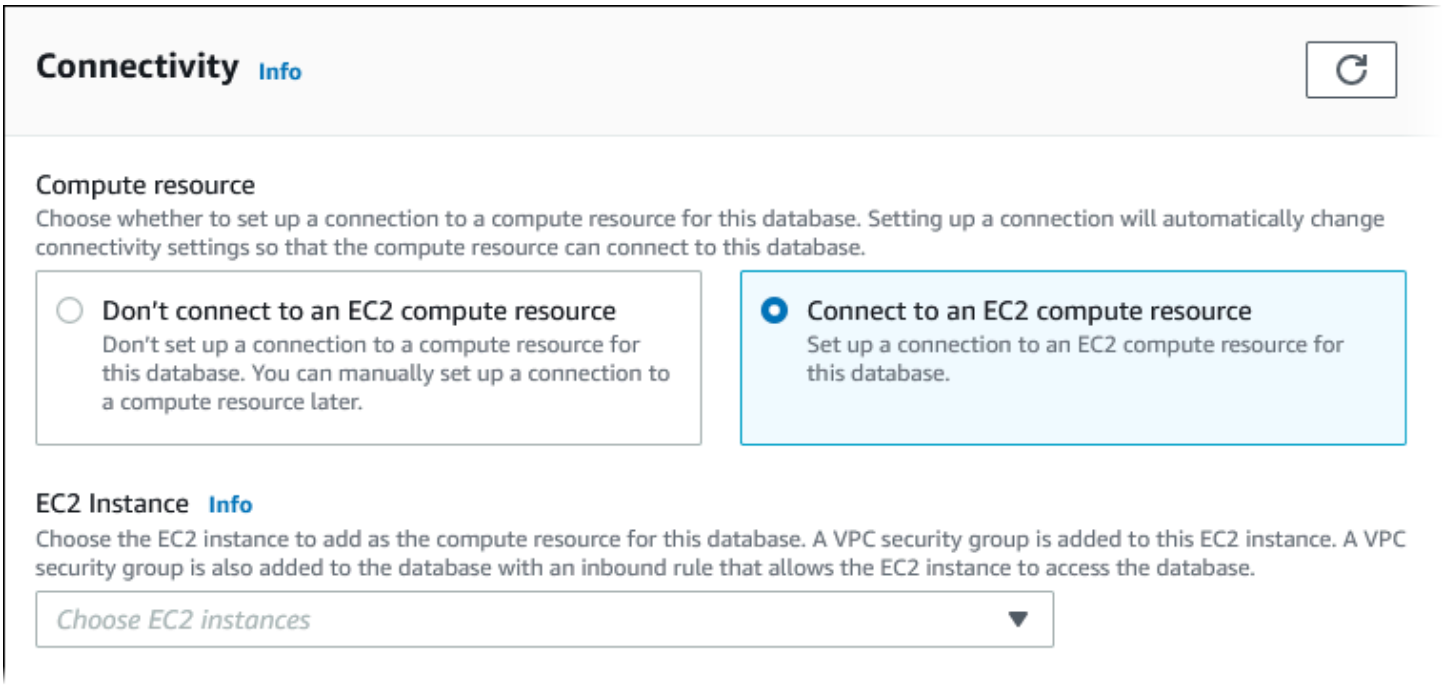
Di seguito sono riportati i requisiti per connettere un' EC2 istanza al cluster DB:

- L' EC2 istanza deve esistere Regione AWS prima di creare il cluster DB.

  Se non esistono EC2 istanze in Regione AWS, la console fornisce un collegamento per crearne una.
- Attualmente, il cluster database non può essere un cluster database Aurora Serverless o parte di un database globale Aurora.
- L'utente che sta creando l'istanza database deve disporre delle autorizzazioni per eseguire le seguenti operazioni:
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

L'utilizzo di questa opzione crea un cluster database privato. Il cluster di database utilizza un gruppo di sottoreti DB con solo sottoreti private per limitare l'accesso alle risorse all'interno del VPC.

Per connettere un' EC2 istanza al cluster DB, scegli Connetti a una risorsa di EC2 calcolo nella sezione Connettività della pagina Crea database.



Quando scegli Connetti a una risorsa di EC2 calcolo, RDS imposta automaticamente le seguenti opzioni. Non è possibile modificare queste impostazioni a meno che non si scelga di non configurare la connettività con un' EC2 istanza scegliendo Non connettersi a una risorsa di EC2 calcolo.

Opzione Console	Impostazione automatica
Tipo di rete	RDS imposta il tipo di rete su. IPv4 Attualmente, la modalità dual-stack non è supportata quando si configura una connessio ne tra un' EC2 istanza e il cluster DB.
Virtual Private Cloud (VPC)	RDS imposta il VPC su quello associato EC2 all'istanza.
DB subnet group (Gruppo di sottoreti DB)	RDS richiede un gruppo di sottoreti DB con una sottorete privata nella stessa zona di disponibilità dell'istanza. EC2 Se esiste un gruppo di sottoreti database che soddisfa questo requisito, RDS lo utilizza. Per impostazione predefinita, questa opzione è impostata su Automatic setup (Configurazione automatica). Quando si sceglie Automatic setup (Configurazione automatica) e non esiste un gruppo di sottoreti database che soddisfi questo

Opzione Console	Impostazione automatica
	requisito, viene eseguita la seguente procedura. RDS utilizza tre sottoreti private disponibili in tre zone di disponibilità in cui una delle zone di disponibilità è uguale all'istanza. EC2 Se una sottorete privata non è disponibile in una zona di disponibilità, RDS crea una sottorete privata nella zona di disponibilità. Quindi RDS crea il gruppo di sottoreti database. Quando è disponibile una sottorete privata, RDS utilizza la tabella di instradamento associata alla sottorete e aggiunge tutte le sottoreti create a questa tabella di instradamento. Quando una sottorete privata non è disponibile, RDS crea una tabella di instradamento senza accesso al gateway Internet e aggiunge le sottoreti create alla tabella di instradamento. RDS consente inoltre di utilizzare i gruppi di sottoreti database esistenti. Seleziona Choose existing (Scegli esistente) se desideri utilizzare un gruppo di sottoreti database esistente.
Accesso pubblico	RDS sceglie No in modo che il cluster di database non sia accessibile pubblicamente. Per motivi di sicurezza, come best practice si consiglia di mantenere il database privato e accertarsi che non sia accessibi le da Internet.

Opzione Console	Impostazione automatica
VPC security group (firewall) (Gruppo di sicurezza VPC (firewall)	RDS crea un nuovo gruppo di sicurezza associato al cluster di database. Il gruppo di sicurezza è denominato `rds-ec2-`*n*, dove *n* è un numero. Questo gruppo di sicurezza include una regola in entrata con il gruppo di sicurezza EC2 VPC (firewall) come origine. Questo gruppo di sicurezza associato al cluster DB consente all' EC2 istanza di accedere al cluster DB. RDS crea anche un nuovo gruppo di sicurezza associato all' EC2 istanza. Il gruppo di sicurezza è denominato `ec2-rds-`*n*, dove *n* è un numero. Questo gruppo di sicurezza include una regola in uscita con il gruppo di sicurezza VPC del cluster di database come origine. Questo gruppo di sicurezza consente all' EC2 istanza di inviare traffico al cluster DB. Puoi aggiungere un nuovo gruppo di sicurezza aggiuntivo scegliendo Create nuovo (Crea nuovo) e digitando il nome del nuovo gruppo di sicurezza. Puoi aggiungere gruppi di sicurezza esistenti scegliendo Choose existing (Scegli esistente) e selezionando i gruppi di sicurezza da aggiungere.
Zona di disponibilità	Quando non si crea una replica Aurora in Disponibilità e durabilità durante la creazione di cluster DB (implementazione Single-AZ), RDS sceglie la zona di disponibilità dell'istanza. EC2 Quando si crea una replica Aurora durante la creazione di un cluster DB (implementazione Multi-AZ), RDS sceglie la zona di disponibilità dell' EC2 istanza per un'istanza DB nel cluster DB. RDS sceglie casualmente una zona di disponibilità diversa per l'altra istanza database nel cluster database. L'istanza DB principale o la replica Aurora vengono create nella stessa zona di disponibilità dell'istanza. EC2 Esiste la possibilità di incrociare i costi della zona di disponibilità se l'istanza e EC2 l'istanza DB principale si trovano in zone di disponibilità diverse.

Per ulteriori informazioni su queste impostazioni, consultare Impostazioni per cluster di database Aurora.

Se si apportano modifiche a queste impostazioni dopo la creazione del cluster DB, le modifiche potrebbero influire sulla connessione tra l' EC2 istanza e il cluster di database.

Configurazione manuale della rete

Se prevedi di connetterti al tuo cluster DB da risorse diverse dalle EC2 istanze nello stesso VPC, puoi configurare le connessioni di rete manualmente. Se utilizzi la AWS Management Console per creare il cluster database, puoi configurare Amazon RDS per creare automaticamente un VPC. Altrimenti, puoi utilizzare un VPC esistente o crearne uno nuovo per il tuo cluster di database Aurora. Qualunque sia l'approccio scelto, il VPC deve disporre di almeno una sottorete in ciascuna di almeno due zone di disponibilità affinché tu possa utilizzarlo con un cluster di database Amazon Aurora.

Per impostazione predefinita, Amazon RDS crea automaticamente l'istanza database primaria e la replica di Aurora nelle zone di disponibilità. Per scegliere una zona di disponibilità specifica, l'impostazione di implementazione Multi-AZ Availability & durability (Disponibilità e durata) in Don't create an Aurora Replica (Non creare una replica di Aurora) deve essere modificata. Questo espone un'impostazione Availability Zone (Zona di disponibilità) che consente di scegliere tra le zone di disponibilità nel VPC. Tuttavia, è consigliabile mantenere le impostazioni predefinite e consentire ad Amazon RDS di creare un'implementazione Multi-AZ e scegliere automaticamente le zone di disponibilità. In questo modo, il cluster di database Aurora viene creato con funzionalità di failover veloce e alta disponibilità che sono due dei principali vantaggi di Aurora.

Se non si dispone di un VPC predefinito o uno non è stato creato, è possibile che un VPC venga creato automaticamente da Amazon RDS durante la creazione di un cluster database mediante la console. In caso contrario, devi completare le attività seguenti:

- Crea un VPC con almeno una sottorete in ciascuna delle almeno due zone di disponibilità nel luogo in Regione AWS cui desideri implementare il cluster DB. Per ulteriori informazioni, consultare Uso di un cluster database in un VPC e Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 .
- Specificare un gruppo di sicurezza VPC che autorizzi le connessioni al proprio cluster di database . Per ulteriori informazioni, consultare Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza e Controllo dell'accesso con i gruppi di sicurezza.
- Specifica di un gruppo di sottoreti del database RDS che definisca almeno due sottoreti nel VPC che possono essere utilizzate dal cluster di database . Per ulteriori informazioni, consulta Utilizzo di gruppi di sottoreti database.

Per informazioni su VPCs, vedere. Amazon VPC e Aurora Per un tutorial di configurazione della rete per un cluster database privato, consulta Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 .

Se desideri connetterti a una risorsa che non si trova nello stesso VPC del cluster Aurora DB, consulta gli scenari appropriati descritti in Scenari per l'accesso a un cluster di DB in un VPC.

Prerequisiti aggiuntivi

Prima di creare il cluster database, considera i seguenti prerequisiti aggiuntivi:

- Se ti connetti AWS utilizzando credenziali AWS Identity and Access Management (IAM), il tuo AWS account deve disporre di politiche IAM che concedano le autorizzazioni necessarie per eseguire le operazioni di Amazon RDS. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.

  Se utilizzi IAM per accedere alla console Amazon RDS, devi prima accedere AWS Management Console con le tue credenziali utente. Quindi vai alla console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
- Se desideri personalizzare i parametri di configurazione per il cluster di database, devi specificare un gruppo di parametri del cluster di database e un gruppo di parametri database con le impostazioni dei parametri richieste. Per informazioni sulla creazione o la modifica di un gruppo di parametri del cluster di database o un gruppo di parametri database, consulta .
- Determina il numero di porta TCP/IP da specificare per il cluster di database. I firewall presso alcune aziende bloccano le connessioni alle porte predefinite (3306 per MySQL, 5432 per PostgreSQL) per Aurora. Se il firewall della tua azienda blocca la porta predefinita, scegli un'altra porta per il cluster di database. Tutte le istanze in un cluster di database utilizzano la stessa porta.
- Se la versione principale del motore per il database ha raggiunto la data di fine del supporto standard RDS, è necessario utilizzare l'opzione Extended Support CLI o il parametro API RDS. Per ulteriori informazioni, vedere RDS Extended Support inImpostazioni per cluster di database Aurora.

Creazione di un cluster di database

È possibile creare un cluster Aurora DB utilizzando l'API AWS Management Console AWS CLI, the o RDS.

Console

È possibile creare un cluster DB utilizzando Easy create abilitato o non abilitato. AWS Management Console Con l'opzione Easy create (Creazione rapida) attivata, specifichi solo il tipo di motore del database, la dimensione dell'istanza di database e l'identificatore dell'istanza di database. Easy create (Creazione rapida) utilizza l'impostazione predefinita per altre opzioni di configurazione. Con l'opzione Easy create (Creazione rapida) disattivata, specifichi più opzioni di configurazione quando crei un database, comprese quelle relative a disponibilità, sicurezza, backup e manutenzione.

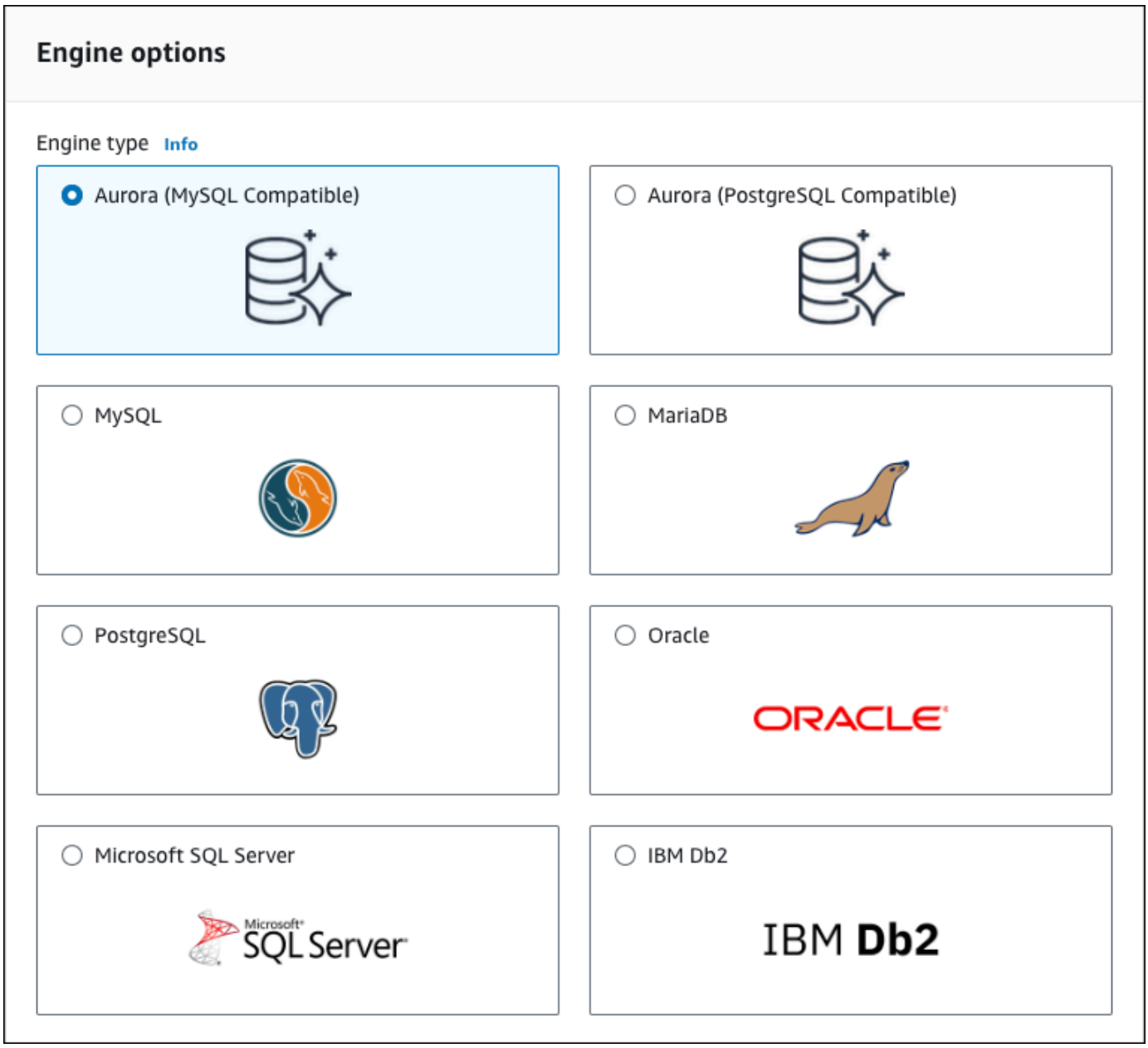
> **Note**
>
> Per questo esempio, Standard create (Creazione standard) è attivato e Easy create (Creazione rapida) non è abilitata. Per informazioni sulla creazione di un cluster DB con Easy create abilitato, vedereNozioni di base su Amazon Aurora.

Per creare un cluster di database Aurora tramite la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli la AWS regione in cui desideri creare il cluster DB.

   Aurora non è disponibile in tutte le AWS regioni. Per un elenco delle AWS regioni in cui Aurora è disponibile, consulta. Disponibilità nelle regioni
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).
5. Per Scegli un metodo di creazione del database seleziona Creazione standard.
6. Per Tipo di motore scegli una delle seguenti opzioni:

   - Aurora (compatibile con MySQL)
   - Aurora (compatibile con PostgreSQL)



7. Scegli la Versione del motore.

   Per ulteriori informazioni, consulta Versioni di Amazon Aurora. Puoi utilizzare i filtri per scegliere versioni compatibili con le funzionalità che desideri, ad esempio Aurora Serverless v2. Per ulteriori informazioni, consulta Utilizzo Aurora Serverless v2.

8. In Templates (Modelli), selezionare il modello che corrisponde al proprio caso d'uso.

9. Per inserire la password principale, procedere come segue:

   a. Nella sezione Impostazioni espandi Impostazioni delle credenziali.

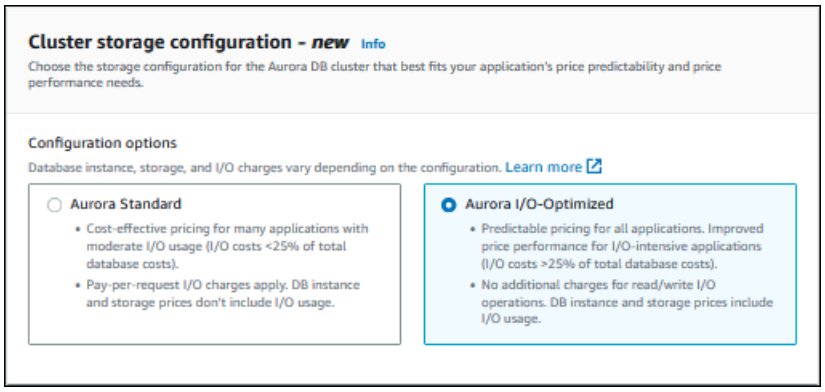
b. Deselezionare la casella di controllo Auto generate a password (Genera automaticamente una password).

c. (Opzionale) Modificare il valore Master username (Nome utente principale) e inserire la stessa password in Master password (Password principale) e Confirm password (Conferma password).

Per impostazione predefinita, la nuova istanza database utilizza una password generata automaticamente per l'utente principale.

10. Nella sezione Connettività in Gruppo di sicurezza VPC (firewall), se selezioni Crea nuovo, viene creato un gruppo di sicurezza VPC con una regola in entrata che consente all'indirizzo IP del computer locale di accedere al database.

11. In Configurazione archiviazione cluster, scegliere Aurora I/O-Optimized o Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.



12. (Facoltativo) Configura una connessione a una risorsa di calcolo per questo cluster di database.

Puoi configurare la connettività tra un' EC2 istanza Amazon e il nuovo cluster DB durante la creazione del cluster DB. Per ulteriori informazioni, consulta Configura la connettività di rete automatica con un'istanza EC2 .

13. Per le restanti sezioni, specifica le impostazioni del cluster di database. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

14. Scegliere Create database (Crea database).

Se scegli di utilizzare una password generata in modo automatico, il pulsante View credential details (Vedi dettagli delle credenziali) appare nella paginaDatabases.

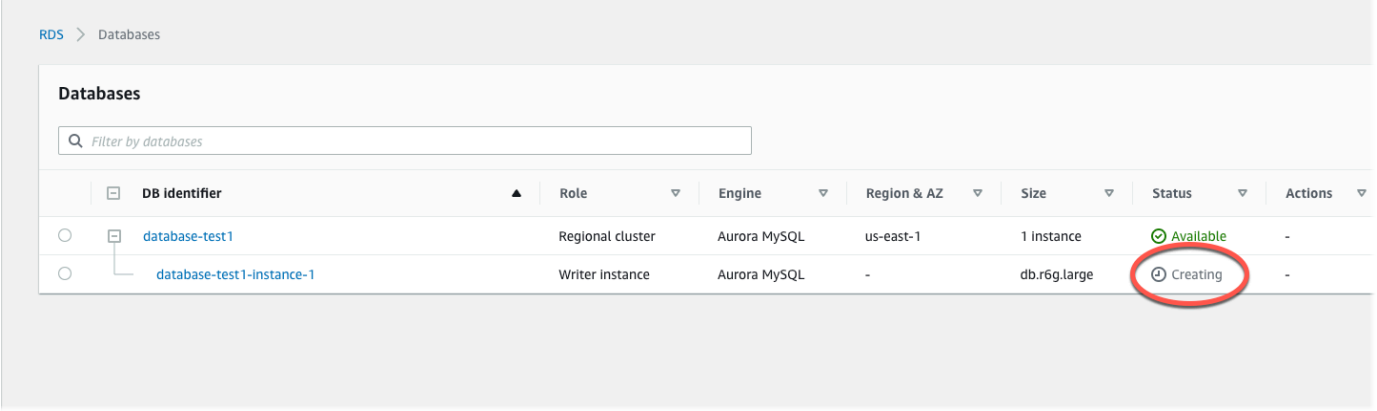
Per vedere nome utente e password per il cluster di database, seleziona View credential details (Vedi dettagli delle credenziali).

Per connetterti all'istanza database come utente principale, utilizza il nome utente e la password visualizzati.

> **Important**
>
> Non potrai visualizzare di nuovo la password dell'utente principale. Se non la registri, potresti doverla modificare. Se devi modificare la password dell'utente principale dopo che l'istanza database è disponibile, puoi modificare l'istanza database per eseguire tale operazione. Per ulteriori informazioni sulla modifica di un'istanza database , consulta Modifica di un cluster database Amazon Aurora.

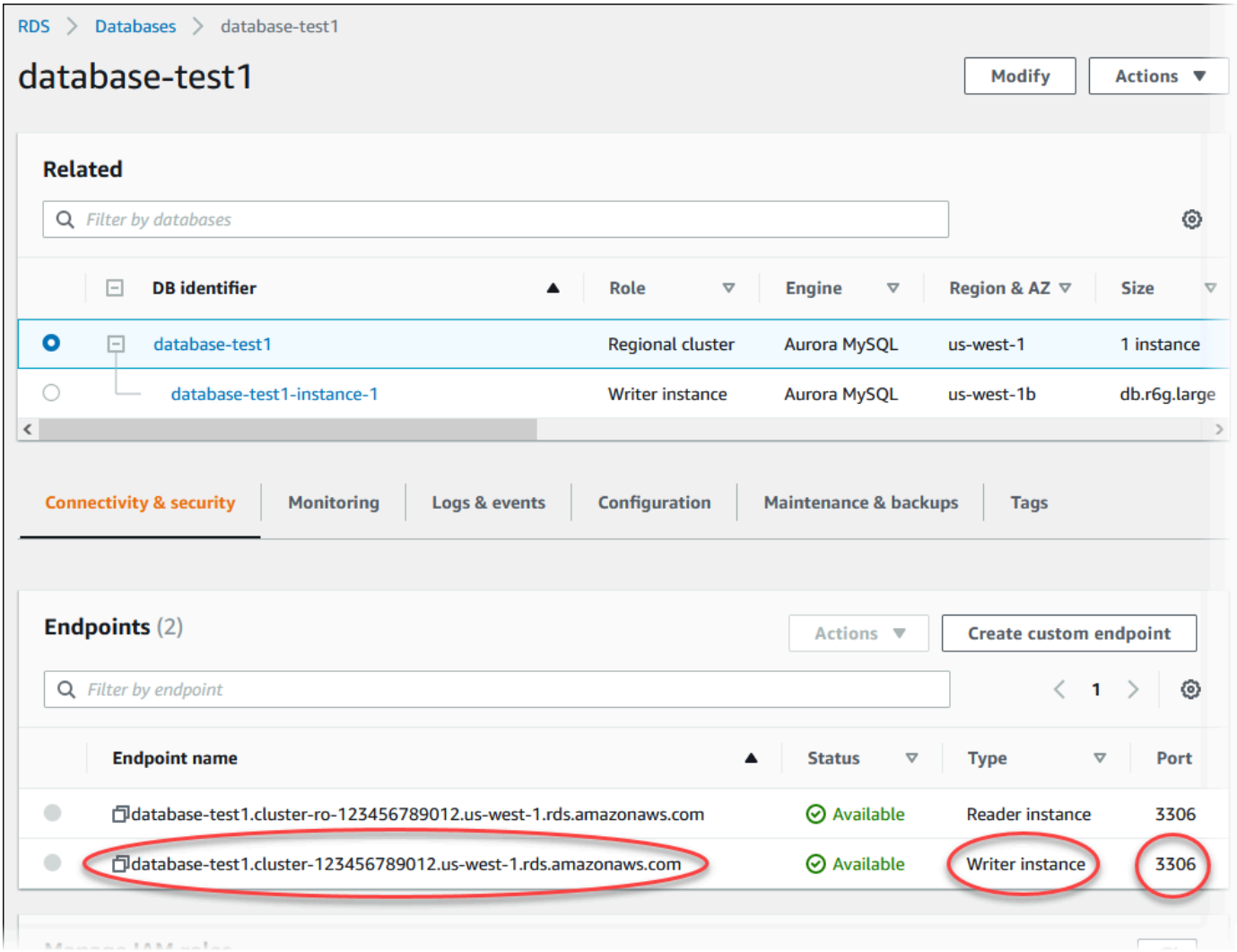
15. Per Databases, seleziona il nome del nuovo cluster di database Aurora.

Nella console RDS vengono visualizzati i dettagli per il nuovo cluster di database. Lo stato del cluster database e della sua istanza database sarà creating (in creazione) fino a quando sarà pronto per essere impiegato.



Quando lo stato cambia in available (disponibile) è possibile connettersi al cluster di database. A seconda della classe di istanza database e della quantità di storage, prima che il nuovo cluster di database sia disponibile possono trascorrere fino a 20 minuti.

Per visualizzare il cluster appena creato, scegli Databases (Database) dal riquadro di navigazione nella console Amazon RDS. Quindi scegli il cluster di database per mostrare i dettagli del cluster di database. Per ulteriori informazioni, consulta Visualizzazione di un cluster di database Amazon Aurora.



Nella scheda Connectivity & security (Connettività e sicurezza), prendi nota della porta e dell'endpoint dell'istanza database di scrittura. Utilizzare l'endpoint e la porta del cluster nelle stringhe di connessione JDBC e ODBC per un'applicazione che esegue operazioni di scrittura o lettura.

AWS CLI

> **Note**
>
> Prima di poter creare un cluster Aurora DB utilizzando, è necessario soddisfare i prerequisiti richiesti AWS CLI, come la creazione di un VPC e di un gruppo di sottoreti DB RDS. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

È possibile utilizzarlo AWS CLI per creare un cluster Aurora MySQL DB o un cluster Aurora PostgreSQL DB.

Per creare un cluster Aurora MySQL DB utilizzando il AWS CLI

Quando si crea un'istanza database o un cluster di database Aurora MySQL compatibile con 8.0 o 5.7, è necessario specificare `aurora-mysql` per l'opzione `--engine`.

Completa questa procedura:

1. Identifica il gruppo di sottorete DB e l'ID del gruppo di sicurezza VPC per il nuovo cluster DB, quindi chiama create-db-cluster AWS CLI il comando per creare il cluster Aurora MySQL DB.

   Il comando seguente crea ad esempio un nuovo cluster di database –compatibile con MySQL 8.0 denominato `sample-cluster`. Il cluster utilizza la versione del motore predefinita e il tipo di archiviazione Aurora I/O-Optimized.

   Per, o: Linux macOS Unix

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-mysql --engine-version 8.0 \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Per Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-mysql --engine-version 8.0 ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Il comando seguente crea un nuovo cluster di database –compatibile con MySQL 5.7 denominato `sample-cluster`. Il cluster utilizza la versione del motore predefinita e il tipo di archiviazione Aurora Standard.

   Per LinuxmacOS, oUnix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster  \
       --engine aurora-mysql --engine-version 5.7 \
   ```

```
    --storage-type aurora \
    --master-username user-name --manage-master-user-password \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

Per Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster sample-cluster  ^
    --engine aurora-mysql --engine-version 5.7 ^
    --storage-type aurora ^
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Se si utilizza la console per creare un cluster di database, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza il AWS CLI per creare un cluster DB, è necessario creare in modo esplicito l'istanza principale per il cluster DB. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

   Chiama il create-db-instance AWS CLI comando per creare l'istanza principale per il tuo cluster DB. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

   > **Note**
   >
   > Non è possibile impostare l'opzione `--storage-type` per le istanze database. È possibile impostarla solo per i cluster database.

   Il comando seguente crea ad esempio una nuova istanza database –compatibile con MySQL 5.7 o con MySQL 8.0 denominata `sample-instance`.

   Per LinuxmacOS, oUnix:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance \
        --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
   class db.r5.large
   ```

   Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Per creare un cluster Aurora PostgreSQL DB utilizzando il AWS CLI

1. Identifica il sottogruppo DB e l'ID del gruppo di sicurezza VPC per il tuo nuovo cluster DB, quindi chiama create-db-cluster AWS CLI il comando per creare il cluster Aurora PostgreSQL DB.

   Il comando seguente crea ad esempio un nuovo cluster di database denominato `sample-cluster`. Il cluster utilizza la versione del motore predefinita e il tipo di archiviazione Aurora I/O-Optimized.

   Per, o: Linux macOS Unix

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-postgresql \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Per Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-postgresql ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

2. Se si utilizza la console per creare un cluster di database, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza il AWS CLI per creare un cluster DB, è necessario creare in modo esplicito l'istanza principale per il cluster DB. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

   Chiama il create-db-instance AWS CLI comando per creare l'istanza principale per il tuo cluster DB. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

Per LinuxmacOS, oUnix:

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
instance-class db.r5.large
```

Per Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
instance-class db.r5.large
```

In questi esempi è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

API RDS

> **Note**
>
> Prima di poter creare un cluster Aurora DB utilizzando, è necessario soddisfare i prerequisiti richiesti AWS CLI, come la creazione di un VPC e di un gruppo di sottoreti DB RDS. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Identifica il gruppo di sottorete DB e l'ID del gruppo di sicurezza VPC per il nuovo cluster DB, quindi richiamate l'operazione DBCluster Create per creare il cluster DB.

Quando si crea un'istanza database o un cluster database Aurora MySQL versione 2 o 3, specificare `aurora-mysql` per il parametro `Engine`.

Quando si crea un'istanza database o un cluster di database Aurora PostgreSQL, specificare `aurora-postgresql` per il parametro `Engine`.

Se si utilizza la console per creare un cluster di database, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza l'API RDS per creare un cluster DB, è necessario creare in modo esplicito l'istanza principale per il cluster DB utilizzando Create.

DBInstance L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

Creazione di un'istanza DB primaria (writer)

Se utilizzi il AWS Management Console per creare un cluster DB, Amazon RDS crea automaticamente l'istanza principale (writer) per il tuo cluster DB. Se utilizzi l'API AWS CLI o RDS per creare un cluster DB, devi creare in modo esplicito l'istanza principale per il tuo cluster DB. L'istanza primaria è la prima istanza creata in un cluster di database. Se non crei l'istanza database primaria, lo stato degli endpoint del cluster database rimane impostato su `Creating`.

Per ulteriori informazioni, consulta Creazione di un cluster di database.

> **Note**
>
> Se disponi di un cluster DB senza un'istanza Writer DB, chiamato anche cluster headless, non puoi utilizzare la console per creare un'istanza di writer. È necessario utilizzare l'API AWS CLI o RDS.

L'esempio seguente utilizza il create-db-instance AWS CLI comando per creare un'istanza writer per un cluster Aurora PostgreSQL DB denominato. `headless-test`

```
aws rds create-db-instance \
    --db-instance-identifier no-longer-headless \
    --db-cluster-identifier headless-test \
    --engine aurora-postgresql \
    --db-instance-class db.t4g.medium
```

Impostazioni per cluster di database Aurora

La tabella seguente contiene informazioni dettagliate sulle impostazioni disponibili quando crei un cluster di database Aurora.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Auto minor version upgrade (Aggiorna	Selezionare Enable auto minor version upgrade (Abilita aggiornam	Imposta questo valore per ogni istanza database nel cluster

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
mento automatic o della versione secondaria)	enti automatici versioni secondari e) affinché il cluster database Aurora riceva automaticamente gli aggiornamenti secondari preferiti della versione nel motore di database non appena diventano disponibili. L'impostazione Auto Minor Version Upgrade (Aggiornamento automatico minore della versione) si applica ai cluster di database Aurora PostgreSQL e Aurora MySQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora PostgreSQL, consultar e Aggiornamenti del motore di database per Amazon Aurora Postgre SQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore di database per Amazon Aurora My SQL.	Aurora. Se un'istanza database del cluster ha questa impostazi one disattivata, il cluster non viene aggiornato automaticamente. Utilizzando AWS CLI, esegui `create-db-instance` e imposta l'opzione. `--auto-mi nor-version-upgrade|-- no-auto-minor-version-u pgrade` Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `AutoMinor VersionUpgrade` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
AWS KMS key	Disponibile solo se Encryption (Crittografia) è impostato su Enable encryption (Abilita crittografia). Scegliere la AWS KMS key da utilizzare per crittografare questo cluster DB. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--kms-key-id` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `KmsKeyId`.
Backtrack	Si applica solo ad Aurora MySQL. Selezionare Enable Backtrack (Abilita backtrack) o Disable Backtrack (Disabilita backtrack) per abilitare o disabilitare il backtrack, rispettivamente. Utilizzando il backtrack, sarà possibile riportare il cluster di database a un momento specifico, senza creare un nuovo cluster di database. Questa opzione è disabilitata per impostazione predefinita. Se si abilita il backtrack, sarà necessario anche specificare l'intervallo di tempo per il quale si desidera eseguire il backtrack del cluster di database (finestra target di backtrack). Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--backtrack-window` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `BacktrackWindow` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Autorità di certificazione	L'autorità di certificazione (CA) per il certificato del server utilizzato dalle istanze database nel cluster database. Per ulteriori informazioni, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.	Utilizzando AWS CLI, esegui `create-db-instance` e imposta l'`--ca-certificate-identifier` opzione. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `CACertificateIdentifier` .
Configurazione dell'archiviazione del cluster	Tipo di archiviazione per il cluster database: Aurora I/O-Optimized o Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--storage-type` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `StorageType` .
Copy tags to snapshots (Copia tag in snapshot)	Selezionare questa opzione per copiare i tag dell'istanza database in una snapshot database quando si crea una snapshot. Per ulteriori informazioni, consulta Etichettatura di Amazon Aurora e risorse Amazon RDS.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--copy-tags-to-snapshot | --no-copy-tags-to-snapshot` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `CopyTagsToSnapshot` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Database authentication (Autenticazione del database)	L'autenticazione del database da utilizzare. Per MySQL: • Scegliere Password authentication (Autenticazione tramite password) per autenticare gli utenti di database solo con le password del database. • Seleziona Autenticazione database tramite password e IAM per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli IAM. Per ulteriori informazioni, consulta Autenticazione del database IAM . Per PostgreSQL: • Scegli IAM database authentication (Autenticazione database IAM) per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli. Per ulteriori informazioni, consulta Autenticazione del database IAM . • Seleziona Autenticazione Kerberos per autenticare le password del database e le credenziali utente utilizzando	Per utilizzare l'autenticazione del database IAM con AWS CLI, esegui `create-db-cluster` e imposta l'`--enable-iam-database-authentication | --no-enable-iam-database-authentication` opzione. Per utilizzare l'autenticazione database IAM con l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EnableIAMDatabaseAuthentication` . Per utilizzare l'autenticazione Kerberos con AWS CLI, esegui `create-db-cluster` e imposta le opzioni `--domain` and`--domain-iam-role-name` . Per utilizzare l'autenticazione Kerberos con l'API RDS, chiama `CreateDBCluster` e imposta i parametri `Domain` e `DomainIAMRoleName` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	l'autenticazione Kerberos. Per ulteriori informazioni, consulta Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL.	
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. Per impostazione predefinita, i cluster di database Aurora MySQL sono impostati sulla porta MySQL predefinita 3306 e i cluster di database Aurora PostgreSQL sulla porta PostgreSQL 5432. I firewall presso alcune aziende bloccano le connessioni di tali porte predefinite. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster di database.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--port`opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `Port`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
DB Cluster Identifier (Identificatore cluster DB)	Inserisci un nome per il tuo cluster DB che sia unico per il tuo account nella AWS regione che hai scelto. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint del cluster per il cluster database. Per informazioni sull'endpoint del cluster, consulta Connessioni endpoint Amazon Aurora. L'identificatore del cluster DB prevede i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoco per tutti i cluster DB per AWS account e per AWS regione.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--db-cluster-identifier` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DBClusterIdentifier` .
DB cluster parameter group (Gruppo di parametri del cluster database)	Scegliere gruppo di parametri del cluster database. Puoi utilizzare il gruppo di parametri del cluster database predefinito di Aurora oppure puoi creare il tuo gruppo di parametri. Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta .	Usando AWS CLI, `create-db-cluster`esegui e imposta l'`--db-cluster-parameter-group-name` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DBClusterParameterGroupName` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
DB instance class (Classe istanza database)	Si applica solo al tipo di Provisioned Capacity (Capacità assegnata): Scegliere una classe di istanza database che definisca i requisiti di elaborazione e di memoria per ogni istanza nel cluster di database. Per altre informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora.	Imposta questo valore per ogni istanza database nel cluster Aurora. Usando AWS CLI, `create-db-instance` esegui e imposta l'`--db-instance-class` opzione. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `DBInstanceClass`.
DB parameter group (Gruppo di parametri database)	Scegliere un gruppo di parametri. Puoi utilizzare il gruppo di parametri predefinito di Aurora oppure puoi creare il tuo gruppo di parametri personalizzato. Per ulteriori informazioni sui gruppi di parametri, consultare .	Imposta questo valore per ogni istanza database nel cluster Aurora. Usando AWS CLI, `create-db-instance` esegui e imposta l'`--db-parameter-group-name`opzione. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `DBParameterGroupName`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
DB subnet group (Gruppo di sottoreti DB)	Il gruppo di sottoreti database da utilizzare per il cluster di database. Seleziona Choose existing (Scegli esistente) per utilizzare un gruppo di sottoreti database esistente. Quindi scegli il gruppo di sottoreti richiesto dall'elenco a discesa Existing DB subnet groups (Gruppi di sottoreti DB esistenti). Scegli Automatic setup (Configurazione automatica) per consentire a RDS di selezionare un gruppo di sottoreti database compatibile. Se non ne esiste uno, RDS crea un nuovo gruppo di sottoreti per il cluster. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--db-subnet-group-name` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DBSubnetGroupName`.
Enable deletion protection (Abilita protezione da eliminazione)	Selezionare Enable deletion protection (Abilita protezione da eliminazione) per impedire l'eliminazione del cluster di database. Se si crea un cluster di database di produzione con la console, la protezione da eliminazione è abilitata per impostazione predefinita.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--deletion-protection | --no-deletion-protection` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DeletionProtection`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Enable Encryption (Abilita crittografia)	Scegliere `Enable encryption` per abilitare la crittografia a riposo per questo cluster di database. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--storage-encrypted | --no-storage-encrypted` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `StorageEncrypted` .
Enable Enhanced Monitoring (Abilita il monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.	Impostare questi valori per ogni istanza database nel cluster Aurora. Utilizzando AWS CLI, esegui `create-db-instance` e imposta le `--monitoring-role-arn` opzioni `--monitoring-interval` and. Tramite l'API RDS, chiama `CreateDBInstance` e imposta i parametri `MonitoringInterval` e `MonitoringRoleArn` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Abilita l'API RDS Data	Scegli Abilita RDS Data API per abilitare RDS Data API (Data API). Data API fornisce un endpoint HTTP sicuro per l'esecuzione di istruzioni SQL senza gestire le connessioni. Per ulteriori informazioni, consulta Utilizzo dell'API dati Amazon RDS.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--enable-http-endpoint | --no-enable-http-endpoint` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EnableHttpEndpoint`.
Tipo di motore	Scegli il motore di database da utilizzare per il cluster di database.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--engine`opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `Engine`.
Versione del motore	Si applica solo al tipo di Provisioned Capacity (Capacità assegnata): Selezionare il numero di versione del motore del database.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--engine-version` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EngineVersion`.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Failover priority (Priorità failover)	Scegliere una priorità di failover per l'istanza. Se non specifich i alcun valore, l'impostazione predefinita è tier-1. Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazi oni, consulta Tolleranza ai guasti di un cluster DB Aurora.	Imposta questo valore per ogni istanza database nel cluster Aurora. Usando AWS CLI, `create-db-instance` esegui e imposta l'`--promotion-tier` opzione. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `PromotionTier` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Initial database name (Nome del database iniziale)	Immettere un nome per il database predefinito. Se non si specifica un nome per il cluster database Aurora MySQL, Amazon RDS non crea un database nel cluster database che si sta creando. Se non si fornisce un nome per un cluster database Aurora PostgreSQL, Amazon RDS crea un database denominato `postgres`. Per Aurora MySQL, il nome del database predefinito presenta questi vincoli: • Deve contenere da 1 a 64 caratteri alfanumerici. • Non può essere una parola riservata del motore di database. Per Aurora PostgreSQL, il nome del database predefinito presenta questi vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici. • Deve iniziare con una lettera. I caratteri successivi possono essere lettere, trattini bassi o cifre (da 0 a 9). • Non può essere una parola riservata del motore di database.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--database-name` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `DatabaseName` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	Per creare database aggiuntivi, eseguire la connessione al cluster di database e utilizzare il comando SQL CREATE DATABASE. Per ulteriori informazioni sulla connessione al cluster di database, consulta Connessione a un cluster database Amazon Aurora.	
Log exports (Esportazioni log)	Nella sezione Esportazioni di log, scegli i log che desideri iniziare a pubblicare su Amazon CloudWatch Logs. Per ulteriori informazioni sulla pubblicazione dei log di Aurora MySQL in Logs, vedere. CloudWatch Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch Per ulteriori informazioni sulla pubblicazione dei log di Aurora PostgreSQL in Logs, vedere. CloudWatch Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs CloudWatch	Utilizzando, esegui e imposta l' AWS CLI opzione. `create-db-cluster --enable-cloudwatch-logs-exports` Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EnableCloudwatchLogsExports` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Maintenance window (Finestra di manutenzione)	Selezionare Select window (Seleziona finestra) e specifica l'intervallo temporale settimanale durante il quale può avvenire la manutenzione del sistema. Oppure, selezionare No preference (Nessuna preferenza) per consentire a Amazon RDS di assegnare un periodo casuale.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--preferred-maintenance-window` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `PreferredMaintenanceWindow` .
Gestisci le credenziali principali in AWS Secrets Manager	Seleziona Manage master credentials in AWS Secrets Manager (Gestione credenziali master in AWS Secrets Manager) per gestire la password dell'utente master in un segreto di Secrets Manager. Facoltativamente, scegli la chiave KMS da utilizzare per proteggere il segreto. Scegliere tra le chiavi KMS presenti nell'account o inserire la chiave da un altro account. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta le `--master-user-secret-kms-key-id` opzioni `--manage-master-user-password | --no-manage-master-user-password` and. Tramite l'API RDS, chiama `CreateDBCluster` e imposta i parametri `ManageMasterUserPassword` e `MasterUserSecretKmsKeyId` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Master password (Password master)	Immettere una password per accedere al cluster DB: • Per Aurora MySQL, la password deve contenere da 8 a 41 caratteri ASCII stampabili. • Per Aurora PostgreSQL, deve contenere da 8 a 99 caratteri ASCII stampabili. • Non può contenere /, ", @ o uno spazio.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--master-user-password` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `MasterUserPassword` .
Master username (Nome utente master)	Immettere un nome da utilizzare come nome utente master per accedere al cluster DB: • Per Aurora MySQL, il nome deve contenere da 1 a 16 caratteri alfanumerici. • Per Aurora PostgreSQL deve contenere da 1 a 63 caratteri alfanumerici. • Il primo carattere deve essere una lettera. • Il nome non può essere una parola riservata del motore di database. Dopo la creazione del cluster database, il nome utente master non può essere modificato.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--master-username` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `MasterUsername` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Multi-AZ deployment (Implementazione Multi-AZ)	Si applica solo al tipo di Provisioned Capacity (Capacità assegnata): Determinare se si intende creare repliche di Aurora in altre zone di disponibilità ai fini del supporto per il failover. Se si seleziona Create Replica in Different Zone (Crea replica in zona diversa), Amazon RDS crea una replica di Aurora nel cluster database in una zona di disponibilità differente rispetto all'istanza principale del cluster di database. Per altre informazioni sulle zone di disponibilità multiple, consulta Regioni e zone di disponibilità.	Usando AWS CLI, `create-db-cluster` esegui e imposta l'`--availability-zones` opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `AvailabilityZones` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Tipo di rete	I protocolli di indirizzo IP supportati dal cluster database. IPv4per specificare che le risorse possono comunicare con il cluster DB solo tramite il protocollo di IPv4 indirizzamento. Modalità dual-stack per specificare che le risorse possono comunicar e con il cluster DB tramite o IPv4 entrambi IPv6. Utilizza la modalità dual-stack se disponi di risorse che devono comunicare con il cluster DB tramite il protocollo di indirizza mento. IPv6 Per utilizzare la modalità dual-stack, assicurati che almeno due sottoreti si estendano su due zone di disponibilità che supportino sia il protocollo di rete che il protocollo di rete. IPv4 IPv6 Inoltre, assicuratevi di associare un blocco IPv6 CIDR alle sottoreti del gruppo di sottoreti DB specifica to. Per ulteriori informazioni, consulta Assegnazione di indirizzi IP in Amazon Aurora.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'opzione. `-network-type` Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `NetworkType` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Accesso pubblico	Scegli Publicly accessible (Accessibile pubblicamente) per assegnare al cluster DB un indirizzo IP pubblico oppure scegli Not publicly accessible (Non accessibile pubblicamente). Le istanze nel cluster database possono essere una combinazione di istanze database pubbliche e private. Per ulteriori informazioni su come non consentire l'accesso pubblico per le istanze, consulta Nascondere cluster database in un VPC da Internet. Per connettersi a un'istanza DB dall'esterno Amazon VPC, l'istanza DB deve essere accessibile pubblicamente, l'accesso deve essere concesso utilizzando le regole in ingresso del gruppo di sicurezza dell'istanza DB e devono essere soddisfatti altri requisiti. Per ulteriori informazioni, consulta Impossibile connettersi all'istanza database di Amazon RDS. Se la tua istanza DB non è accessibile al pubblico, puoi anche utilizzare una connessione AWS Site-to-Site VPN o una AWS Direct Connect connessione per accedervi da una rete privata. Per	Imposta questo valore per ogni istanza database nel cluster Aurora. Utilizzando AWS CLI, esegui `create-db-instance` e imposta l'`--publicly-accessible | --no-publicly-accessible` opzione. Tramite l'API RDS, chiama `CreateDBInstance` e imposta il parametro `PubliclyAccessible` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	ulteriori informazioni, consulta Riservatezza del traffico Internet.	
Supporto esteso RDS	Seleziona Enable RDS Extended Support per consentire alle versioni dei motori principal i supportate di continuare a funzionare oltre la data di fine del supporto standard di Aurora. Quando crei un cluster DB, Amazon Aurora utilizza per impostazione predefinita RDS Extended Support. Per impedire la creazione di un nuovo cluster DB dopo la data di fine del supporto standard di Aurora e per evitare addebiti per RDS Extended Support, disabilita questa impostazione. I cluster DB esistenti non verranno addebitati fino alla data di inizio dei prezzi di RDS Extended Support. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con .	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'opzione. `--engine-lifecycle-support` Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `EngineLifecycleSupport` .

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Server proxy per RDS	Scegli Create an RDS Proxy (Crea un server proxy per RDS) per creare un proxy per il tuo cluster database. Amazon RDS crea automaticamente per il proxy un ruolo IAM e un segreto in Secrets Manager. Per ulteriori informazioni, consulta Proxy Amazon RDS per Aurora.	Non disponibile durante la creazione di un cluster database.
Periodo di conservazione	Selezionare l'intervallo di tempo, da 1 a 35 giorni, per il quale Aurora conserva le copie di backup del database. Le copie di Backup possono essere utilizzate per il point-in-time ripristino (PITR) del database fino al secondo.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--backup-retention-period`opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `BackupRetentionPeriod` .
Attiva DevOps Guru	Scegli Attiva DevOps Guru per attivare Amazon DevOps Guru per il tuo database Aurora. DevOpsAffinché Guru for RDS fornisca un'analisi dettagliata delle anomalie delle prestazioni, è necessario attivare Performance Insights. Per ulteriori informazioni, consulta Configurare Guru per DevOps RDS.	Puoi attivare DevOps Guru for RDS dalla console RDS, ma non utilizzando l'API RDS o la CLI. Per ulteriori informazioni sull'attivazione di DevOps Guru, consulta la Amazon DevOps Guru User Guide.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
Attiva Performance Insights	Scegli Turn on Performance Insights (Attiva Performance Insights) per attivare Amazon RDS Performance Insights. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.	Impostare questi valori per ogni istanza database nel cluster Aurora. Tramite la AWS CLI, esegui `create-db-instance` e imposta le opzioni `--enable-performance-insights` | `--no-enable-performance-insights`, `--performance-insights-kms-key-id` e `--performance-insights-retention-period`. Tramite l'API RDS, chiama `CreateDBInstance` e imposta i parametri `EnablePerformanceInsights`, `PerformanceInsightsKMSKeyId` e `PerformanceInsightsRetentionPeriod`.
Virtual Private Cloud (VPC)	Scegliere il VPC che ospita il cluster DB. Scegliere Create a new VPC (Crea un nuovo VPC) per fare in modo che Amazon RDS crei automaticamente un VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.	Per l'API AWS CLI and, si specifica il gruppo di sicurezza VPC. IDs

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
VPC security group (firewall) (Gruppo di sicurezza VPC (firewall))	Scegli Create new (Crea nuovo) per fare in modo che Amazon RDS crei un gruppo di sicurezza VPC. Altrimenti, seleziona Choose existing (Seleziona esistenti) e specifica uno o più gruppi di sicurezza VPC per proteggere l'accesso di rete al cluster di database. Quando scegli Create new (Crea nuovo) nella console RDS, viene creato un nuovo gruppo di sicurezza con una regola in entrata che consente l'accesso all'istanza database dall'indirizzo IP rilevato nel browser. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.	Utilizzando AWS CLI, esegui `create-db-cluster` e imposta l'`--vpc-security-group-ids`opzione. Tramite l'API RDS, chiama `CreateDBCluster` e imposta il parametro `VpcSecurityGroupIds` .

Impostazioni che non si applicano ad Amazon Aurora per i cluster di database

Le seguenti impostazioni nel AWS CLI comando `create-db-cluster`e nell'operazione dell'API RDS `CreateDBCluster`non si applicano ai cluster Amazon Aurora DB.

> **Note**
>
> AWS Management Console non mostra queste impostazioni per i cluster Aurora DB.

AWS CLI impostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`
`--publicly-accessible | --no-publicly-accessible`	`PubliclyAccessible`

Impostazioni che non si applicano alle istanze database Amazon Aurora

Le seguenti impostazioni nel AWS CLI comando `create-db-instance`e nell'operazione dell'API RDS `CreateDBInstance`non si applicano alle istanze DB del cluster Amazon Aurora DB.

> **Note**
>
> AWS Management Console non mostra queste impostazioni per le istanze Aurora DB.

AWS CLI impostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--availability-zone`	`AvailabilityZone`
`--backup-retention-period`	`BackupRetentionPeriod`
`--backup-target`	`BackupTarget`
`--character-set-name`	`CharacterSetName`
`--character-set-name`	`CharacterSetName`
`--custom-iam-instance-profile`	`CustomIamInstanceProfile`
`--db-security-groups`	`DBSecurityGroups`
`--deletion-protection | --no-deletion-protection`	`DeletionProtection`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--enable-cloudwatch-logs-exports`	`EnableCloudwatchLogsExports`
`--enable-customer-owned-ip | --no-enable-customer-owned-ip`	`EnableCustomerOwnedIp`
`--enable-iam-database-authentication | --no-enable-iam-database-authentication`	`EnableIAMDatabaseAuthentication`
`--engine-version`	`EngineVersion`
`--iops`	`Iops`
`--kms-key-id`	`KmsKeyId`
`--master-username`	`MasterUsername`

AWS CLI impostazione	Impostazione API RDS
`--master-user-password`	`MasterUserPassword`
`--max-allocated-storage`	`MaxAllocatedStorage`
`--multi-az | --no-multi-az`	`MultiAZ`
`--nchar-character-set-name`	`NcharCharacterSetName`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--preferred-backup-window`	`PreferredBackupWindow`
`--processor-features`	`ProcessorFeatures`
`--storage-encrypted | --no-storage-encrypted`	`StorageEncrypted`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--timezone`	`Timezone`
`--vpc-security-group-ids`	`VpcSecurityGroupIds`

Creazione di RDS risorse Aurora con AWS CloudFormation

Aurora è integrato con AWS CloudFormation un servizio che ti aiuta a modellare e configurare AWS le tue risorse in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. Crei un modello che descrive tutte le AWS risorse che desideri di parametri di cluster DB), e fornisce e AWS CloudFormation configura tali risorse per te.

Quando lo usi AWS CloudFormation, puoi riutilizzare il modello per configurare le risorse Aurora in modo coerente e ripetuto. Descrivi le tue risorse una sola volta, quindi fornisci le stesse risorse più e più volte in più AWS account e regioni.

Aurora e modelli AWS CloudFormation

AWS CloudFormation i modelli sono file di testo formattati in JSON o. YAML Questi modelli descrivono le risorse che desideri inserire negli AWS CloudFormation stack. Se non conosci JSON oYAML, puoi usare AWS CloudFormation Designer per iniziare a usare i AWS CloudFormation modelli. Per ulteriori informazioni, consulta Che cos'è AWS CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation .

Aurora supporta la creazione di risorse in. AWS CloudFormation Per ulteriori informazioni, inclusi esempi JSON e YAML modelli per queste risorse, consulta il riferimento ai tipi di RDS risorse nella Guida per l'AWS CloudFormation utente.

Scopri di più su AWS CloudFormation

Per ulteriori informazioni AWS CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- AWS CloudFormation Guida per l'utente
- AWS CloudFormation APIRiferimento
- AWS CloudFormation Guida per l'utente dell'interfaccia a riga di comando

Connessione a un cluster database Amazon Aurora

Puoi eseguire la connessione a un cluster di database Aurora utilizzando gli stessi strumenti che utilizzi per connetterti a un database MySQL o PostgreSQL. Devi specificare una stringa di connessione con qualsiasi script, utility o applicazione che si connette a un'istanza di database MySQL o PostgreSQL. Utilizzi la stessa chiave pubblica per le connessioni Secure Sockets Layer (SSL).

Nella stringa di connessione, in genere utilizzerai le informazioni sulla porta e sull'host dall'endpoint speciali associate al cluster di database. Con questi endpoint, puoi utilizzare gli stessi parametri di connessione indipendentemente dal numero di istanze di database presenti nel cluster. Per attività specializzate come la risoluzione dei problemi, puoi anche utilizzare le informazioni sull'host e sulla porta da un'istanza database specifica nel tuo cluster di database di Aurora.

> **Note**
>
> Per i cluster di database di Aurora Serverless, puoi connetterti all'endpoint del database anziché all'istanza database. Puoi trovare l'endpoint di database per un cluster di database Aurora Serverless nella scheda Connectivity & security (Connettività e sicurezza) della AWS Management Console. Per ulteriori informazioni, consulta Usare Amazon Aurora Serverless v1.

Indipendentemente dal motore di database Aurora e dagli strumenti specifici che utilizzi con il cluster o l'istanza database, l'endpoint deve essere accessibile. Un cluster Aurora DB può essere creato solo in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Ciò significa accedi all'endpoint dall'interno o dall'esterno del VPC utilizzando uno dei seguenti approcci.

- Accedi al cluster Aurora DB all'interno del VPC: abilita l'accesso al cluster Aurora DB tramite il VPC. A tale scopo, modifica le regole in entrata nel gruppo di sicurezza per il VPC in modo da consentire l'accesso al cluster di database Aurora specifico. Per ulteriori informazioni, incluso come configurare il VPC per diversi scenari di cluster Aurora DB, consulta Amazon Virtual Private Cloud e Amazon Aurora. VPCs
- Accedi al cluster Aurora DB all'esterno del VPC: per accedere a un cluster Aurora DB dall'esterno del VPC, utilizza l'indirizzo endpoint pubblico del cluster DB.

Per ulteriori informazioni, consulta Risoluzione dei problemi di connessione in Aurora.

Indice

- Connessione ai cluster Aurora DB con i driver AWS
- Connessione a un cluster di database Amazon Aurora MySQL
  - Utilità di connessione per Aurora MySQL
  - Connessione ad Aurora MySQL con l'utilità MySQL
  - Connessione ad Aurora MySQL con il driver JDBC Amazon Web Services ()AWS
  - Connessione ad Aurora MySQL con il driver Python di Amazon Web Services ()AWS
  - Connessione ad Aurora MySQL con il driver ODBC di Amazon Web Services () per MySQL AWS
  - Connessione ad Aurora MySQL con Amazon Web Services () Advanced NodeJS Wrapper AWS
  - Connessione ad Aurora MySQL tramite SSL
- Connessione a un cluster di database Amazon Aurora PostgreSQL
  - Utilità di connessione per Aurora PostgreSQL
  - Connessione ad Aurora PostgreSQL con il driver JDBC Amazon Web Services ()AWS
  - Connessione ad Aurora PostgreSQL con il driver Python di Amazon Web Services ()AWS
  - Connessione ad Aurora PostgreSQL con Amazon Web Services () Advanced NodeJS Wrapper AWS
- Risoluzione dei problemi di connessione in Aurora

Connessione ai cluster Aurora DB con i driver AWS

La AWS suite di driver è stata progettata per fornire supporto per tempi di switchover e failover più rapidi e l'autenticazione con AWS Secrets Manager, AWS Identity and Access Management (IAM) e Federated Identity. I AWS driver si basano sul monitoraggio dello stato del cluster DB e sulla conoscenza della topologia del cluster per determinare il nuovo writer. Questo approccio riduce i tempi di switchover e failover a secondi a una cifra, rispetto alle decine di secondi dei driver open source.

La tabella seguente elenca le funzionalità supportate per ciascuno dei driver. Con l'introduzione di nuove funzionalità di servizio, l'obiettivo della AWS suite di driver è disporre di un supporto integrato per tali funzionalità di servizio.

Funzionalità	AWS Driver JDBC	AWS Driver Python	AWS Driver ODBC per MySQL	AWS Wrapper NodeJS avanzato
Supporto per il failover	Sì	Sì	Sì	Sì
Monitoraggio avanzato del failover	Sì	Sì	Sì	Sì
Suddivisione in lettura/scrittura	Sì	Sì	No	Sì
Localizzatore di connessione Aurora	Sì	Sì	No	Sì
Connessione ai metadati del driver	Sì	N/D	N/D	N/D
Telemetria	Sì	Sì	No	Sì
Secrets Manager	Sì	Sì	Sì	Sì
Autenticazione IAM	Sì	Sì	Sì	Sì
Identità federata (ADFS)	Sì	Sì	No	Sì
Identità federata (Okta)	Sì	Sì	Sì	Sì
Database Aurora PostgreSQL Limitless	Sì (solo Aurora PostgreSQL)	No	No	Sì (solo Aurora PostgreSQL)

Per ulteriori informazioni sui AWS driver, consulta il driver di lingua corrispondente per il cluster Aurora MySQL o Aurora PostgreSQL DB.

Connessione a un cluster di database Amazon Aurora MySQL

Per autenticarsi sul cluster Aurora MySQL DB, è possibile utilizzare l'autenticazione con nome utente e password MySQL o l'autenticazione del database (IAM). AWS Identity and Access Management Per ulteriori informazioni sull'utilizzo dell'autenticazione tramite nome utente e password MySQL, consulta la pagina relativa a Access Control and Account Management nella documentazione di MySQL. Per ulteriori informazioni sull'utilizzo dell'autenticazione database IAM, consulta Autenticazione del database IAM .

Dopo la connessione a un cluster di database Amazon Aurora con compatibilità MySQL 8.0, puoi eseguire comandi SQL compatibili con MySQL versione 8.0. La versione minima compatibile è MySQL 8.0.23. Per ulteriori informazioni sulla sintassi SQL di MySQL 8.0, consulta il manuale di riferimento di MySQL 8.0. Per informazioni sui limiti relativi a Aurora MySQL 3, consulta Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition.

Dopo la connessione a un cluster di database Amazon Aurora con compatibilità MySQL 5.7, puoi eseguire comandi SQL compatibili con MySQL versione 5.7. Per ulteriori informazioni sulla sintassi SQL di MySQL 5.7, consulta il manuale di riferimento di MySQL 5.7. Per informazioni sui limiti relativi a Aurora MySQL 5.7, consulta Aurora My SQL versione 2 compatibile con My 5.7 SQL.

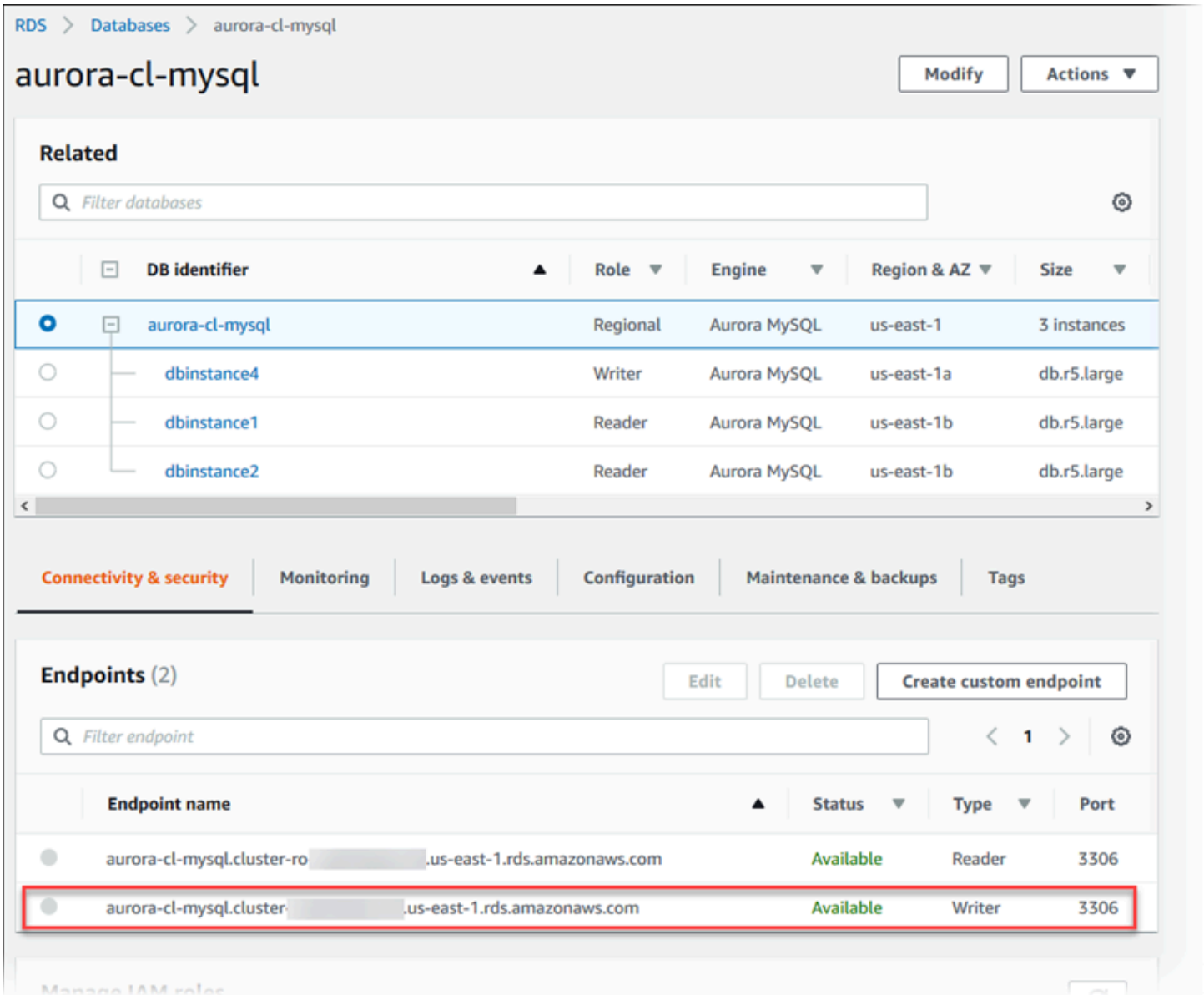
> **Note**
>
> Per una guida dettagliata e utile sulla connessione a un cluster di database Amazon Aurora MySQL, consulta il manuale relativo alla gestione delle connessioni di Aurora.

Nella vista dettagliata del cluster di database, puoi trovare l'endpoint del cluster, utilizzabile nella stringa di connessione MySQL. L'endpoint è costituito da una porta e da un nome di dominio per il cluster di database. Ad esempio, se il valore di un endpoint è `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, specifica i valori seguenti in una stringa di connessione MySQL:

- Per l'host o il nome host, specifica `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Per la porta, specifica `3306` o il valore di porta utilizzato alla creazione del cluster di database.

L'endpoint del cluster consente la connessione all'istanza principale del cluster di database. Puoi eseguire operazioni di lettura e scrittura utilizzando l'endpoint del cluster. Il cluster di database può anche avere fino a 15 repliche di Aurora che supportano l'accesso in sola lettura ai dati nel cluster di database. L'istanza principale e ogni replica Aurora hanno un endpoint univoco che è indipendente dall'endpoint del cluster e che ti consente di eseguire la connessione direttamente a una specifica istanza database nel cluster. L'endpoint del cluster punta sempre all'istanza principale. Se l'istanza principale non riesce e viene sostituita, l'endpoint del cluster punta alla nuova istanza principale.

Per visualizzare l'endpoint del cluster (endpoint di scrittura), scegli Databases (Database) nella console Amazon RDS, quindi scegli il nome del cluster di database per visualizzare i dettagli corrispondenti.



Argomenti

- Utilità di connessione per Aurora MySQL
- Connessione ad Aurora MySQL con l'utilità MySQL
- Connessione ad Aurora MySQL con il driver JDBC Amazon Web Services ()AWS
- Connessione ad Aurora MySQL con il driver Python di Amazon Web Services ()AWS
- Connessione ad Aurora MySQL con il driver ODBC di Amazon Web Services () per MySQL AWS
- Connessione ad Aurora MySQL con Amazon Web Services () Advanced NodeJS Wrapper AWS
- Connessione ad Aurora MySQL tramite SSL

Utilità di connessione per Aurora MySQL

Alcune delle utilità di connessione che puoi utilizzare sono elencate di seguito:

- Riga di comando – Puoi eseguire la connessione a un cluster di database Amazon Aurora utilizzando strumenti come l'utilità a riga di comando MySQL. Per ulteriori informazioni sull'utilizzo dell'utilità di MySQL, consulta mysql - Il client della linea di comando di MySQL nella documentazione di MySQL.
- GUI – Puoi utilizzare l'utilità MySQL Workbench per eseguire la connessione utilizzando un'interfaccia utente. Per ulteriori informazioni, consulta la pagina Download MySQL Workbench.
- AWS driver:
  - Connessione ad Aurora MySQL con il driver JDBC Amazon Web Services ()AWS
  - Connessione ad Aurora MySQL con il driver Python di Amazon Web Services ()AWS
  - Connessione ad Aurora MySQL con il driver ODBC di Amazon Web Services () per MySQL AWS
  - Connessione ad Aurora MySQL con Amazon Web Services () Advanced NodeJS Wrapper AWS

Connessione ad Aurora MySQL con l'utilità MySQL

Attenersi alla seguente procedura: Presuppone che il cluster database sia stato configurato in una sottorete privata nel VPC. Ti connetti utilizzando un' EC2 istanza Amazon configurata in base ai tutorial in. Tutorial: creazione di un server Web e un cluster database Amazon Aurora

> **Note**
>
> Questa procedura non richiede l'installazione del server web nel tutorial, ma richiede l'installazione di MariaDB 10.5.

Eseguire la connessione a un cluster database con l'utilità MySQL

1. Accedi all' EC2 istanza che stai utilizzando per connetterti al tuo cluster DB.

   Verrà visualizzato un output simile al seguente.

   ```
   Last login: Thu Jun 23 13:32:52 2022 from xxx.xxx.xxx.xxx

          __|  __|_  )
          _|  (     /   Amazon Linux 2 AMI
         ___|\___|___|

   https://aws.amazon.com/amazon-linux-2/
   [ec2-user@ip-10-0-xxx.xxx ~]$
   ```

2. Digita il comando seguente al prompt dei comandi per eseguire la connessione all'istanza database principale di un cluster database.

   Per il parametro `-h`, sostituisci il nome DNS dell'endpoint per l'istanza principale. Per il parametro `-u`, sostituisci l'ID utente di un account utente di database.

   ```
   mysql -h primary-instance-endpoint.AWS_account.AWS_Region.rds.amazonaws.com -P 3306
    -u database_user -p
   ```

   Ad esempio:

   ```
   mysql -h my-aurora-cluster-instance.c1xy5example.123456789012.eu-
   central-1.rds.amazonaws.com -P 3306 -u admin -p
   ```

3. Immetti la password per l'utente del database.

   Verrà visualizzato un output simile al seguente.

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MySQL connection id is 1770
   ```

```
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

4. Immetti i comandi SQL.

Connessione ad Aurora MySQL con il driver JDBC Amazon Web Services ()AWS

Il driver JDBC di Amazon Web Services (AWS) è progettato come wrapper JDBC avanzato. Questo wrapper è complementare ed estende la funzionalità di un driver JDBC esistente per aiutare le applicazioni a sfruttare le funzionalità dei database cluster come Aurora MySQL. Il driver è compatibile direttamente con il driver MySQL della community. Connector/J driver and the community MariaDB Connector/J

Per installare il driver AWS JDBC, aggiungi il file.jar del driver AWS JDBC (che si trova nell'applicazione`CLASSPATH`) e mantieni i riferimenti al rispettivo driver della community. Aggiorna il rispettivo prefisso dell'URL di connessione come segue:

- `jdbc:mysql://` Da a `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` Da a `jdbc:aws-wrapper:mariadb://`

Per ulteriori informazioni sul driver AWS JDBC e istruzioni complete per il suo utilizzo, consulta l'archivio dei driver JDBC di Amazon Web Services (AWS). GitHub

> **Note**
>
> La versione 3.0.3 dell'utilità MariaDB Connector/J interrompe il supporto per i cluster Aurora DB, quindi consigliamo vivamente di passare al driver JDBC. AWS

Connessione ad Aurora MySQL con il driver Python di Amazon Web Services ()AWS

Il driver Python di Amazon Web Services (AWS) è progettato come wrapper Python avanzato. Questo wrapper è complementare ed estende le funzionalità del driver open source Psycopg. Il AWS Python Driver supporta le versioni Python 3.8 e successive. È possibile installare il `aws-`

`advanced-python-wrapper` pacchetto utilizzando il `pip` comando, insieme ai pacchetti open source. `psycopg`

Per ulteriori informazioni sul driver AWS Python e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services ()AWS Python Driver. GitHub

Connessione ad Aurora MySQL con il driver ODBC di Amazon Web Services () per MySQL AWS

Il driver AWS ODBC per MySQL è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Il driver può esistere insieme al driver MySQL Connector/ODBC ed è compatibile con gli stessi flussi di lavoro.

Per ulteriori informazioni sul driver AWS ODBC per MySQL e istruzioni complete per installarlo e utilizzarlo, consulta l'archivio Amazon Web Services AWS() ODBC Driver for MySQL. GitHub

Connessione ad Aurora MySQL con Amazon Web Services () Advanced NodeJS Wrapper AWS

L' AWS Advanced NodeJS Wrapper è complementare ed estende la funzionalità di un driver NodeJS esistente. Aiuta le applicazioni a sfruttare le funzionalità dei database cluster come Aurora MySQL.

Per ulteriori informazioni su AWS Advanced NodeJS Wrapper e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services (AWS) Advanced NodeJS Wrapper. GitHub

Connessione ad Aurora MySQL tramite SSL

Puoi utilizzare la crittografia SSL sulle connessioni a un'istanza database Aurora MySQL. Per informazioni, consulta Connessioni TLS ai cluster Aurora MySQL DB.

Per eseguire una connessione mediante SSL, utilizza l'utilità MySQL come descritto nella procedura successiva. Se utilizzi l'autenticazione database IAM, devi utilizzare una connessione SSL. Per informazioni, consulta Autenticazione del database IAM .

> **Note**
>
> Per eseguire la connessione all'endpoint del cluster mediante SSL, l'utilità di connessione client deve supportare Subject Alternative Names (SAN). Se l'utilità di connessione client non supporta SAN, puoi eseguire la connessione direttamente alle istanze nel cluster di database

Aurora. Per ulteriori informazioni sugli endpoint Aurora, consulta Connessioni endpoint Amazon Aurora.

Per eseguire la connessione a un cluster di database con SSL utilizzando l'utilità MySQL

1. Scaricare la chiave pubblica per il certificato di firma Amazon RDS.

   Per ulteriori informazioni sul download dei certificati, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.
2. Digitare il comando seguente a un prompt dei comandi per eseguire la connessione all'istanza principale di un cluster di database con SSL mediante l'utilità MySQL. Per il parametro `-h`, sostituisci il nome DNS dell'endpoint per l'istanza principale. Per il parametro `-u`, sostituisci l'ID utente di un account utente di database. Per il parametro `--ssl-ca`, sostituire il nome file del certificato SSL come appropriato. Digitare la password dell'utente master, quando richiesto.

   ```
   mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com -u admin_user -p --ssl-ca=[full path]global-bundle.pem --ssl-verify-server-cert
   ```

Verrà visualizzato un output simile al seguente.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 8.0.26-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Per istruzioni generali sulla costruzione delle stringhe di connessione di RDS for MySQL e sulla ricerca della chiave pubblica per le connessioni SSL, consulta Connessione a un'istanza database che esegue il motore di database di MySQL.

Connessione a un cluster di database Amazon Aurora PostgreSQL

Puoi eseguire la connessione a un'istanza database nel cluster di database Amazon Aurora PostgreSQL utilizzando gli stessi strumenti che utilizzi per connetterti a un database PostgreSQL. A questo proposito, utilizzi la stessa chiave pubblica per le connessioni Secure Sockets Layer (SSL).

Puoi utilizzare le informazioni relative a endpoint e porta dell'istanza principale o le repliche Aurora del cluster di database PostgreSQL di Aurora nella stringa di connessione di qualsiasi script, utilità o applicazione che si connette a un'istanza database PostgreSQL. Nella stringa di connessione, specifica l'indirizzo DNS dell'endpoint dell'istanza principale o della replica Aurora come parametro host. Specifica il numero di porta dell'endpoint come parametro della porta.

Dopo la connessione a un'istanza database nel cluster di database Amazon Aurora PostgreSQL, puoi eseguire qualsiasi comando SQL compatibile con PostgreSQL.

Nella visualizzazione dettagliata del cluster di database Aurora PostgreSQL, puoi trovare il nome lo status, il tipo e il numero di porta dell'endpoint del cluster. Puoi utilizzare il numero di endpoint e di porta nella stringa di connessione PostgreSQL. Ad esempio, se il valore di un endpoint è `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`, specifica i valori seguenti in una stringa di connessione PostgreSQL:

- Per l'host o il nome host, specifica `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Per la porta, specifica `5432` o il valore di porta utilizzato alla creazione del cluster di database.

L'endpoint del cluster consente la connessione all'istanza principale del cluster di database. Puoi eseguire operazioni di lettura e scrittura utilizzando l'endpoint del cluster. Il cluster di database può anche avere fino a 15 repliche di Aurora che supportano l'accesso in sola lettura ai dati nel cluster di database. Ogni istanza database nel cluster Aurora, ovvero l'istanza principale e ogni replica Aurora, ha un endpoint univoco che è indipendente dall'endpoint del cluster. e che ti consente di eseguire la connessione direttamente a una specifica istanza database nel cluster. L'endpoint del cluster punta sempre all'istanza principale. Se l'istanza principale non riesce e viene sostituita, l'endpoint del cluster punta alla nuova istanza principale.

Per visualizzare l'endpoint del cluster (endpoint di scrittura), scegli Databases (Database) nella console Amazon RDS, quindi scegli il nome del cluster di database per visualizzare i dettagli corrispondenti.

Utilità di connessione per Aurora PostgreSQL

Alcune delle utilità di connessione che puoi utilizzare sono elencate di seguito:

- Riga di comando: puoi eseguire la connessione a cluster database Aurora PostgreSQL utilizzando strumenti come psql, il terminale interattivo PostgreSQL. Per ulteriori informazioni sull'utilizzo del terminale interattivo PostgreSQL, consulta psql nella documentazione di PostgreSQL.

- GUI: puoi utilizzare l'utilità pgAdmin per eseguire la connessione a cluster database Aurora PostgreSQL utilizzando un'interfaccia utente. Per ulteriori informazioni, consulta la pagina dei download sul sito Web di pgAdmin.
- AWS driver:
  - Connessione ad Aurora PostgreSQL con il driver JDBC Amazon Web Services ()AWS
  - Connessione ad Aurora PostgreSQL con il driver Python di Amazon Web Services ()AWS
  - Connessione ad Aurora PostgreSQL con Amazon Web Services () Advanced NodeJS Wrapper AWS

Connessione ad Aurora PostgreSQL con il driver JDBC Amazon Web Services ()AWS

Il driver JDBC di Amazon Web Services (AWS) è progettato come wrapper JDBC avanzato. Questo wrapper è complementare ed estende la funzionalità di un driver JDBC esistente per aiutare le applicazioni a sfruttare le funzionalità dei database cluster come Aurora PostgreSQL. Il driver è compatibile direttamente con il driver PgJDBC della community.

Per installare il driver AWS JDBC, aggiungi il file.jar del driver AWS JDBC (che si trova nell'applicazione) e mantieni i riferimenti al driver della community pgJDBC. `CLASSPATH` Aggiorna `jdbc:postgresql://` `jdbc:aws-wrapper:postgresql://` il prefisso dell'URL di connessione da a.

Per ulteriori informazioni sul driver AWS JDBC e istruzioni complete per il suo utilizzo, consulta l'archivio dei driver JDBC di Amazon Web Services (AWS). GitHub

Connessione ad Aurora PostgreSQL con il driver Python di Amazon Web Services ()AWS

Il driver Python di Amazon Web Services (AWS) è progettato come wrapper Python avanzato. Questo wrapper è complementare ed estende le funzionalità del driver open source Psycopg. Il AWS Python Driver supporta le versioni Python 3.8 e successive. È possibile installare il `aws-advanced-python-wrapper` pacchetto utilizzando il `pip` comando, insieme ai pacchetti open source. `psycopg`

Per ulteriori informazioni sul driver AWS Python e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services ()AWS Python Driver. GitHub

Connessione ad Aurora PostgreSQL con Amazon Web Services () Advanced NodeJS Wrapper AWS

L' AWS Advanced NodeJS Wrapper è complementare ed estende la funzionalità di un driver NodeJS esistente. Aiuta le applicazioni a sfruttare le funzionalità dei database cluster come Aurora PostgreSQL.

Per ulteriori informazioni su AWS Advanced NodeJS Wrapper e istruzioni complete per il suo utilizzo, consulta il repository Amazon Web Services (AWS) Advanced NodeJS Wrapper. GitHub

Risoluzione dei problemi di connessione in Aurora

Di seguito, le cause più comuni dei problemi di connessione a un nuovo cluster di database Aurora:

- Il gruppo di sicurezza nel VPC non consente l'accesso: il tuo VPC deve consentire le connessioni dal tuo dispositivo o da un' EC2 istanza Amazon mediante una corretta configurazione del gruppo di sicurezza nel VPC. Per risolvere il problema, modifica le regole del gruppo di sicurezza del VPC in ingresso in modo da consentire le connessioni. Per un esempio, consulta Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 .
- La porta è bloccata per via delle regole del firewall – Controlla il valore della porta configurato per il cluster di database Aurora. Se una regola del firewall blocca tale porta, puoi ricreare l'istanza utilizzando una porta diversa.
- La configurazione di IAM è incompleta o non corretta – Se l'istanza database Aurora che hai creato richiede l'utilizzo dell'autenticazione basata su IAM, assicurarti che la configurazione sia corretta. Per ulteriori informazioni, consulta Autenticazione del database IAM .

Per ulteriori informazioni sulla risoluzione dei problemi di connessione al database Aurora, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Parametri database specificano la modalità di configurazione del database. Ad esempio, i parametri del database possono specificare la quantità di risorse, come la memoria, da allocare a un database.

È possibile gestire la configurazione del database associando le istanze database e i cluster di database Aurora con i gruppi di parametri. Aurora definisce i gruppi di parametri con le impostazioni di default. Puoi definire i gruppi di parametri anche con le impostazioni personalizzate.

Argomenti

- Panoramica dei gruppi di parametri
- Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora
- Gruppi di parametri DB per DB Amazon Aurora
- Confronto di gruppi di parametri database
- Specifica dei parametri del database

Panoramica dei gruppi di parametri

Un gruppo di parametri cluster database agisce da container per i valori di configurazione del motore che si applicano a ogni istanza database in un cluster database Aurora. Ad esempio, il modello di storage condiviso di Aurora richiede che ogni istanza database in un cluster Aurora utilizzi la stessa impostazione per i parametri come ad esempio `innodb_file_per_table`. Pertanto, i parametri che influenzano il layout di storage fisico sono parte del gruppo di parametri del cluster. Il gruppo di parametri cluster database include anche i valori predefiniti per tutti i parametri a livello di istanza.

Un gruppo di parametri database agisce da container per i valori di configurazione del motore che si applicano a una o più istanze database. I gruppi di parametri di database si applicano alle istanze database in Amazon RDS e Aurora. Queste impostazioni di configurazione si applicano alle proprietà che possono variare tra le istanze database all'interno di un cluster Aurora, ad esempio le dimensioni dei buffer di memoria.

Argomenti

- Gruppi di parametri predefiniti e personalizzati
- Parametri statici e dinamici del cluster database
- Parametri statici e dinamici dell'istanza database
- Parametri del set di caratteri
- Parametri e valori dei parametri supportati

Gruppi di parametri predefiniti e personalizzati

Se decidi di creare un'istanza database senza specificare un gruppo di parametri di database, l'istanza database utilizza un gruppo di parametri predefinito. Allo stesso modo, se crei un cluster di database Aurora senza specificare un gruppo di parametri cluster di database, il cluster di database utilizza un gruppo di parametri cluster di databasedi default. Ogni gruppo di parametri di default contiene le impostazioni predefinite del motore del database e le impostazioni predefinite di sistema di Amazon RDS in base a motore, classe di elaborazione e storage allocato dell'istanza.

Non puoi modificare le impostazioni dei parametri di un gruppo di parametri predefinito. Puoi invece procedere come descritto di seguito:

1. Crea un nuovo set di parametri.
2. Modifica le impostazioni dei parametri desiderati. Non tutti i parametri del motore di database presenti nel gruppo di parametri possono essere modificati.
3. Modifica l'istanza DB o il cluster DB per associare il nuovo gruppo di parametri.

   Per informazioni sulla modifica di un cluster database o un'istanza database, consulta Modifica di un cluster database Amazon Aurora.

   > **Note**
   >
   > Se hai modificato l'istanza database per utilizzare un gruppo di parametri personalizzati e avvii l'istanza database, RDS riavvia automaticamente l'istanza database come parte del processo di avvio. Per le istanze RDS for SQL Server Multi-AZ con AlwaysOn l'opzione Mirroring abilitata, è previsto un failover al riavvio dell'istanza dopo il processo di avvio.

RDS applica i parametri statici e dinamici modificati in un nuovo gruppo di parametri associato solo dopo il riavvio dell'istanza DB. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio. Per informazioni sulla modifica del gruppo di parametri database, consulta Modifica di un cluster database Amazon Aurora.

Se aggiorni i parametri all'interno di un gruppo di parametri database, le modifiche si applicano a tutte le istanze database associate al gruppo di parametri. Allo stesso modo, se aggiorni i parametri in un gruppo di parametri cluster database Aurora, le modifiche si applicano a tutti i cluster database Aurora associati al gruppo di parametri cluster database.

Se non si desidera creare un gruppo di parametri da zero, è possibile copiare un gruppo di parametri esistente con il AWS CLIcopy-db-parameter-groupcomando o copy-db-cluster-parameter il comando -group. In alcuni casi la copia di un gruppo di parametri è utile. Ad esempio quando devi includere la maggior parte dei valori e dei parametri personalizzati del gruppo di parametri esistente in un nuovo gruppo di parametri .

Parametri statici e dinamici del cluster database

I parametri di cluster di database sono statici o dinamici. Differiscono nei seguenti modi:

- Quando modifichi un parametro statico e salvi il gruppo di parametri del cluster di database, la modifica del parametro diventa effettiva al riavvio manuale di ogni istanza database sul cluster di database associato. Quando si utilizza il AWS Management Console per modificare i valori dei parametri statici del cluster DB, viene sempre utilizzato `pending-reboot` for. `ApplyMethod`
- Quando si modifica un parametro dinamico, per impostazione predefinita la modifica del parametro diventa immediatamente effettiva, senza richiedere il riavvio. Quando usi la console, utilizza sempre `immediate` per `ApplyMethod`. Per rimandare la modifica dei parametri a dopo il riavvio delle istanze DB in un cluster DB associato, utilizza l'API AWS CLI o RDS. Quindi, imposta il valore `ApplyMethod` su `pending-reboot` per la modifica del parametro.

Per ulteriori informazioni sull'utilizzo di per modificare il valore AWS CLI di un parametro, consulta -group. modify-db-cluster-parameter Per ulteriori informazioni sull'utilizzo dell'API RDS per modificare il valore di un parametro, consulta Modify. DBCluster ParameterGroup

Se si modifica il gruppo di parametri cluster database associato a un cluster database, occorre riavviare le istanze database nel cluster database per applicare le modifiche a tutte le istanze database nel cluster database. Per determinare se le istanze database di un cluster di database devono essere riavviate per applicare le modifiche, esegui il seguente comando AWS CLI .

```
aws rds describe-db-clusters --db-cluster-identifier db_cluster_identifier
```

Controlla il valore `DBClusterParameterGroupStatus` per l'istanza database primaria nell'output. Se il valore è `pending-reboot`, riavvia l'istanza database del cluster di database.

Parametri statici e dinamici dell'istanza database

I parametri di istanza database sono statici o dinamici. Di seguito sono riportate le differenze:

- Quando modifichi un parametro statico e salvi il gruppo parametri del database, la modifica del parametro diventa effettiva al riavvio manuale delle istanze database associate. Per i parametri statici, la console utilizza sempre `pending-reboot` per `ApplyMethod`.
- Quando si modifica un parametro dinamico, per impostazione predefinita la modifica del parametro diventa immediatamente effettiva, senza richiedere il riavvio. Quando si utilizza il AWS Management Console per modificare i valori dei parametri dell'istanza DB, viene sempre utilizzato `immediate` `ApplyMethod` per i parametri dinamici. Per posticipare la modifica dei parametri fino al riavvio di un'istanza DB associata, utilizza l'API AWS CLI o RDS. Quindi, imposta il valore `ApplyMethod` su `pending-reboot` per la modifica del parametro.

Per ulteriori informazioni sull'utilizzo di per modificare il valore AWS CLI di un parametro, vedere. modify-db-parameter-group Per ulteriori informazioni sull'utilizzo dell'API RDS per modificare il valore di un parametro, consulta Modify DBParameter Group.

Se l'istanza database non usa le modifiche più recenti apportate al gruppo di parametri database associato, la console mostra il gruppo di parametri database con lo stato pending-reboot. Questo stato non comporta il riavvio automatico durante la successiva finestra di manutenzione. Per applicare le ultime modifiche del parametro su quella istanza database, riavvia manualmente l'istanza database.

Parametri del set di caratteri

Prima di creare il cluster database imposta tutti i parametri correlati al set di caratteri o alla regola di confronto del database nel gruppo di parametri. prima di creare un database. In questo modo, il database predefinito e i nuovi database usano i valori della regola di confronto e del set di caratteri specificati. Se modifichi parametri di confronto o del set di caratteri, le modifiche dei parametri non vengono applicate a database esistenti.

Per alcuni motori database puoi modificare valori di confronto o del set di caratteri per un database esistente usando il comando `ALTER DATABASE`, ad esempio:

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

Per ulteriori informazioni su come modificare il set di caratteri o i valori di confronto relativi a un database, consulta la documentazione relativa al motore database.

Parametri e valori dei parametri supportati

Per determinare i parametri supportati per il motore del database, vedi i parametri nel gruppo di parametri database e il gruppo di parametri cluster database usato dall'istanza database o dal cluster database. Per ulteriori informazioni, consultare Visualizzazione dei valori dei parametri per un gruppo di parametri DB in Aurora e Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster DB in Amazon Aurora.

In molti casi è possibile specificare valori interi e booleani per i parametri di database utilizzando espressioni, formule e funzioni. Le funzioni possono includere un'espressione logaritmica matematica. Tuttavia, non tutti i parametri supportano espressioni, formule e funzioni per i valori dei parametri. Per ulteriori informazioni, consulta Specifica dei parametri del database.

Per un Aurora Global Database, puoi specificare impostazioni di configurazione diverse per i singoli cluster Aurora. Assicurati che le impostazioni siano abbastanza simili da produrre un comportamento coerente se promuovi un cluster secondario a cluster primario. Ad esempio, utilizza le stesse impostazioni per fusi orari e set di caratteri su tutti i cluster di un Aurora Global Database.

Un'impostazione errata dei parametri in un gruppo di parametri può avere conseguenze negative impreviste, tra cui il peggioramento delle prestazioni e l'instabilità del sistema. Fai sempre attenzione quando modifichi i parametri database ed esegui il backup dei dati prima di modificare un gruppo di parametri. Prova le modifiche delle impostazioni del gruppo di parametri in un'istanza database o un cluster database di test prima di applicare le modifiche a un'istanza database o un cluster database di produzione.

Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora

I cluster database Amazon Aurora utilizzano gruppi di parametri di cluster di database. Le sezioni seguenti descrivono la configurazione e la gestione dei gruppi di parametri del cluster di database.

Argomenti

- Parametri dell'istanza database e del cluster database di Amazon Aurora
- Creazione di un gruppo di parametri del cluster DB in Amazon Aurora
- Associazione di un gruppo di parametri del cluster DB a un cluster DB in Amazon Aurora
- Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora
- Reimpostazione dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora
- Copia di un gruppo di parametri del cluster DB in Amazon Aurora
- Elenco dei gruppi di parametri del cluster DB in Amazon Aurora

- Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster DB in Amazon Aurora
- Eliminazione di un gruppo di parametri del cluster DB in Amazon Aurora

Parametri dell'istanza database e del cluster database di Amazon Aurora

Aurora utilizza un sistema a due livelli di impostazioni di configurazione:

- I parametri in un gruppo di parametri cluster database si applicano a ogni istanza database in un cluster database. I dati vengono memorizzati nel sistema secondario di storage condiviso Aurora. Per questo motivo, tutti i parametri correlati al layout fisico dei dati della tabella devono essere gli stessi per tutte le istanze database in un cluster Aurora. Allo stesso modo, poiché le istanze database Aurora sono connesse dalla replica, tutti i parametri per le impostazioni di replica devono essere identici in un cluster Aurora.
- I parametri in un gruppo di parametri database si applicano a una sola istanza database in un cluster database Aurora. Questi parametri sono correlati ad aspetti quali l'utilizzo di memoria che puoi modificare tra le istanze database nello stesso cluster Aurora. Ad esempio, un cluster spesso contiene istanze database con classi di istanza AWS differenti.

Ogni cluster Aurora è associato a un gruppo di parametri del cluster database. Questo gruppo di parametri assegna i valori di default per ogni valore di configurazione per il motore del database corrispondente. Il gruppo di parametri del cluster include i valori di default per i parametri a livello di cluster e di istanza. Ogni istanza database all'interno di un cluster Aurora Serverless v2 o con provisioning eredita le impostazioni dal gruppo di parametri del cluster di database.

Ogni istanza database è associata anche a un gruppo parametri del database. I valori nel gruppo parametri del database possono sostituire i valori di default del gruppo di parametri del cluster. Ad esempio, se un'istanza in un cluster ha riscontrato problemi, è possibile assegnare a tale istanza un gruppo parametri del database personalizzato. Il gruppo di parametri personalizzati può avere impostazioni specifiche per i parametri relativi al debug o all'ottimizzazione delle prestazioni.

Quando crei un cluster o una nuova istanza database, Aurora assegna gruppi di parametri predefiniti in base al motore database e alla versione specificati. È invece possibile specificare gruppi di parametri personalizzati. Puoi creare autonomamente tali gruppi di parametri e puoi modificare i valori dei parametri. È possibile specificare questi gruppi di parametri personalizzati al momento della creazione. È inoltre possibile modificare un cluster o un'istanza database in un secondo momento per utilizzare un gruppo di parametri personalizzati.

Per le istanze Aurora Serverless v2 e con provisioning, eventuali valori di configurazione che si modificano nel gruppo di parametri del cluster di database sostituiscono i valori di default nel gruppo parametri del database. Se modifichi i valori corrispondenti nel gruppo di parametri di database, tali valori sostituiranno le impostazioni del gruppo di parametri del cluster database.

Ogni impostazione dei parametri di database che modifichi acquisisce la precedenza sui valori del gruppo di parametri del cluster database, anche se modifichi i parametri di configurazione riportandoli ai valori di default. Puoi vedere quali parametri vengono sovrascritti utilizzando il describe-db-parameters AWS CLI comando o l'operazione Descrivi DBParameters RDS API. Il campo `Source` contiene il valore `user` se hai modificato quel parametro. Per reimpostare uno o più parametri in modo che il valore del gruppo di parametri del cluster DB abbia la precedenza, utilizza il reset-db-parameter-group AWS CLI comando o l'operazione API Reset DBParameter Group RDS.

I parametri dell'istanza database e del cluster database disponibili in Aurora variano in base alla compatibilità del motore del database.

Motore del database	Parametri
Aurora MySQL	Per informazioni, consulta Parametri di configurazione Aurora MySQL. Per i cluster Aurora Serverless, vedi i dettagli aggiuntivi in Uso di gruppi di parametri per Aurora Serverless v2 e Gruppi di parametri per Aurora Serverless v1.
Aurora PostgreSQL	Per informazioni, consulta Amazon Aurora PostgreSQL parametri. Per i cluster Aurora Serverless, vedi i dettagli aggiuntivi in Uso di gruppi di parametri per Aurora Serverless v2 e Gruppi di parametri per Aurora Serverless v1.

> **Note**
>
> I cluster Aurora Serverless v1 hanno solo gruppi di parametri del cluster di database, non gruppi parametri del database. Per i cluster Aurora Serverless v2, apporti tutte le modifiche ai parametri personalizzati nel gruppo di parametri del cluster di database.

Aurora Serverless v2 utilizza sia gruppi di parametri del cluster di database che gruppi parametri del database. Con Aurora Serverless v2, è possibile modificare quasi tutti i parametri di configurazione. Aurora Serverless v2 sostituisce le impostazioni di alcuni parametri di configurazione relativi alla capacità in modo che il carico di lavoro non venga interrotto quando le istanze Aurora Serverless v2 si riducono.
Per ulteriori informazioni sulle impostazioni di configurazione di Aurora Serverless e sulle impostazioni che puoi modificare, consulta Uso di gruppi di parametri per Aurora Serverless v2 e Gruppi di parametri per Aurora Serverless v1.

Creazione di un gruppo di parametri del cluster DB in Amazon Aurora

Puoi creare un nuovo gruppo di parametri del cluster DB utilizzando l' AWS Management Console AWS CLI, la o l'API RDS.

Dopo aver creato un gruppo di parametri del cluster database, devi attendere almeno 5 minuti prima di creare un cluster database che utilizza tale gruppo. In questo modo, Amazon RDS può completare l'operazione di creazione del gruppo di parametri prima che tale gruppo venga usato dal nuovo cluster database. Puoi utilizzare la pagina Gruppi di parametri nella console Amazon RDS o il describe-db-cluster-parameterscomando per verificare che il gruppo di parametri del cluster DB sia stato creato.

Le seguenti limitazioni si applicano al nome del gruppo di parametri del·cluster database:

- Il nome deve contenere da 1 a 255 lettere, numeri o trattini.

  I nomi del gruppo di parametri predefiniti possono includere un punto, ad esempio `default.aurora-mysql5.7`. Tuttavia, i nomi del gruppo di parametri personalizzati non possono includere un punto.
- Il primo carattere deve essere una lettera.
- Il nome non può terminare con un trattino o contenere due trattini consecutivi.

Console

Per creare un gruppo di parametri del cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione, scegli Parameter groups (Gruppi di parametri).
3. Scegli Create parameter group (Crea gruppo di parametri).
4. Per il nome del gruppo di parametri, inserisci il nome del nuovo gruppo di parametri del cluster DB.
5. In Descrizione, immettete una descrizione per il nuovo gruppo di parametri del cluster DB.
6. Per Tipo di motore, scegli il tuo motore di database.
7. Per Famiglia di gruppi di parametri, scegli una famiglia di gruppi di parametri DB.
8. Scegli Create (Crea).

AWS CLI

Per creare un gruppo di parametri del cluster DB, utilizzare il AWS CLI `create-db-cluster-parameter-group`comando.

L'esempio seguente crea un gruppo di parametri cluster di database denominato mydbclusterparametergroup per Aurora MySQL versione 5.7 con la descrizione "My new cluster parameter group (Il mio nuovo gruppo di parametri cluster)".

Includi i parametri obbligatori seguenti:

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Per elencare tutte le famiglie del gruppo di parametri disponibili, usa il comando seguente:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> L'output contiene duplicati.

Example

Per LinuxmacOS, oUnix:

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new cluster parameter group"
```

Per Windows:

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new cluster parameter group"
```

Questo comando genera un output simile al seguente:

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
    }
}
```

API RDS

Per creare un gruppo di parametri del cluster di database, utilizza l'operazione `CreateDBClusterParameterGroup` dell'API RDS.

Includi i parametri obbligatori seguenti:

- `DBClusterParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Associazione di un gruppo di parametri del cluster DB a un cluster DB in Amazon Aurora

Puoi creare gruppi di parametri del cluster database personalizzati con impostazioni personalizzate. È possibile associare un gruppo di parametri del cluster DB a un cluster DB utilizzando il AWS

Management Console AWS CLI, il o il RDSAPI. Ciò è possibile quando si crea o si modifica un cluster database.

Per ulteriori informazioni sulla creazione di un gruppo di parametri del cluster database, consulta Creazione di un gruppo di parametri del cluster DB in Amazon Aurora. Per ulteriori informazioni sulla creazione di un cluster database, consulta Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster database, consulta Modifica di un cluster database Amazon Aurora.

> **Note**
>
> Per Aurora Postgre SQL 15.2, 14.7, 13.10, 12.14 e tutte le 11 versioni, quando modifichi il gruppo di parametri del cluster DB associato a un cluster DB, riavvia ogni istanza di replica per applicare le modifiche.
> Per determinare se l'istanza DB primaria di un cluster DB deve essere riavviata per applicare le modifiche, esegui il comando seguente: AWS CLI
> `aws rds describe-db-clusters --db-cluster-identifier` *`db_cluster_identifier`*
> Controlla il valore `DBClusterParameterGroupStatus` per l'istanza database primaria nell'output. Se il valore è `pending-reboot`, riavvia l'istanza database primaria del cluster di database.

Console

Per associare un gruppo di parametri del cluster database a un cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare il cluster di database che si desidera modificare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. Modifica l'impostazione del gruppo di parametri del cluster database .
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.

   La modifica viene applicata immediatamente indipendentemente dall'impostazione Programmazione delle modifiche .

6. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify cluster (Modifica cluster) per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per associare un gruppo di parametri del cluster DB a un cluster DB, utilizza il AWS CLI `modify-db-cluster`comando con le seguenti opzioni:

- `--db-cluster-name`
- `--db-cluster-parameter-group-name`

Nell'esempio seguente il gruppo di parametri database `mydbclpg` viene associato al cluster database `mydbcluster`.

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --db-cluster-parameter-group-name mydbclpg
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --db-cluster-parameter-group-name mydbclpg
```

RDS API

Per associare un gruppo di parametri del cluster DB a un cluster DB, utilizzare l'RDSAPI`ModifyDBCluster`operazione con i seguenti parametri:

- `DBClusterIdentifier`
- `DBClusterParameterGroupName`

Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora

Non puoi modificare i valori di parametri in un gruppo di parametri cluster database creato dal cliente. Non puoi modificare i valori dei parametri in un gruppo di parametri cluster database predefinito. Le modifiche ai parametri in un gruppo di parametri cluster database creato dal cliente vengono applicate a tutti i cluster database associati al gruppo di parametri del cluster database.

Console

Per modificare un gruppo di parametri del cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, selezionare il gruppo di parametri da modificare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Modificare i valori dei parametri desiderati. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile modificare valori nel gruppo di parametri predefinito.
6. Scegli Save changes (Salva modifiche).
7. Riavvia l'istanza DB principale (writer) nel cluster per applicarvi le modifiche.
8. Quindi riavviate le istanze DB del lettore per applicare le modifiche ad esse.

   Se non riavvii le istanze DB, un'operazione di failover potrebbe richiedere più tempo del normale.

AWS CLI

Per modificare un gruppo di parametri del cluster DB, utilizzate il AWS CLI `modify-db-cluster-parameter-group`comando con i seguenti parametri obbligatori:

- `--db-cluster-parameter-group-name`
- `--parameters`

L'esempio seguente modifica i valori `server_audit_logging` e `server_audit_logs_upload` nel gruppo di parametri cluster database denominato mydbclusterparametergroup.

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

API RDS

Per modificare un gruppo di parametri cluster database, utilizza l'operazione `ModifyDBClusterParameterGroup` dell'API RDS con i parametri obbligatori seguenti:

- `DBClusterParameterGroupName`
- `Parameters`

Reimpostazione dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora

È possibile reimpostare i parametri ai valori predefiniti in un gruppo di parametri del cluster di database creato dal cliente. Le modifiche ai parametri in un gruppo di parametri cluster database creato dal cliente vengono applicate a tutti i cluster database associati al gruppo di parametri cluster database.

> **Note**
>
> In un gruppo di parametri cluster di database predefinito, i parametri vengono sempre impostati sui valori di default.

Console

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Scegliere i parametri per i quali si desidera ripristinare i valori predefiniti. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile reimpostare valori in un gruppo di parametri predefinito.
6. Scegli Reimposta , quindi conferma selezionando Ripristina parametri.
7. Riavvia l'istanza DB principale nel cluster DB per applicare le modifiche a tutte le istanze DB del cluster DB.

AWS CLI

Per ripristinare i parametri in un gruppo di parametri del cluster DB ai valori predefiniti, utilizzare il AWS CLI `reset-db-cluster-parameter-group`comando con la seguente opzione obbligatoria:. `--db-cluster-parameter-group-name`

Per reimpostare tutti i parametri nel gruppo di parametri del cluster di database, specificare l'opzione `--reset-all-parameters`. Per reimpostare parametri specifici, specifica l'opzione `--parameters`.

Nell'esempio seguente tutti i parametri del gruppo di parametri del database denominato mydbparametergroup vengono reimpostati sui valori predefiniti.

Example

Per LinuxmacOS, oUnix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Per Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

L'esempio seguente reimposta i valori `server_audit_logging` e `server_audit_logs_upload` nel gruppo di parametri cluster database denominato mydbclusterparametergroup.

Example

Per LinuxmacOS, oUnix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Per Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

API RDS

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri del cluster di database, utilizzare il comando RDS API `ResetDBClusterParameterGroup` con il seguente parametro obbligatorio: `DBClusterParameterGroupName`.

Per reimpostare tutti i parametri nel gruppo di parametri del cluster di database, impostare il parametro `ResetAllParameters` su `true`. Per reimpostare parametri specifici, specifica il parametro `Parameters`.

Copia di un gruppo di parametri del cluster DB in Amazon Aurora

Puoi copiare i gruppi di parametri cluster database personalizzati che hai creato. La copia di un gruppo di parametri è una soluzione pratica quando hai già creato un gruppo di parametri cluster database e vuoi includere la maggior parte dei parametri e valori personalizzati dal gruppo in un nuovo gruppo di parametri cluster database. Puoi copiare un gruppo di parametri del cluster DB utilizzando il comando AWS CLIcopy-db-cluster-parameter-group o l'operazione RDS API Copy. DBCluster ParameterGroup

Dopo aver copiato un gruppo di parametri del cluster database, devi attendere almeno 5 minuti prima di creare un cluster database che utilizza tale gruppo. In questo modo, Amazon RDS può completare l'operazione di copia del gruppo di parametri prima che tale gruppo venga usato dal nuovo cluster database. Puoi utilizzare la pagina Gruppi di parametri nella console Amazon RDS o il describe-db-cluster-parameterscomando per verificare che il gruppo di parametri del cluster DB sia stato creato.

> **Note**
>
> Non puoi copiare un gruppo di parametri predefinito. Tuttavia, puoi creare un nuovo gruppo di parametri basato su un gruppo di parametri predefinito.
> Non puoi copiare un gruppo di parametri del cluster DB in un altro Account AWS o Regione AWS.

Console

Per copiare un gruppo di parametri cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri personalizzato da copiare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Copy (Copia).
5. In New DB parameter group identifier (Identificatore nuovo gruppo di parametri database) immettere un nome per il nuovo gruppo di parametri.
6. Nella casella Description (Descrizione), immettere una descrizione per il nuovo gruppo di parametri.
7. Scegliere Copy (Copia).

AWS CLI

Per copiare un gruppo di parametri del cluster DB, usa il AWS CLI `copy-db-cluster-parameter-group`comando con i seguenti parametri obbligatori:

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

L'esempio seguente crea un nuovo gruppo di parametri cluster database denominato `mygroup2`, che è una copia del gruppo di parametri cluster database `mygroup1`.

Example

Per LinuxmacOS, oUnix:

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Per Windows:

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

API RDS

Per copiare un gruppo di parametri cluster database, utilizza l'operazione API RDS `CopyDBClusterParameterGroup` con i parametri obbligatori seguenti:

- `SourceDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupDescription`

Elenco dei gruppi di parametri del cluster DB in Amazon Aurora

Puoi elencare i gruppi di parametri del cluster DB che hai creato per il tuo AWS account.

> **Note**
>
> I gruppi di parametri predefiniti vengono creati automaticamente da un modello di parametro predefinito quando crei un cluster database per un motore e una versione di database specifici. Questi gruppi di parametri predefiniti contengono le impostazioni dei parametri preferite e non possono essere modificati. Quando crei un gruppo di parametri personalizzato, puoi modificare le impostazioni dei parametri.

Console

Per elencare tutti i gruppi di parametri del cluster DB per un AWS account

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   Il gruppo di parametri cluster database viene visualizzato nell'elenco con Type (Tipo) impostato su DB cluster parameter group (Gruppo di parametri cluster database).

AWS CLI

Per elencare tutti i gruppi di parametri del cluster DB per un AWS account, usa il AWS CLI `describe-db-cluster-parameter-groups`comando.

Example

Nell'esempio seguente sono elencati tutti i gruppi di parametri del cluster database disponibili per un account AWS .

```
aws rds describe-db-cluster-parameter-groups
```

L'esempio seguente descrive il gruppo di parametri mydbclusterparametergroup.

Per LinuxmacOS, oUnix:

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Per Windows:

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Questo comando restituisce una risposta simile alla seguente:

```
{
    "DBClusterParameterGroups": [
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
    ]
}
```

API RDS

Per elencare tutti i gruppi di parametri del cluster DB per un AWS account, utilizza l'DescribeDBClusterParameterGroupsazione API RDS.

Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster DB in Amazon Aurora

Puoi ottenere un elenco di tutti i parametri in un gruppo di parametri del cluster database e dei rispettivi valori.

Console

Per visualizzare i valori dei parametri per un gruppo di parametri del cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   Il gruppo di parametri cluster database viene visualizzato nell'elenco con Type (Tipo) impostato su DB cluster parameter group (Gruppo di parametri cluster database).
3. Scegliere il nome del gruppo di parametri cluster di database per visualizzare il relativo elenco di parametri.

AWS CLI

Per visualizzare i valori dei parametri per un gruppo di parametri del cluster DB, usa il AWS CLI `describe-db-cluster-parameters`comando con il seguente parametro obbligatorio.

- `--db-cluster-parameter-group-name`

Example

L'esempio seguente elenca i parametri e i valori dei parametri per un gruppo di parametri cluster di database denominato mydbclusterparametergroup, in formato JSON.

Questo comando restituisce una risposta simile alla seguente:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
    "Parameters": [
```

```
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
        {
            "ParameterName": "aurora_binlog_read_buffer_size",
            "ParameterValue": "5242880",
            "Description": "Read buffer size used by master dump thread when the switch
 aurora_binlog_use_large_read_buffer is ON.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "8192-536870912",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },

...
```

API RDS

Per visualizzare i valori dei parametri per un gruppo di parametri cluster database, utilizza il comando API RDS `DescribeDBClusterParameters` con il parametro obbligatorio seguente.

- `DBClusterParameterGroupName`

In alcuni casi, i valori consentiti per un parametro non vengono visualizzati. Questi sono sempre parametri in cui l'origine è l'impostazione predefinita del motore di database.

Per visualizzare i valori di questi parametri, puoi eseguire le seguenti istruzioni SQL:

- MySQL:

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL:

```
-- Show the value of a particular parameter
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

Eliminazione di un gruppo di parametri del cluster DB in Amazon Aurora

Puoi eliminare un gruppo di parametri del cluster DB utilizzando l' AWS Management Console API AWS CLI, o RDS. Un gruppo di parametri del gruppo di parametri del cluster DB può essere eliminato solo se non è associato a un cluster DB.

Console

Per eliminare i gruppi di parametri

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri vengono visualizzati in un elenco.
3. Scegli il nome dei gruppi di parametri del cluster DB da eliminare.
4. Scegli Azioni e poi Elimina.
5. Controllate i nomi dei gruppi di parametri, quindi scegliete Elimina.

AWS CLI

Per eliminare un gruppo di parametri del cluster DB, utilizzate il AWS CLI `delete-db-cluster-parameter-group`comando con il seguente parametro richiesto.

- `--db-parameter-group-name`

Example

L'esempio seguente elimina un gruppo di parametri del cluster DB denominato mydbparametergroup.

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

API RDS

Per eliminare un gruppo di parametri del cluster DB, utilizzate il `DeleteDBClusterParameterGroup`comando API RDS con il seguente parametro obbligatorio.

- `DBParameterGroupName`

Gruppi di parametri DB per DB Amazon Aurora

Le istanze database utilizzano gruppi di parametri database. Le sezioni seguenti descrivono la configurazione e la gestione dei gruppi di parametri dell'istanza database.

Argomenti

- Creazione di un gruppo di parametri DB in Amazon Aurora
- Associazione di un gruppo di parametri DB a un'istanza DB in
- Modifica dei parametri in un gruppo di parametri DB in
- Reimpostazione dei parametri in un gruppo di parametri DB ai valori predefiniti in Amazon
- Copia di un gruppo di parametri DB in
- Elenco dei gruppi di parametri DB in Amazon Aurora
- Visualizzazione dei valori dei parametri per un gruppo di parametri DB in Aurora
- Eliminazione di un gruppo di parametri DB in

Creazione di un gruppo di parametri DB in Amazon Aurora

Puoi creare un nuovo gruppo di parametri DB utilizzando il AWS Management Console, il AWS CLI, o il RDSAPI.

Le seguenti limitazioni si applicano al nome del gruppo di parametri database:

- Il nome deve contenere da 1 a 255 lettere, numeri o trattini.

  I nomi del gruppo di parametri predefiniti possono includere un punto, ad esempio `default.mysql8.0`. Tuttavia, i nomi del gruppo di parametri personalizzati non possono includere un punto.
- Il primo carattere deve essere una lettera.
- Il nome non può terminare con un trattino o contenere due trattini consecutivi.

Console

Per creare un gruppo di parametri del database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliParameter groups (Gruppi di parametri).
3. Scegli Create parameter group (Crea gruppo di parametri).
4. Per il nome del gruppo di parametri, inserisci il nome del tuo nuovo gruppo di parametri DB.
5. In Descrizione, inserisci una descrizione per il tuo nuovo gruppo di parametri DB.
6. Per Tipo di motore, scegli il tuo motore DB.
7. Per Famiglia di gruppi di parametri, scegli una famiglia di gruppi di parametri DB.
8. Per Tipo, se applicabile, scegliete DB Parameter Group.
9. Scegli Create (Crea).

AWS CLI

Per creare un gruppo di parametri DB, utilizzate il AWS CLI `create-db-parameter-group`comando. L'esempio seguente crea un gruppo di parametri DB denominato mydbparametergroup per My SQL version 8.0 con una descrizione di "Il mio nuovo gruppo di parametri».

Includi i parametri obbligatori seguenti:

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Per elencare tutte le famiglie del gruppo di parametri disponibili, usa il comando seguente:

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> L'output contiene duplicati.

Example

Per LinuxmacOS, oUnix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new parameter group"
```

Per Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new parameter group"
```

Questo comando genera un output simile al seguente:

```
DBPARAMETERGROUP  mydbparametergroup  aurora-mysql5.7  My new parameter group
```

RDS API

Per creare un gruppo di parametri DB, utilizzare l'RDSAPI`CreateDBParameterGroup`operazione.

Includi i parametri obbligatori seguenti:

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Associazione di un gruppo di parametri DB a un'istanza DB in

Puoi creare i tuoi gruppi di parametri database con impostazioni personalizzate. Puoi associare un gruppo di parametri DB a un'istanza DB utilizzando il AWS Management Console AWS CLI, il o il RDSAPI. Ciò è possibile quando crei o modifichi un'istanza database.

Per ulteriori informazioni sulla creazione di un gruppo di parametri del database, consulta Creazione di un gruppo di parametri DB in Amazon Aurora. Per ulteriori informazioni sulla modifica di un'istanza database, consulta Modifica di un'istanza database in un cluster database.

> **Note**
>
> Quando si associa un nuovo gruppo parametri del database a un'istanza database, i parametri statici e dinamici modificati vengono applicati solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio.

Console

Per associare un gruppo di parametri del database a un'istanza database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e selezionare l'istanza database da modificare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
4. Modifica l'impostazione del gruppo di parametri database .
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. (Facoltativo) Scegliere Applica immediatamente per applicare immediatamente le modifiche. In alcuni casi, la chiusura di questa opzione può causare un'interruzione.
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, seleziona Modifica istanza database per salvare le modifiche.

   Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per associare un gruppo di parametri DB a un'istanza DB, usa il AWS CLI `modify-db-instance`comando con le seguenti opzioni:

- `--db-instance-identifier`
- `--db-parameter-group-name`

Nell'esempio seguente il gruppo di parametri database `mydbpg` viene associato all'istanza database `database-1`. Le modifiche vengono applicate immediatamente tramite `--apply-immediately`. Utilizza `--no-apply-immediately` per applicare le modifiche durante la successiva finestra di manutenzione.

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

RDS API

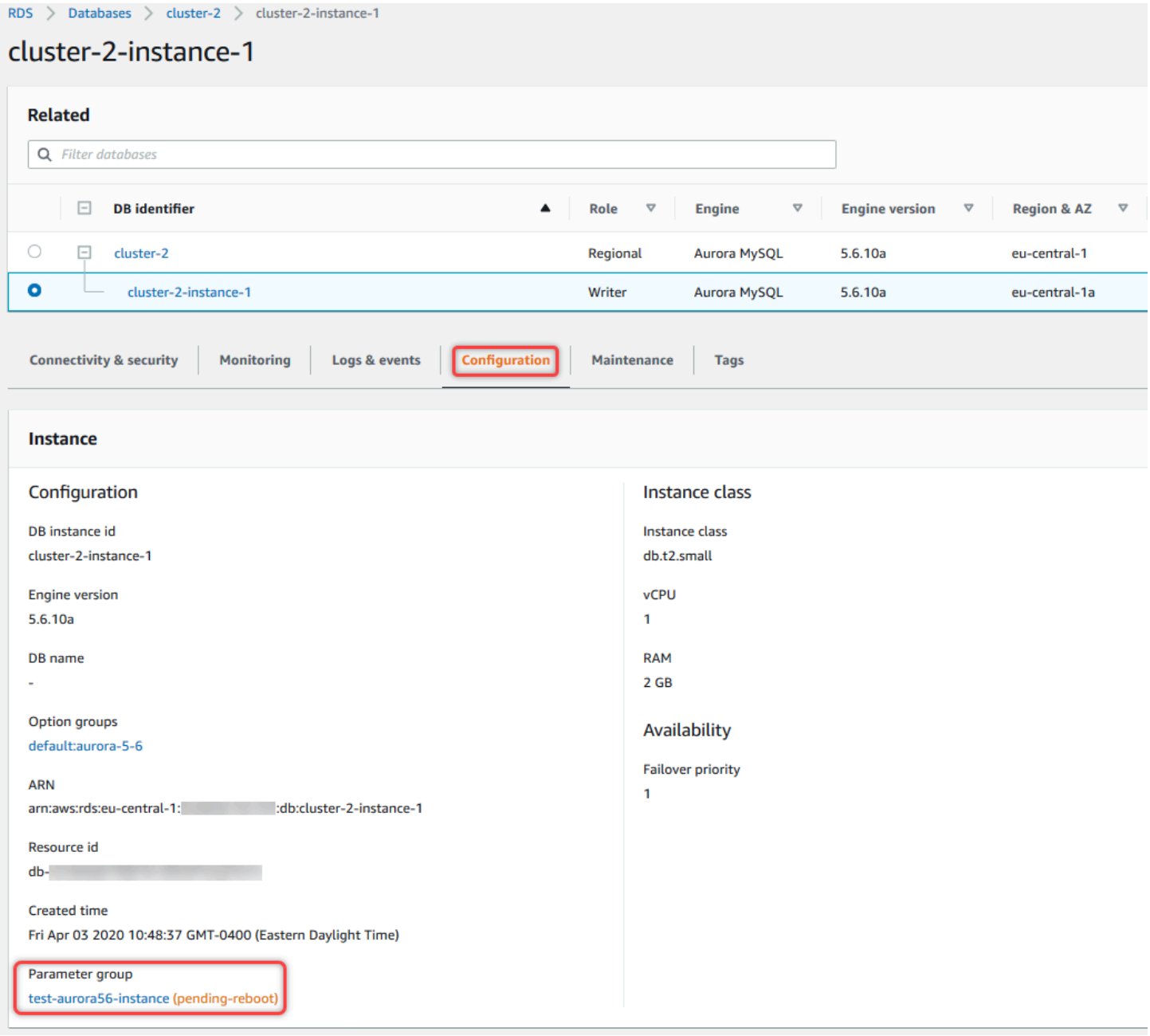
Per associare un gruppo di parametri DB a un'istanza DB, utilizzate l'RDSAPI`ModifyDBInstance`operazione con i seguenti parametri:

- `DBInstanceName`
- `DBParameterGroupName`

Modifica dei parametri in un gruppo di parametri DB in

Puoi modificare i valori dei parametri in un gruppo di parametri database creato dal cliente, ma non puoi modificare i valori dei parametri in un gruppo di parametri database predefinito. Le modifiche ai parametri in un gruppo di parametri database creato dal cliente vengono applicate a tutte le istanze database associate al gruppo di parametri database.

Le modifiche apportate ad alcuni parametri vengono applicate all'istanza database immediatamente senza un riavvio. Le modifiche apportate ad altri parametri vengono applicate solo dopo che l'istanza database viene riavviata. La console RDS mostra lo stato del gruppo di parametri database associato a un'istanza database nella scheda Configuration (Configurazione). Supponi, ad esempio che l'istanza database non utilizzi le modifiche più recenti apportate al gruppo di parametri database associato. In questo caso, la console RDS mostra il gruppo di parametri database con lo stato pending-reboot. Per applicare le ultime modifiche del parametro su quella istanza database, riavvia manualmente l’istanza database. Ti consigliamo di riavviare sempre l'istanza DB dopo aver modificato i parametri. Se non riavvii l'istanza DB, un'operazione di failover potrebbe richiedere più tempo del normale.



Console

Per modificare i parametri in un gruppo di parametri DB

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegli il nome del gruppo di parametri che desideri modificare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).

5. Modificare i valori dei parametri desiderati. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile modificare valori nel gruppo di parametri predefinito.

6. Scegli Save changes (Salva modifiche).

AWS CLI

Per modificare un gruppo di parametri DB, utilizzate il AWS CLI `modify-db-parameter-group`comando con le seguenti opzioni richieste:

- `--db-parameter-group-name`
- `--parameters`

L'esempio seguente modifica i valori `max_connections` e `max_allowed_packet` nel gruppo di parametri database denominato mydbparametergroup.

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBPARAMETERGROUP  mydbparametergroup
```

API RDS

Per modificare un gruppo parametri del database, utilizza l'operazione `ModifyDBParameterGroup` dell'API RDS con i seguenti parametri obbligatori:

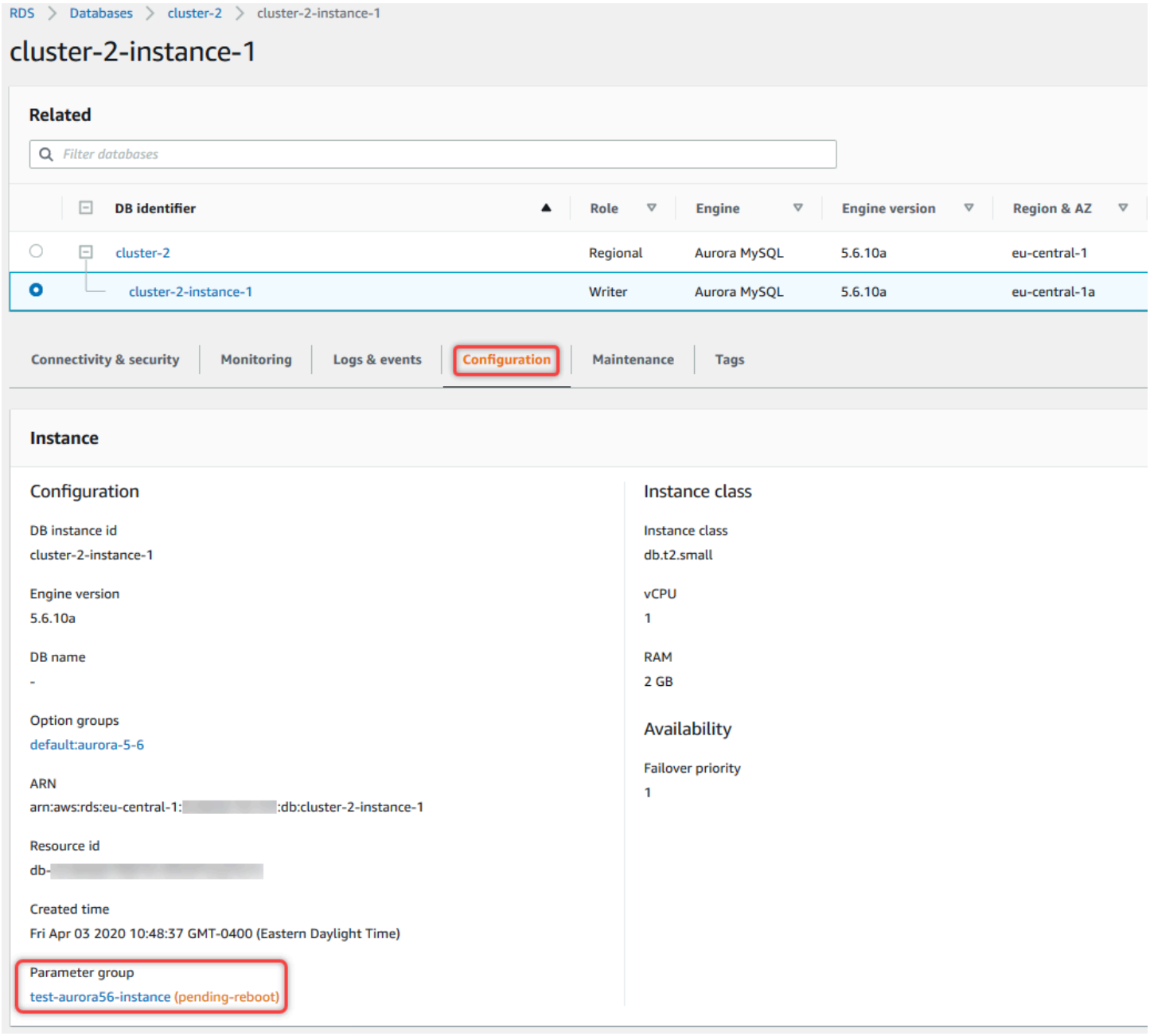
- `DBParameterGroupName`
- `Parameters`

Reimpostazione dei parametri in un gruppo di parametri DB ai valori predefiniti in Amazon

Puoi reimpostare i valori dei parametri in un gruppo di parametri database creato dal cliente sui valori predefiniti. Le modifiche ai parametri in un gruppo di parametri database creato dal cliente vengono applicate a tutte le istanze database associate al gruppo di parametri database.

Quando utilizzi la console, puoi reimpostare specifici parametri sui valori predefiniti. Tuttavia, non è possibile reimpostare altrettanto facilmente tutti i parametri nel gruppo di parametri database contemporaneamente. Quando usi AWS CLI o RDSAPI, puoi ripristinare i valori predefiniti di parametri specifici. Puoi anche reimpostare tutti i parametri del gruppo di parametri database contemporaneamente.

Le modifiche apportate ad alcuni parametri vengono applicate all'istanza database immediatamente senza un riavvio. Le modifiche apportate ad altri parametri vengono applicate solo dopo che l'istanza database viene riavviata. La RDS console mostra lo stato del gruppo di parametri DB associato a un'istanza DB nella scheda Configurazione. Supponi, ad esempio che l'istanza database non utilizzi le modifiche più recenti apportate al gruppo di parametri database associato. In tal caso, la RDS console mostra il gruppo di parametri DB con lo stato di pending-reboot. Per applicare le ultime modifiche del parametro su quella istanza database, riavvia manualmente l'istanza database.



> **Note**
>
> In un gruppo di parametri database predefinito, i parametri vengono sempre impostati sui valori di default.

Console

Per ripristinare i valori predefiniti dei parametri di un gruppo di parametri database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Scegliere i parametri per i quali si desidera ripristinare i valori predefiniti. È possibile scorrere i parametri usando i tasti freccia in alto a destra nella finestra di dialogo.

   Non è possibile reimpostare valori in un gruppo di parametri predefinito.
6. Scegli Reimposta , quindi conferma selezionando Ripristina parametri.

AWS CLI

Per ripristinare alcuni o tutti i parametri in un gruppo di parametri DB, usa il AWS CLI `reset-db-parameter-group`comando con la seguente opzione richiesta:`--db-parameter-group-name`.

Per reimpostare tutti i parametri nel gruppo di parametri database, specifica l'opzione `--reset-all-parameters`. Per reimpostare parametri specifici, specifica l'opzione `--parameters`.

Nell'esempio seguente tutti i parametri del gruppo di parametri del database denominato mydbparametergroup vengono reimpostati sui valori predefiniti.

Example

Per LinuxmacOS, oUnix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Per Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
```

```
    --reset-all-parameters
```

Nel seguente esempio le opzioni `max_connections` e `max_allowed_packet` vengono reimpostate sui loro valori predefiniti nel gruppo di parametri database denominato mydbparametergroup.

Example

Per LinuxmacOS, oUnix:

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Per Windows:

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Il comando genera un output simile al seguente:

```
DBParameterGroupName  mydbparametergroup
```

RDS API

Per ripristinare i parametri in un gruppo di parametri DB ai valori predefiniti, utilizzare il RDS API `ResetDBParameterGroup`comando con il seguente parametro richiesto:`DBParameterGroupName`.

Per reimpostare tutti i parametri nel gruppo di parametri database, imposta il parametro `ResetAllParameters` su `true`. Per reimpostare parametri specifici, specifica il parametro `Parameters`.

Copia di un gruppo di parametri DB in

Puoi copiare i gruppi di parametri database personalizzati che hai creato. La copia di un gruppo di parametri può essere una soluzione utile. Ad esempio quando crei un gruppo di parametri database

e vuoi includere la maggior parte dei parametri e dei valori personalizzati in un nuovo gruppo di parametri del database. Puoi copiare un gruppo di parametri DB utilizzando AWS Management Console. È inoltre possibile utilizzare il AWS CLI copy-db-parameter-groupcomando o l'operazione RDS API C opyDBParameter Group.

Dopo aver copiato un gruppo di parametri database, attendi almeno 5 minuti prima di creare la prima istanza database che usa il gruppo di parametri database come predefinito. In questo modo Amazon può RDS completare completamente l'azione di copia prima di utilizzare il gruppo di parametri. Questo è particolarmente importante per parametri critici durante la creazione del database predefinito per un'istanza database. Un esempio è il set di caratteri per il database predefinito definito dal parametro `character_set_database`. Usa l'opzione Parameter Groups della RDSconsole Amazon o il describe-db-parameterscomando per verificare che il tuo gruppo di parametri DB sia stato creato.

> **Note**
>
> Non puoi copiare un gruppo di parametri predefinito. Tuttavia, puoi creare un nuovo gruppo di parametri basato su un gruppo di parametri predefinito.
> Non puoi copiare un gruppo di parametri DB in un altro Account AWS o Regione AWS.

Console

Per copiare un gruppo di parametri del database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere il gruppo di parametri personalizzato da copiare.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Copy (Copia).
5. In New DB parameter group identifier (Identificatore nuovo gruppo di parametri database) immettere un nome per il nuovo gruppo di parametri.
6. Nella casella Description (Descrizione), immettere una descrizione per il nuovo gruppo di parametri.
7. Scegliere Copy (Copia).

AWS CLI

Per copiare un gruppo di parametri DB, usa il AWS CLI `copy-db-parameter-group`comando con le seguenti opzioni obbligatorie:

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

L'esempio seguente crea un nuovo gruppo di parametri database denominato `mygroup2`, che è una copia del gruppo di parametri database `mygroup1`.

Example

Per LinuxmacOS, oUnix:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Per Windows:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

RDS API

Per copiare un gruppo di parametri DB, utilizzate l'RDSAPI`CopyDBParameterGroup`operazione con i seguenti parametri obbligatori:

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Elenco dei gruppi di parametri DB in Amazon Aurora

Puoi elencare i gruppi di parametri DB che hai creato per il tuo AWS account.

> **Note**
>
> I gruppi di parametri predefiniti vengono creati automaticamente da un modello di parametro predefinito quando crei un'istanza database per un motore e una versione di database specifici. Questi gruppi di parametri predefiniti contengono le impostazioni dei parametri preferite e non possono essere modificati. Quando crei un gruppo di parametri personalizzato, puoi modificare le impostazioni dei parametri.

Console

Per elencare tutti i gruppi di parametri DB per un AWS account

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri database vengono visualizzati in un elenco.

AWS CLI

Per elencare tutti i gruppi di parametri DB per un AWS account, usa il AWS CLI `describe-db-parameter-groups`comando.

Example

L'esempio seguente elenca tutti i gruppi di parametri database disponibili per un account AWS .

```
aws rds describe-db-parameter-groups
```

Questo comando restituisce una risposta simile alla seguente:

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

L'esempio seguente descrive il gruppo di parametri mydbparamgroup1.

Per LinuxmacOS, oUnix:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Per Windows:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

Questo comando restituisce una risposta simile alla seguente:

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

RDS API

Per elencare tutti i gruppi di parametri DB per un AWS account, utilizzare l'RDSAPIDescribeDBParameterGroupsoperazione.

Visualizzazione dei valori dei parametri per un gruppo di parametri DB in Aurora

Puoi ottenere un elenco di tutti i parametri in un gruppo di parametri del database e dei rispettivi valori.

Console

Per visualizzare i valori dei parametri per un gruppo di parametri del database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri database vengono visualizzati in un elenco.
3. Scegliere il nome del gruppo di parametri per visualizzarne l'elenco di parametri.

AWS CLI

Per visualizzare i valori dei parametri per un gruppo di parametri DB, usa il AWS CLI describe-db-parameterscomando con il seguente parametro obbligatorio.

- `--db-parameter-group-name`

Example

L'esempio seguente elenca i parametri e i valori dei parametri per un gruppo di parametri database denominato mydbparametergroup.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

Questo comando restituisce una risposta simile alla seguente:

```
DBPARAMETER  Parameter Name            Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                   engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                      engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size         32768            system           integer
 dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system           string
 static      false
```

RDS API

Per visualizzare i valori dei parametri per un gruppo di parametri DB, utilizzate il RDS API `DescribeDBParameters`comando con il seguente parametro richiesto.

- `DBParameterGroupName`

Eliminazione di un gruppo di parametri DB in

Puoi eliminare un gruppo di parametri DB utilizzando AWS Management Console AWS CLI, o RDSAPI. Un gruppo di parametri può essere eliminato solo se non è associato a un'istanza DB.

Console

Per eliminare un gruppo di parametri DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).

   I gruppi di parametri database vengono visualizzati in un elenco.
3. Scegli il nome dei gruppi di parametri da eliminare.
4. Scegli Azioni e poi Elimina.
5. Controllate i nomi dei gruppi di parametri, quindi scegliete Elimina.

AWS CLI

Per eliminare un gruppo di parametri DB, utilizzate il AWS CLI `delete-db-parameter-group`comando con il seguente parametro richiesto.

- `--db-parameter-group-name`

Example

L'esempio seguente elimina un gruppo di parametri DB denominato mydbparametergroup.

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

RDS API

Per eliminare un gruppo di parametri DB, utilizzate il RDS API `DeleteDBParameterGroup`comando con il seguente parametro obbligatorio.

- `DBParameterGroupName`

Confronto di gruppi di parametri database

È possibile utilizzare il AWS Management Console per visualizzare le differenze tra due gruppi di parametri DB.

I gruppi di parametri specificati devono essere entrambi gruppi di parametri database o gruppi di parametri cluster database, anche se il motore e la versione del database sono uguali. Ad esempio, non è possibile confrontare un gruppo di parametri DB `aurora-mysql8.0` (Aurora My SQL versione 3) e un gruppo di parametri del cluster `aurora-mysql8.0` DB.

È possibile confrontare i gruppi di parametri Aurora My SQL e RDS for My SQL DB, anche per versioni diverse, ma non è possibile confrontare i gruppi di parametri Aurora Postgre e for SQL Postgre RDS DB. SQL

Per confrontare due gruppi di parametri DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegliere i due gruppi di parametri da confrontare.

   > **Note**
   >
   > Per confrontare un gruppo di parametri predefinito con un gruppo di parametri personalizzato, scegli prima il gruppo di parametri predefinito nella scheda Predefinito, quindi scegli il gruppo di parametri personalizzati nella scheda Personalizzato.

4. Da Azioni, scegliete Confronta.

Specifica dei parametri del database

I tipi di parametri database comprendono:

- Numero intero
- Boolean
- Stringa
- Long
- Doppio
- Timestamp
- Oggetto di altri tipi di dati definiti
- Array di valori di tipo integer, booleano, string, long, double, timestamp o oggetto

È inoltre possibile specificare parametri interi e booleani utilizzando espressioni, formule e funzioni.

Indice

- Formule dei parametri database
  - Variabili di formula dei parametri database
  - Operatori delle formule dei parametri database
- Funzioni dei parametri database
- Espressioni di log di parametri database
- Esempi di valori dei parametri database

Formule dei parametri database

Una formula per un parametro database è un'espressione che restituisce un valore intero o un valore booleano. L'espressione va racchiusa tra parentesi graffe: {}. Puoi utilizzare una formula per un valore di parametro database o come argomento per una funzione di parametro database.

Sintassi

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

Variabili di formula dei parametri database

Ogni variabile di formula restituisce un valore intero o booleano. I nomi delle variabili fanno distinzione tra maiuscole e minuscole.

AllocatedStorage

Restituisce un numero intero che rappresenta la dimensione, in byte, del volume di dati.

DBInstanceClassMemory

Restituisce un numero intero per il numero di byte di memoria disponibili per il processo del database. Questo numero viene calcolato internamente a partire dalla quantità totale di memoria per la classe di istanza database. Da ciò, il calcolo sottrae la memoria riservata al sistema operativo e ai processi che gestiscono l'istanza. RDS Pertanto, il numero è sempre leggermente

inferiore alle figure di memoria mostrate nelle tabelle della classe di istanza in Classi di istanze DB Amazon Aurora. Il valore esatto dipende da una combinazione di fattori: Questi includono la classe di istanza, il motore DB e se si applica a un'RDSistanza o a un'istanza che fa parte di un cluster Aurora.

DBInstanceVCPU

Restituisce un numero intero che rappresenta il numero di unità di elaborazione centrale virtuali (vCPUs) utilizzate da Amazon RDS per gestire l'istanza.

EndPointPort

Restituisce un numero intero che rappresenta la porta utilizzata durante la connessione all'istanza database.

TrueIfReplica

Restituisce 1 se l'istanza database è una replica di lettura e `0` in caso contrario. Questo è il valore predefinito per il `read_only` parametro in Aurora My. SQL

Operatori delle formule dei parametri database

Le formule dei parametri database supportano due operatori, di divisione e di moltiplicazione.

Operatore di divisione: /

Divide il dividendo per il divisore, restituendo un quoziente intero. I decimali nel quoziente vengono troncati, non arrotondati.

Sintassi

```
dividend / divisor
```

Gli argomenti del dividendo e del divisore devono essere espressioni intere.

Operatore di moltiplicazione: *

Moltiplica le espressioni, restituendone il prodotto. I decimali nelle espressioni vengono troncati, non arrotondati.

Sintassi

```
expression * expression
```

Entrambe le espressioni devono essere valori interi.

Funzioni dei parametri database

Puoi specificare gli argomenti delle funzioni dei parametri database come numeri interi o formule. Ogni funzione deve avere almeno un argomento. Specifica più argomenti come elenco separato da virgole. L'elenco non può includere membri vuoti, come argomento1,,argomento3. I nomi di funzione non fanno distinzione tra maiuscole e minuscole.

IF

Restituisce un argomento.

Sintassi

```
IF(argument1, argument2, argument3)
```

Restituisce il secondo argomento se il primo argomento restituisce true. In caso contrario, restituisce il terzo argomento.

GREATEST

Restituisce il valore più grande da un elenco di valori interi o formule di parametro.

Sintassi

```
GREATEST(argument1, argument2,...argumentn)
```

Restituisce un integer.

LEAST

Restituisce il valore più piccolo da un elenco di valori interi o formule di parametro.

Sintassi

```
LEAST(argument1, argument2,...argumentn)
```

Restituisce un integer.

SUM

Aggiunge i valori delle formule di parametro o dei numeri interi specificati.

Sintassi

```
SUM(argument1, argument2,...argumentn)
```

Restituisce un integer.

Espressioni di log di parametri database

Puoi impostare un valore del parametro database intero in una espressione di log. L'espressione va racchiusa tra parentesi graffe: {}. Ad esempio:

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

La funzione `log` rappresenta la base di log 2. In questo esempio viene utilizzato anche la variabile di formula `DBInstanceClassMemory`. Per informazioni, consulta Variabili di formula dei parametri database.

Esempi di valori dei parametri database

Questi esempi illustrano l'utilizzo di formule, funzioni ed espressioni per i valori dei parametri database.

> **Warning**
>
> L'impostazione errata dei parametri in un gruppo di parametri database può avere effetti negativi non intenzionali. Questi potrebbero includere prestazioni ridotte e instabilità del sistema. Presta sempre attenzione quando modifichi i parametri database ed esegui il backup dei dati prima di modificare il gruppo di parametri database. Prova le modifiche ai gruppi di parametri su un'istanza DB di test, creata utilizzando point-in-time-restores, prima di applicare tali modifiche al gruppo di parametri alle istanze DB di produzione.

Example utilizzando la funzione dei parametri DB LEAST

È possibile specificare la `LEAST` funzione in un valore del SQL `table_definition_cache` parametro Aurora My. Usalo per impostare il numero di definizioni di tabella che possono essere memorizzate nella cache delle definizioni sul valore minimo di `DBInstanceClassMemory`/393040 o 20.000.

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Migrazione di dati a un cluster di database Amazon Aurora

Per la migrazione dei dati da un database esistente a un cluster database Amazon Aurora sono disponibili diverse opzioni, a seconda della compatibilità del motore di database. Le opzioni di migrazione dipendono anche dal database da cui esegui la migrazione e dalle dimensioni dei dati sottoposti a migrazione.

Migrazione dei dati verso un cluster Amazon Aurora My DB SQL

Puoi migrare i dati da una delle seguenti fonti a un cluster Amazon Aurora SQL My DB.

- E RDS per l'istanza My DB SQL
- A: Il mio SQL database esterno ad Amazon RDS
- Un database che non è compatibile SQL con My

Per ulteriori informazioni, consulta Migrazione dei dati verso un cluster Amazon Aurora My DB SQL.

Migrazione dei dati verso un cluster Amazon Aurora SQL Postgre DB

Puoi migrare i dati da una delle seguenti fonti a un cluster Amazon Aurora SQL Postgre DB.

- Un'istanza Amazon RDS Postgree DB SQL
- Un database non compatibile con Postgre SQL

Per ulteriori informazioni, consulta Migrazione dei dati su Amazon Aurora con compatibilità Postgre SQL.

Creazione di una ElastiCache cache Amazon utilizzando le impostazioni dell'istanza Amazon DB del cluster Aurora RDS

ElastiCache è un servizio di caching in memoria completamente gestito che fornisce latenze di lettura e scrittura in microsecondi che supportano casi d'uso flessibili e in tempo reale. ElastiCache può aiutarti ad accelerare le prestazioni di applicazioni e database. È possibile utilizzarlo ElastiCache come archivio dati primario per casi d'uso che non richiedono la durabilità dei dati, ad esempio classiifiche di gioco, streaming e analisi dei dati. ElastiCache aiuta a rimuovere la complessità associata all'implementazione e alla gestione di un ambiente di elaborazione distribuito. Per ulteriori informazioni, consulta Casi ElastiCache d'uso comuni e How ElastiCache Can Help for Memcached and Common ElastiCache Use Cases e How ElastiCache Can Help for Redis. OSS Puoi utilizzare la RDS console Amazon per creare ElastiCache cache.

Puoi utilizzare Amazon ElastiCache in due formati. Puoi iniziare con una cache serverless o scegliere di progettare il tuo cluster di cache. Se scegli di progettare il tuo cluster di cache, ElastiCache funziona con entrambi i motori Redis OSS e Memcached. Se non sei sicuro del motore da usare, consulta Confronto tra Memcached e RedisOSS. Per ulteriori informazioni su Amazon ElastiCache, consulta la Amazon ElastiCache User Guide.

Argomenti

- Panoramica della creazione di ElastiCache cache con le impostazioni dell'istanza DB del cluster Aurora RDS
-

Panoramica della creazione di ElastiCache cache con le impostazioni dell'istanza DB del cluster Aurora RDS

Alcuni casi d'uso per associare una ElastiCache cache all':

- È possibile risparmiare sui costi e migliorare le prestazioni utilizzando ElastiCache con RDS anziché eseguendolo RDS singolarmente.
- È possibile utilizzare la ElastiCache cache come archivio dati principale per applicazioni che non richiedono la durabilità dei dati. Le applicazioni che utilizzano Redis OSS o Memcached possono essere utilizzate quasi ElastiCache senza modifiche.

Quando si crea una ElastiCache cache daRDS, la ElastiCache cache eredita le seguenti impostazioni dall'istanza DB del Aurora DB associata:

- ElastiCache impostazioni di connettività
- ElastiCache impostazioni di sicurezza

È inoltre possibile configurare le impostazioni di configurazione della cache in base alle proprie esigenze.

Configurazione ElastiCache nelle tue applicazioni

Le applicazioni devono essere configurate per utilizzare la ElastiCache cache. È inoltre possibile ottimizzare e migliorare le prestazioni della cache configurando le applicazioni in modo che utilizzino strategie di memorizzazione nella cache in base alle proprie esigenze.

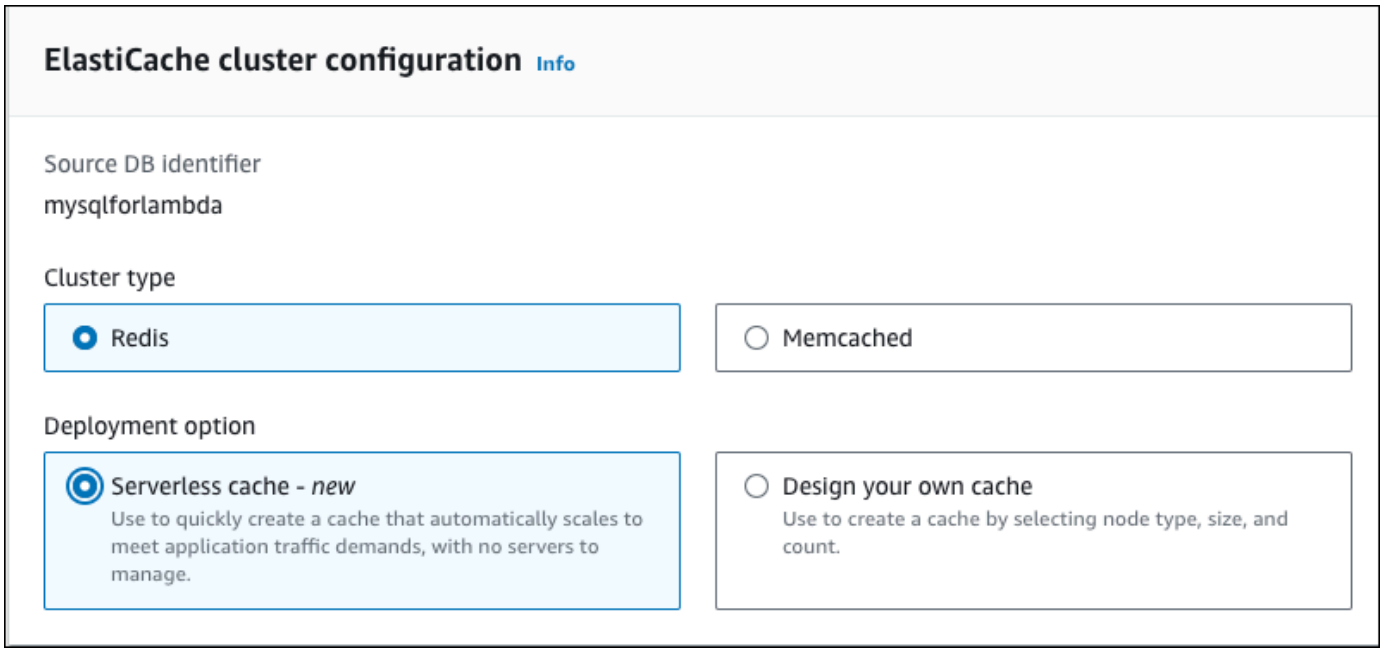
- Per accedere alla ElastiCache cache e iniziare, consulta Guida introduttiva a ElastiCache (RedisOSS) e Guida introduttiva a ElastiCache (Memcached).
- Per ulteriori informazioni sulle strategie di caching, consulta Strategie e best practice di caching per Memcached e Strategie e best practice di caching per Redis. OSS
- Per ulteriori informazioni sull'alta disponibilità nei cluster ElastiCache (RedisOSS), consulta Alta disponibilità utilizzando gruppi di replica.
- Potrebbero essere sostenuti costi associati allo storage di backup, al trasferimento dei dati all'interno o tra regioni o all'uso di. AWS Outposts Per i dettagli sui prezzi, consulta la pagina ElastiCache dei prezzi di Amazon.

1. Per creare un cluster di database, segui le istruzioni in Creazione di un cluster database Amazon Aurora.
2. Dopo aver creato un', la console visualizza la finestra Componenti aggiuntivi consigliati. Seleziona Crea un ElastiCache cluster RDS utilizzando le impostazioni del database.

   Per un database esistente, nella pagina Database, seleziona l' del cluster DB richiesta. Nel menu a discesa Azioni, scegli Crea ElastiCache cluster per creare una ElastiCache cache con le stesse impostazioni dell'istanza DB del Aurora DB esistente. RDS

Nella sezione di ElastiCache configurazione, l'identificatore Source DB mostra da quale del cluster DB la ElastiCache cache eredita le impostazioni.

3. Scegli se vuoi creare un cluster Redis OSS o Memcached. Per ulteriori informazioni, consulta Confronto tra Memcached e RedisOSS.



4. Dopodiché, scegli se vuoi creare una cache serverless o progettare la tua cache. Per ulteriori informazioni, consulta Scelta tra le opzioni di distribuzione.

   Se scegli Serverless cache:

   a. Nelle impostazioni della cache, inserisci i valori per Nome e Descrizione.

   b. In Visualizza impostazioni predefinite, lascia le impostazioni predefinite per stabilire la connessione tra la cache e l' del cluster DB.

   c. Puoi anche modificare le impostazioni predefinite scegliendo Personalizza impostazioni predefinite. Seleziona le impostazioni di ElastiCache connettività, le impostazioni ElastiCache di sicurezza e i limiti massimi di utilizzo.

5. Se scegli Progetta la tua cache:

   a. Se hai scelto il OSScluster Redis, scegli se vuoi mantenere la modalità cluster abilitata o disabilitata. Per ulteriori informazioni, consulta Replica: Redis OSS (modalità cluster disabilitata) e Redis OSS (modalità cluster abilitata).

   b. Immetti i valori per Nome, Descrizione e Versione del motore.

Per Versione del motore, il valore predefinito consigliato è la versione più recente del motore. Puoi anche scegliere una versione del motore per la ElastiCache cache che meglio soddisfa i tuoi requisiti.

c. Scegli il tipo di nodo per l'opzione Tipo di nodo. Per ulteriori informazioni, consulta Gestione di nodi.

   Se scegli di creare un OSS cluster Redis con la modalità Cluster impostata su Enabled, inserisci il numero di shard (partizioni/gruppi di nodi) nell'opzione Numero di shard.

   Immetti il numero di repliche di ogni partizione per l'opzione Numero di repliche.

   > **Note**
   >
   > Il tipo di nodo selezionato, il numero di shard e il numero di repliche influiscono tutti sulle prestazioni della cache e sui costi delle risorse. Assicurati che queste impostazioni corrispondano alle esigenze del tuo database. Per informazioni sui prezzi, consulta la pagina ElastiCache dei prezzi di Amazon.

d. Seleziona le impostazioni di ElastiCache connettività e le impostazioni ElastiCache di sicurezza. È possibile mantenere le impostazioni predefinite o personalizzarle in base alle proprie esigenze.

6. Verifica le impostazioni predefinite ed ereditate della ElastiCache cache. Alcune impostazioni non possono essere modificate dopo la creazione.

   > **Note**
   >
   > RDSpotrebbe modificare la finestra di backup della ElastiCache cache per soddisfare il requisito minimo di 60 minuti. La finestra di backup del database di origine rimane invariata.

7. Quando sei pronto, scegli Crea ElastiCache cache.

La console visualizza un banner di conferma per la creazione ElastiCache della cache. Segui il link contenuto nel banner verso la ElastiCache console per visualizzare i dettagli della cache. La ElastiCache console visualizza la ElastiCache cache appena creata.

Migrazione automatica EC2 dei database su Amazon Aurora AmazonAWS Database Migration Service

È possibile utilizzare la console Aurora per migrare un EC2 database in Aurora. Aurora utilizza AWS Database Migration Service (AWS DMS) per migrare il database di origine. EC2 AWS DMS ti consente di migrare i database relazionali nel tuo Cloud. AWS Per ulteriori informazioni su AWS Database Migration Service, consulta What is? AWS Database Migration Service nella Guida AWS Database Migration Service per l'utente.

Per iniziare la migrazione, è necessario creare un' database del cluster Aurora DB equivalente in cui migrare i dati. Dopo aver creato il database di destinazione, puoi importare il EC2 database al suo interno. Per i database di origine più piccoli di 1 TiB, questa azione di migrazione riduce il tempo e le risorse necessari per migrare i dati in Aurora.

Panoramica

La console Aurora consente di migrare i database in EC2 database Aurora equivalenti. È necessario creare un database Aurora per abilitare la migrazione dalla console.

È possibile migrare EC2 i database per i seguenti motori di database:

- Mio SQL
- Poster SQL

Il processo di migrazione prevede i seguenti passaggi:

- Crea un database equivalente in Aurora . Affinché i database siano equivalenti, devono avere lo stesso motore di database e versioni del motore compatibili. Devono inoltre essere ugualiVPC. Per istruzioni sulla creazione del database, vedere Creazione di un cluster database Amazon Aurora.
- Scegli il tipo di replica per il tuo database:
  - Migrazione a pieno carico: Aurora copia l'intero database di origine nel database di destinazione, creando nuove tabelle nella destinazione quando necessario.

    > **Note**
    >
    > Questa opzione causa un'interruzione nel database Aurora.

- Migrazione a pieno carico e modifica dell'acquisizione dei dati (CDC): simile alla migrazione a pieno carico, con questa opzione, Aurora copia l'intero database di origine nel database di destinazione. Tuttavia, dopo la migrazione a pieno carico, Aurora applica tutte le modifiche acquisite nell'origine al database di destinazione. Change Data Capture raccoglie le modifiche ai log del database utilizzando il motore di database nativo. API

  > **Note**
  >
  > Questa opzione causa un'interruzione nel database Aurora.

- Modifica acquisizione dati (CDC): utilizzate questa opzione per mantenere disponibile il database di destinazione durante la migrazione. Aurora migra le modifiche in corso nel database di origine nel database di destinazione.

- Aurora crea le risorse di rete necessarie per facilitare la migrazione. Una volta create le risorse richieste, Aurora notifica all'utente le risorse create e consente di avviare il trasferimento dei dati.

  Il tempo necessario per completare la migrazione dipende dal tipo di replica e dalla dimensione del database di origine.

Prerequisiti

My SQL

Prima di iniziare a utilizzare un database My SQL come database di origine, assicurati di avere i seguenti prerequisiti. Questi prerequisiti si applicano alle fonti gestite. AWS

È necessario disporre di un account con il AWS DMS ruolo di amministratore di replica. Il ruolo richiede i seguenti privilegi:

- REPLICATIONCLIENT— Questo privilegio è richiesto solo per le CDC attività. In altre parole, le full-load-only attività non richiedono questo privilegio.
- REPLICATIONSLAVE— Questo privilegio è richiesto solo per le CDC attività. In altre parole, le full-load-only attività non richiedono questo privilegio.

L' AWS DMS utente deve inoltre disporre dei SELECT privilegi per le tabelle di origine destinate alla replica.

Concedi i seguenti privilegi se utilizzi le valutazioni di premigrazione specifiche SQL di My.

```
grant select on mysql.user to <dms_user>;
grant select on mysql.db to <dms_user>;
grant select on mysql.tables_priv to <dms_user>;
grant select on mysql.role_edges to <dms_user>  #only for MySQL version 8.0.11 and
 higher
```

Postgree SQL

Prima di migrare i dati da un database SQL sorgente Postgre AWS gestito, procedi come segue:

- Ti consigliamo di utilizzare un account AWS utente con le autorizzazioni minime richieste per l'istanza SQL DB di Postgre come account utente per l'endpoint sorgente di Postgre per. SQL AWS DMS L'utilizzo di un account master è sconsigliato. L'account deve avere il ruolo `rds_superuser` e il ruolo `rds_replication`. Il ruolo `rds_replication` fornisce le autorizzazioni per gestire gli slot logici e per eseguire lo streaming dei dati utilizzando gli slot logici.

> **Note**
>
> Alcune AWS DMS transazioni rimangono inattive per qualche tempo prima che il motore le utilizzi nuovamente. DMS Utilizzando il parametro `idle_in_transaction_session_timeout` nelle SQL versioni 9.6 e successive di Postgre, puoi causare il timeout e il fallimento delle transazioni inattive.

Limitazioni

Le seguenti limitazioni si applicano al processo di migrazione automatica:

- Lo stato del database di destinazione deve essere Disponibile per iniziare la migrazione del database di origine.
- Quando si esegue la migrazione da un database My SQL source, l'account Aurora deve avere il ruolo di amministratore di replica. È inoltre necessario disporre dei privilegi appropriati per quel ruolo.
- L'EC2istanza e il database di destinazione devono trovarsi nello stesso VPC database.
- Non è possibile migrare il EC2 database ai seguenti database di destinazione quando si utilizza l'azione Migra dati dal EC2 database:
  - Aurora global database

- Aurora Limitless database
- Aurora Serverless v1
- Database con SQL versione My precedente alla 5.7
- Database con SQL versione Postgre precedente alla 10.4

Creazione di IAM risorse per migrazioni omogenee

Aurora lo utilizza AWS DMS per migrare i tuoi dati. Per accedere ai database e migrare i dati, AWS DMS crea un ambiente senza server per migrazioni di dati omogenee. In questo ambiente, AWS DMS richiede l'accesso a VPC peering, tabelle di routing, gruppi di sicurezza e altre risorse. AWS Inoltre, AWS DMS archivia log, metriche e progressi per ogni migrazione di dati in Amazon. CloudWatch Per creare un progetto di migrazione dei dati, è AWS DMS necessario accedere a questi servizi.

Inoltre, AWS DMS richiede l'accesso ai segreti che rappresentano un set di credenziali utente per autenticare la connessione al database per la connessione di origine e di destinazione.

> **Note**
>
> Utilizzando l'azione Migra dati dall'EC2istanza, è possibile utilizzare la console Aurora per generare IAM queste risorse. Salta questo passaggio se utilizzi le risorse generate dalla console. IAM

Per questo processo sono IAM necessarie le seguenti risorse:

Argomenti

- Creazione di una IAM politica per migrazioni omogenee dei dati
- Creazione di un IAM ruolo per migrazioni di dati omogenee
- Creazione di una politica e di un ruolo di accesso segreti
- Creare un IAM ruolo AWS DMS per gestire Amazon VPC

Creazione di una IAM politica per migrazioni omogenee dei dati

In questa fase, crei una IAM policy che AWS DMS fornisca l'accesso ad Amazon EC2 e alle CloudWatch risorse. Quindi, crea un IAM ruolo e allega questa politica.

Per creare una IAM politica per la migrazione dei dati

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella pagina Crea policy, scegli la JSONscheda.
5. Incolla quanto segue JSON nell'editor.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. Scegli Next: Tags (Successivo: Tag) e Next: Review (Successivo: Rivedi).
7. Immetti **`HomogeneousDataMigrationsPolicy`** per Nome* e scegli Crea policy.

Creazione di un IAM ruolo per migrazioni di dati omogenee

In questo passaggio, crei un IAM ruolo che fornisce l'accesso a AWS Secrets Manager EC2, Amazon e CloudWatch.

Per creare un IAM ruolo per le migrazioni dei dati

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Roles (Ruoli).
3. Selezionare Create role (Crea ruolo).
4. Nella pagina Seleziona un'entità attendibile scegli Servizio AWS per Tipo di entità attendibile. Per i casi d'uso per altri AWS servizi, scegli DMS.
5. Seleziona la DMScasella di controllo e scegli Avanti.

6. Nella pagina Aggiungi autorizzazioni, scegli HomogeneousDataMigrationsPolicyquello che hai creato in precedenza. Scegli Next (Successivo).
7. Nella pagina Nomina, verifica e crea immetti **`HomogeneousDataMigrationsRole`** per Nome ruolo e scegli Crea ruolo.
8. Nella pagina Ruoli immetti **`HomogeneousDataMigrationsRole`** per Nome ruolo. Scegli HomogeneousDataMigrationsRole.
9. Nella HomogeneousDataMigrationsRolepagina, scegli la scheda Relazioni di fiducia. Seleziona Modifica policy di attendibilità.
10. Nella pagina Modifica politica di fiducia, incolla quanto segue JSON nell'editor, sostituendo il testo esistente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Nell'esempio precedente, sostituiscilo *`your_region`* con il nome del tuo Regione AWS.

La precedente politica basata sulle risorse fornisce ai responsabili del AWS DMS servizio le autorizzazioni per eseguire attività in base alla politica gestita dal cliente. HomogeneousDataMigrationsPolicy

11. Scegli Aggiorna policy.

Creazione di una politica e di un ruolo di accesso segreti

Segui le procedure seguenti per creare la politica e il ruolo di accesso segreti che consentano di accedere DMS alle credenziali utente per i database di origine e di destinazione.

Per creare la politica e il ruolo di accesso segreti, che consentano RDS ad Amazon di accedere AWS Secrets Manager per accedere al segreto appropriato

1. Accedi a AWS Management Console e apri la console AWS Identity and Access Management (IAM) all'indirizzo https://console.aws.amazon.com/iam/.
2. Seleziona Policy, quindi scegli Crea policy.
3. Scegli JSONe inserisci la seguente politica per abilitare l'accesso e la decrittografia del tuo segreto.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": secret_arn,
        },
        {
             "Effect": "Allow",
             "Action": [
                        "kms:Decrypt",
                        "kms:DescribeKey"
                      ],
             "Resource": kms_key_arn,
        }
     ]
}
```

Qui *secret_arn* c'è il ARN tuo segreto, che puoi ottenere da entrambi `SecretsManagerSecretId` a seconda ARN dei casi, ed *kms_key_arn* è la AWS KMS chiave che stai usando per crittografare il tuo segreto, come nell'esempio seguente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
             "Effect": "Allow",
             "Action": [
                       "kms:Decrypt",
                       "kms:DescribeKey"
                     ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
     ]
}
```

> **Note**
>
> Se si utilizza la chiave di crittografia predefinita creata da AWS Secrets Manager, non è necessario specificare le AWS KMS autorizzazioni per. *kms_key_arn*
> Se desideri che la tua politica fornisca l'accesso a entrambi i segreti, specifica semplicemente un oggetto JSON risorsa aggiuntivo per l'altro*secret_arn*.

4. Rivedi e crea la policy con un nome descrittivo e, facoltativamente, una descrizione.
5. Seleziona Ruoli, quindi scegli Crea ruolo.
6. Per il tipo di entità attendibile, scegli Servizio AWS .
7. Scegli DMSdall'elenco dei servizi come servizio affidabile, quindi scegli Avanti: Autorizzazioni.
8. Cerca e collega la policy che hai creato nella fase 4, quindi procedi con l'aggiunta di eventuali tag ed esamina il ruolo. A questo punto, modifica le relazioni di trust per il ruolo in modo da utilizzare il responsabile del servizio RDS regionale Amazon come entità affidabile. Questo principale ha il seguente formato.

   ```
   dms.region-name.amazonaws.com
   ```

   Qui *region-name* è il nome della regione, ad esempio `us-east-1`. Pertanto, segue un responsabile del servizio RDS regionale Amazon per questa regione.

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

Creare un IAM ruolo AWS DMS per gestire Amazon VPC

È necessario creare un IAM ruolo AWS DMS per gestire le VPC impostazioni delle risorse. Questo ruolo deve essere disponibile per una migrazione corretta.

Creazione del file **`dms-vpc-role`** per la migrazione del database

<result>

Questo crea il ruolo DMS per la gestione delle VPC impostazioni per la migrazione.
</result>

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione della console, selezionare Roles (Ruoli) e Crea ruolo.
3. Scegli l'Servizio AWSopzione per l'opzione Seleziona un'entità affidabile.

   Per Caso d'uso, seleziona DMS.
4. Per il passaggio Aggiungi autorizzazioni, seleziona `AmazonDMSVPCManagementRole` e scegli Avanti.
5. Nella pagina Nome, rivedi e crea, imposta il nome del ruolo su `dms-vpc-role` e scegli Crea ruolo.

Configurazione della migrazione dei dati per il EC2 database

Per iniziare la migrazione dei dati dal database di EC2 origine, è necessario creare un database Aurora equivalente. Per istruzioni sulla creazione del database, consulta. Creazione di un cluster database Amazon Aurora

Dopo aver creato il database di destinazione, utilizza i seguenti passaggi per configurare la migrazione dei dati:

Configura il progetto di migrazione dei dati

1. Seleziona il database di destinazione nella pagina Database della RDS console.

2. Scegli il menu a discesa Azioni e seleziona l'opzione Migra dati dal EC2 database. Per visualizzare i database di destinazione supportati, consulta. Limitazioni
3. Nella sezione Seleziona il EC2 database di origine:

    1. Controlla il tipo di motore e assicurati che sia lo stesso del database di origine.

        Inoltre, controlla se le versioni del motore sono compatibili.
    2. Ad EC2esempio, scegli l'EC2istanza in cui risiede il database di origine.
    3. Per Porta, inserisci la porta su cui il database di origine consente il traffico.
    4. Per Segreto, scegli Crea e usa un nuovo segreto se non ne hai uno esistente. Inserisci il nome utente e la password per il database di origine. Scegli anche la KMS chiave con cui crittografare il tuo segreto.

        Se disponi di un segreto esistente, seleziona Usa un segreto esistente, quindi scegli segreto dal menu a discesa.
    5. Per IAMruolo per segreto, se disponi di un IAM ruolo esistente, seleziona Usa un IAM ruolo esistente e scegli un IAM ruolo dal menu a discesa che possa accedere all'ID segreto del passaggio precedente.

        Se non disponi di un IAM ruolo esistente, scegli Crea e usa un nuovo IAM ruolo. Inserisci il nuovo nome per il tuo ruolo come nome del IAM ruolo. Puoi vedere le autorizzazioni associate a questo ruolo nel link sottostante.
4. Nella sezione Visualizza il RDS database di destinazione:

    1. Conferma le impostazioni del database di destinazione nella parte superiore della sezione.
    2. Per Segreto, scegli Crea e usa un nuovo segreto se non disponi di un segreto esistente che contenga le credenziali del database di destinazione.

        Se disponi di un segreto esistente, seleziona il segreto dal menu a discesa.
    3. Per IAMruolo per segreto, seleziona un IAM ruolo che può accedere al segreto del passaggio precedente. Puoi anche creare un nuovo IAM ruolo se non hai un IAM ruolo esistente.

        Se il menu a discesa non compila i IAM ruoli, specifica il IAMruolo ARN nel formato. `arn:aws:iam:`*`account_id`*`:role/`*`roleName`*
5. Nella sezione Configura la migrazione dei dati:

1. Seleziona il tipo di migrazione dei dati selezionando tra Caricamento completo, Caricamento completo e modifica acquisizione dati (CDC) o Modifica acquisizione dati (CDC). Per ulteriori informazioni su queste opzioni, consulta Panoramica.

   Non è possibile modificare il tipo di migrazione dopo l'inizio della migrazione.
2. Per IAMil ruolo per la migrazione dei dati, se disponi di un IAM ruolo esistente, seleziona Usa un IAM ruolo esistente e scegli un IAM ruolo dal menu a discesa che conceda DMS le autorizzazioni per creare le risorse necessarie per la migrazione. Se non disponi di un IAM ruolo esistente, scegli Crea e usa un nuovo ruolo. IAM
6. Verifica che la scheda Visualizza impostazioni di migrazione mostri le impostazioni necessarie per configurare correttamente la migrazione dei dati.
7. Seleziona Migra per completare la configurazione della migrazione.

Dopo aver completato questi passaggi, puoi vedere le risorse configurate per la migrazione dei dati scegliendo Visualizza dettagli nel banner di avanzamento della console. Una volta configurate le risorse richieste, la migrazione viene avviata automaticamente. Se crei

Per migrare più database nel database di destinazione, riavvia questo processo con i dettagli sul nuovo EC2 database.

Gestione delle migrazioni dei dati

Dopo aver utilizzato l'azione Migra dati dal EC2 database dalla RDS console, Aurora avvia automaticamente la migrazione.

Se hai utilizzato la AWS DMS console per creare le risorse di migrazione, puoi avviare il processo di migrazione.

Avvio della migrazione dei dati

Segui questi passaggi per avviare la migrazione dei dati:

Avvio di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della RDS console.
2. Nella pagina dei dettagli del database, scegli la scheda Migrazioni dei dati.
3. Nella scheda Migrazioni dei dati, la sezione Migrazioni di dati associate elenca le migrazioni di dati disponibili.

Le migrazioni configurate utilizzando la console Aurora si avviano automaticamente una volta configurate le risorse richieste.

Le migrazioni configurate utilizzando la DMS console sono impostate su Ready.

Per iniziare queste migrazioni, seleziona il menu a discesa Azioni e seleziona Avvia.

4. Questo avvia la migrazione dei dati per il tuo EC2 database.

Interruzione della migrazione dei dati

Per le migrazioni di dati il cui tipo di replica è a pieno carico, l'interruzione della migrazione causa l'interruzione del processo e non può essere ripreso. Una volta interrotta, è necessario riavviare la migrazione.

Per le migrazioni con il tipo di replica impostato per modificare l'acquisizione dei dati (CDC) o il caricamento completoCDC, è possibile interrompere il processo di replica continua e riprenderlo in un secondo momento.

Interruzione della migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della RDS console.
2. Nella pagina dei dettagli del database, scegli la scheda Migrazioni dei dati.
3. Nella scheda Migrazioni dei dati, la sezione Migrazioni di dati associate elenca le migrazioni di dati in corso.

   Per interrompere una migrazione, seleziona una migrazione dei dati e seleziona Interrompi nel menu a discesa Azioni.
4. Ciò interrompe la migrazione dei dati per il EC2 database.

Ripresa della migrazione dei dati

Per le migrazioni di dati il cui tipo di replica è la migrazione a pieno carico e modifica acquisizione dati (CDC) o modifica acquisizione dati (CDC), è possibile riprendere il CDC processo dall'ultimo punto di arresto.

Ripresa della migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della RDS console.

2. Nella pagina dei dettagli del database, scegli la scheda Migrazioni dei dati.
3. Nella scheda Migrazioni dei dati, la sezione Migrazioni di dati associate elenca le migrazioni di dati interrotte.

   Per riprendere una migrazione, seleziona una migrazione dei dati e seleziona Riprendi l'elaborazione nel menu a discesa Azioni.
4. Questo riprende la migrazione dei dati per il tuo database. EC2

Eliminazione della migrazione dei dati

Per eliminare una migrazione di dati associata, utilizzare le seguenti istruzioni

Eliminazione di una migrazione di dati

1. Scegli il database di destinazione nella pagina Database della RDS console.
2. Nella pagina dei dettagli del database, scegli la scheda Migrazioni dei dati.
3. Per eliminare una migrazione, seleziona una migrazione di dati e seleziona Elimina nel menu a discesa Azioni.
4. Ciò elimina la migrazione dei dati.

L'eliminazione di una migrazione di dati in corso non ha alcun impatto sui dati già caricati nel database di destinazione.

Riavvio della migrazione dei dati

Per riavviare una migrazione di dati associata da un punto CDC iniziale, usa le seguenti istruzioni

Riavvio di una migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della RDS console.
2. Nella pagina dei dettagli del database, scegli la scheda Migrazioni dei dati.
3. Per riavviare una migrazione, seleziona una migrazione dei dati e seleziona Riavvia nel menu a discesa Azioni.
4. Ciò riavvia la migrazione dei dati da un punto CDC iniziale.

Il riavvio di una migrazione di dati in corso non ha alcun impatto sui dati già caricati nel database di destinazione.

Monitoraggio delle migrazioni dei dati

Dopo l'inizio delle migrazioni dei dati, puoi monitorarne lo stato e l'avanzamento. Le migrazioni di dati di set di dati di grandi dimensioni richiedono ore per essere completate. Per mantenere l'affidabilità, la disponibilità e le prestazioni elevate della migrazione dei dati, è necessario monitorarne regolarmente l'avanzamento.

Per verificare lo stato e l'avanzamento della migrazione dei dati

1. Scegli il database di destinazione nella pagina Database della RDS console.
2. Nella pagina dei dettagli del database, scegli la scheda Migrazioni dei dati.
3. La sezione Migrazioni di dati associate elenca le migrazioni dei dati. Controlla la colonna Status.
4. Per eseguire le migrazioni dei dati, la colonna Processo di migrazione mostra la percentuale di dati migrati.
5. Per monitorare il processo in CloudWatch, usa il link nella colonna in CloudWatch.

Stati di migrazione

Per ogni migrazione di dati eseguita, la console Aurora visualizza lo stato. L'elenco seguente include gli stati:

- `Ready`: La migrazione dei dati è pronta per iniziare.
- `Starting`: Aurora sta creando l'ambiente serverless per la migrazione dei dati.
- `Load running`: Aurora sta eseguendo la migrazione a pieno carico.
- `Load complete, replication ongoing`: Aurora ha completato il caricamento completo e ora replica le modifiche in corso. Questo stato si applica solo alle migrazioni a pieno carico e CDC di tipo.
- `Replication ongoing`: Aurora sta replicando le modifiche in corso. Questo stato si applica solo alle migrazioni dei CDC tipi.
- `Stopping`: Aurora sta interrompendo le migrazioni dei dati. Questo stato si applica quando si sceglie di interrompere la migrazione dei dati dal menu Azioni.
- `Stopped`: Aurora ha interrotto la migrazione dei dati.
- `Failed`: La migrazione dei dati non è riuscita. Per ulteriori informazioni, consulta i file di log.
- `Restarting`: La migrazione dei dati ha riavviato una replica continua dei dati da un CDC punto di partenza.

Tutorial: creazione di un cluster con un personalizzato e un nuovo gruppo di opzioni

In questo tutorial, crei un cluster DB di istanze con un un nuovo gruppo di opzioni. Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

> **Important**
>
> La creazione di un AWS account è gratuita. Tuttavia, completando questo tutorial, potresti incorrere in costi per le AWS risorse che utilizzi. È possibile eliminare queste risorse dopo aver completato l'esercitazione se non sono più necessarie.

Per creare un cluster DB con configurazioni e impostazioni personalizzate, puoi utilizzare parametri personalizzati gruppi di opzioni. I parametri personalizzati sono particolarmente utili se lavori con più database e desideri configurare le relative impostazioni in modo uniforme.

Completando questi passaggi, imparerai:

-
- Come utilizzare parametri specifici per i cluster DB delle istanze .

Per completare questo tutorial, esegui queste attività:

1.
2. Crea un cluster DB di istanze con il gruppo di parametri del cluster DB personalizzato che hai creato.

Argomenti

- Prerequisiti
- Crea un gruppo di parametri del Amazon Aurora DB
- Modificate il valore parametri nel gruppo di parametri personalizzato
- Crea un cluster con un parametro cluster DB nuovo gruppo di opzioni

Prerequisiti

Questo tutorial richiede che tu abbia un utente Account AWS e un utente con accesso amministrativo. Se non li hai già configurati, completa i passaggi nelle seguenti sezioni:

- Registrati per un Account AWS
- Crea un utente con accesso amministrativo

Crea un gruppo di parametri del Amazon Aurora DB

`default_password_lifetime`Il parametro controlla il numero di giorni prima della scadenza automatica della password del client. Per ulteriori informazioni su altri parametri disponibili per i cluster DB delle istanze , vedere.

Per creare un gruppo di parametri

1. Apri la console Amazon RDS e scegli Gruppi di parametri.
2. Per i gruppi di parametri personalizzati, scegli Crea gruppo di parametri.
3. Imposta i dettagli del gruppo di parametri.
    1. Immettete un nome per il gruppo di parametri.
    2. Inserite una descrizione del gruppo di parametri.
    3. Per Tipo di motore, scegli . Aurora MySQL.
    4. Per la famiglia di gruppi di parametri, scegli . aurora-mysql8.0.
    5. Per Tipo, scegliete il gruppo di parametri del cluster DB.
4. Scegli Create (Crea) .

Il nuovo gruppo di parametri del cluster DB viene visualizzato nella pagina Gruppi di parametri nella console Amazon RDS. I passaggi seguenti illustrano come modificare i valori dei parametri per personalizzare il gruppo di parametri.

Modificate il valore parametri nel gruppo di parametri personalizzato

Utilizzate i seguenti passaggi per modificare i valori dei valori parametri nel gruppo di parametri in cui avete creatoCrea un gruppo di parametri del Amazon Aurora DB.

Per modificare i valori dei parametri nel gruppo di parametri

1. Apri la console Amazon RDS e scegli Gruppi di parametri.
2. Per i gruppi di parametri personalizzati, scegli il nome del gruppo di parametri del cluster DB che hai creato.
3. Scegli Modifica.
4. Nella casella di ricerca Filter parameters, cerca il parametro personalizzatodefault_password_lifetime.
5. Seleziona la casella di controllo accanto al parametro e inserisci un valore che indica il numero di giorni da impostare per questo parametro di durata della password.
6. Seleziona Save changes (Salva modifiche).

Cluster DB.

Crea un cluster con un parametro cluster DB nuovo gruppo di opzioni

Infine, crea un cluster con il gruppo di opzioni che hai creato nei passaggi precedenti. I passaggi seguenti mostrano come creare il cluster DB dell'istanza con il il nuovo gruppo di opzioni.

Per creare un cluster DB con un parametro personalizzato e un nuovo gruppo di opzioni

1. Apri la console Amazon RDS e scegli Databases.
2. Scegliere Crea database.
3. Per Scegli un metodo di creazione del database, scegli Standard Create.
4. Per le opzioni Engine, scegli Aurora (MySQL Compatible).
5. Seleziona Configurazione aggiuntiva.
   - Per il nome iniziale del database, scegli un nome per il cluster di database dell'.
   - Nel menu a discesa del gruppo di parametri del cluster DB, seleziona il nome del gruppo di parametri del cluster DB che hai creato in precedenza.
6. Per questo tutorial, puoi lasciare le impostazioni predefinite per qualsiasi altra impostazione del DB o modificarle secondo necessità.
7. Scegliere Crea database.

RDS crea un nuovo cluster DB di istanze con un gruppo di parametri personalizzato e un nuovo di opzioni. Per ulteriori informazioni su questo database, consulta la pagina Database della console Amazon RDS.

In questo tutorial, hai configurato un cluster DB di istanze con impostazioni personalizzate utilizzando un nuovo gruppo di opzioni. Questo cluster DB di istanze di nuova creazione gestisce la durata della password dell'utente utilizzando il parametro. `default_password_lifetime` Per ottimizzare il database, è possibile applicare impostazioni aggiuntive nel gruppo di parametri personalizzato e aggiungere opzioni.

Dopo aver completato la creazione del cluster DB, è necessario eliminare le risorse per evitare costi indesiderati. Eliminazione di cluster e istanze database di Aurora.

Gestione di un cluster DB Amazon Aurora

In questa sezione viene descritto come gestire e mantenere il cluster di database Aurora. Aurora opera su cluster di server di database che sono collegati in una topologia di replica. Pertanto, la gestione di Aurora spesso implica l'implementazione di modifiche su più server e la verifica che tutte le repliche Aurora stiano al passo con il server di origine. Poiché Aurora dimensiona in modo trasparente lo storage sottostante in base alla crescita dei dati, la gestione di Aurora richiede relativamente poca gestione dello storage su disco. Analogamente, dato che Aurora esegue backup continui in automatico, un cluster Aurora non richiede particolari pianificazioni o tempi di inattività per l'esecuzione dei backup.

Argomenti

- Avvio e arresto di un cluster di database Amazon Aurora
- Connessione automatica di un'EC2istanza e a un cluster Aurora DB
- Connessione automatica di una funzione Lambda e di un cluster database Aurora
- Modifica di un cluster database Amazon Aurora
- Aggiunta di repliche di Aurora a un cluster di database
- Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora
- Clonazione di un volume per un cluster di database Amazon Aurora
- Integrazione di Aurora con altri servizi AWS
- Manutenzione di un cluster database Amazon Aurora
- Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB
- Failing su un cluster Amazon Aurora DB
- Eliminazione di cluster e istanze database di Aurora
- Etichettatura di Amazon Aurora e risorse Amazon RDS
- Nomi di risorse Amazon (ARNs) in Amazon RDS
- Aggiornamenti di Amazon Aurora

Avvio e arresto di un cluster di database Amazon Aurora

L'arresto e l'avvio dei cluster Aurora DB ti aiuta a gestire i costi per gli ambienti di sviluppo e test. Puoi arrestare temporaneamente tutte le istanze database nel cluster invece di impostare e rimuovere tutte le istanze database ogni volta che utilizzi il cluster.

Argomenti

- Panoramica dell'avvio e dell'arresto di un cluster di database Aurora
- Limitazioni per l'arresto e l'avvio di cluster di database Aurora
- Arresto di un cluster di database Aurora
- Operazioni possibili durante l'arresto di un cluster di database Aurora
- Avvio di un cluster di database Aurora

Panoramica dell'avvio e dell'arresto di un cluster di database Aurora

Nei periodi in cui non è necessario un cluster Aurora DB, è possibile arrestare tutte le istanze in quel cluster contemporaneamente. Puoi avviare nuovamente il cluster ogni volta che devi utilizzarlo. L'avvio e l'arresto semplificano i processi di impostazione e rimozione dei cluster utilizzati per lo sviluppo, i test o attività simili che non richiedono una disponibilità continua. È possibile eseguire tutte le AWS Management Console procedure necessarie con una sola azione, indipendentemente dal numero di istanze presenti nel cluster.

Quando il cluster DB è fermo, ti vengono addebitati solo i costi dello storage del cluster, delle istantanee manuali e dello storage di backup automatizzato entro la finestra di conservazione specificata. Non è previsto alcun addebito per le ore dell'istanza database.

> **Important**
>
> È possibile arrestare un cluster di database per un massimo di sette giorni. Se non si avvia manualmente il cluster di database dopo sette giorni, esso viene avviato automaticamente in modo che non resti indietro rispetto agli aggiornamenti di manutenzione necessari.

Per ridurre gli addebiti per un cluster Aurora con poco carico, puoi arrestarlo anziché eliminarne tutte le repliche di Aurora. Per i cluster con più di una o due istanze, eliminare e ricreare frequentemente le istanze DB è pratico solo utilizzando l'API o AWS CLI Amazon RDS. Questa sequenza di operazioni

può anche essere difficile da eseguire nell'ordine corretto, ad esempio eliminare tutte le repliche di Aurora prima dell'istanza primaria per evitare l'attivazione del meccanismo di failover.

Non utilizzare l'avvio e l'arresto se il cluster di database deve rimanere in esecuzione ma ha più capacità di quanta ne occorre. Se il cluster è troppo costoso o non molto impiegato, elimina una o più istanze database o modifica tutte le istanze database in una classe di istanze small. Non puoi arrestare una sola istanza database di Aurora.

Il tempo necessario per arrestare il cluster DB varia a seconda di fattori quali le classi di istanze DB, lo stato della rete, il tipo di motore DB e lo stato del database. Il processo può richiedere diversi minuti. Il servizio Amazon RDS esegue le seguenti azioni:

- Arresta i processi del motore di database.
- Arresta i processi della piattaforma RDS.
- Termina le EC2 istanze Amazon sottostanti.

Il tempo necessario per riavviare il cluster DB varia a seconda di fattori quali le dimensioni del database, le classi di istanze DB, lo stato della rete, il tipo di motore DB e lo stato del database al momento della chiusura del cluster. Il processo di avvio può richiedere da minuti a ore, ma in genere richiede diversi minuti. Ti consigliamo di considerare la variabilità del tempo di avvio durante la creazione del piano di disponibilità.

Per avviare il cluster DB, il servizio esegue azioni come le seguenti:

- Effettua il provisioning delle EC2 istanze Amazon sottostanti.
- Avvia i processi della piattaforma RDS.
- Avvia i processi del motore di database.
- Recupera le istanze DB (il ripristino avviene anche dopo un normale arresto).

Limitazioni per l'arresto e l'avvio di cluster di database Aurora

Non è possibile arrestare e avviare alcuni cluster Aurora:

- È possibile arrestare e avviare un cluster che fa parte di un database globale Aurora solo se è l'unico cluster nel database globale.
- Non è possibile arrestare e avviare un cluster con una replica di lettura tra Regioni.

- Non è possibile interrompere e avviare un cluster che fa parte di una distribuzione blu/verde.
- Non è possibile interrompere e avviare un Aurora Serverless v1 grappolo. Con Aurora Serverless v2, è possibile arrestare e avviare il cluster.

Arresto di un cluster di database Aurora

Per utilizzare o amministrare un cluster di database di Aurora, si inizia sempre da un cluster di database Aurora in esecuzione, poi si arresta il cluster e lo si avvia di nuovo. Per tutta la durata dell'arresto di un cluster, vengono addebitati i costi per lo storage del cluster, gli snapshot manuali e lo storage di backup automatici all'interno della finestra di retention specificata, ma non per le ore di utilizzo dell'istanza database.

L'operazione di arresto interrompe prima le istanze di replica di Aurora e poi l'istanza primaria, per evitare l'attivazione del meccanismo di failover.

Console

Per arrestare un cluster Aurora

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database), quindi scegliere un cluster. Si può eseguire l'operazione di arresto da questa pagina o andare alla pagina dei dettagli del cluster di database da arrestare.
3. Per Actions (Operazioni), scegli Stop temporarily (Arresta temporaneamente).
4. Nella finestra Stop DB cluster temporarily, (Arresta temporaneamente il cluster di database) seleziona la conferma per il riavvio automatico del cluster database dopo 7 giorni.
5. Scegli Stop temporarily (Arresta temporaneamente) per arrestare il cluster database oppure Cancel (Annulla) per annullare l'operazione.

AWS CLI

Per interrompere un'istanza DB utilizzando il AWS CLI, chiama il stop-db-clustercomando con i seguenti parametri:

- `--db-cluster-identifier` – il nome del cluster Aurora.

Example

```
aws rds stop-db-cluster --db-cluster-identifier mydbcluster
```

API RDS

Per interrompere un'istanza DB utilizzando l'API Amazon RDS, chiama l'DBClusteroperazione Stop con il seguente parametro:

- `DBClusterIdentifier` – il nome del cluster Aurora.

Operazioni possibili durante l'arresto di un cluster di database Aurora

Mentre un cluster Aurora è fermo, è possibile eseguire un point-in-time ripristino in qualsiasi punto all'interno della finestra di conservazione dei backup automatizzata specificata. Per informazioni dettagliate sull'esecuzione di un point-in-time ripristino, consultaRipristino dei dati.

Non è possibile modificare la configurazione di un cluster di database Aurora o delle istanze database durante l'arresto del cluster. Inoltre, non è possibile aggiungere o rimuovere istanze database dal cluster o eliminarlo se ha ancora istanze database associate. Devi avviare il cluster prima di eseguire questo tipo di operazioni amministrative.

L'arresto di un cluster di database rimuove le azioni in sospeso, ad eccezione del gruppo di parametri del cluster di database o dei gruppi di parametri del database delle istanze del cluster di database.

Aurora applica la manutenzione pianificata al cluster in arresto dopo il suo riavvio. Ricorda che dopo sette giorni Aurora avvia automaticamente i cluster in arresto in modo che non rimangano troppo indietro rispetto allo stato di manutenzione.

Inoltre, Aurora non esegue backup automatici, perché i dati sottostanti non possono cambiare nel periodo di arresto del cluster. Aurora non estende il tempo di conservazione del backup per il cluster database mentre è arrestato.

Avvio di un cluster di database Aurora

Puoi sempre avviare un cluster di database Aurora partendo da un cluster Aurora che è già nello stato di arresto. Quando avvii il cluster, tutte le sue istanze database tornano disponibili. Il cluster mantiene le sue impostazioni di configurazione, come gli endpoint, i gruppi di parametri e i gruppi di sicurezza VPC.

L'avvio del cluster DB richiede in genere alcuni minuti.

Console

Per avviare un cluster Aurora

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database), quindi scegliere un cluster. Si può eseguire l'operazione di avvio da questa pagina o andare alla pagina dei dettagli del cluster di database da avviare.
3. In Actions (Operazioni), scegliere Start (Avvia).

AWS CLI

Per avviare un cluster DB utilizzando il AWS CLI, chiama il start-db-clustercomando con i seguenti parametri:

- `--db-cluster-identifier` – il nome del cluster Aurora. Questo nome è sia lo specifico identificatore del cluster scelto durante la sua creazione sia l'identificatore istanze database scelto con `-cluster` aggiunto alla fine.

Example

```
aws rds start-db-cluster --db-cluster-identifier mydbcluster
```

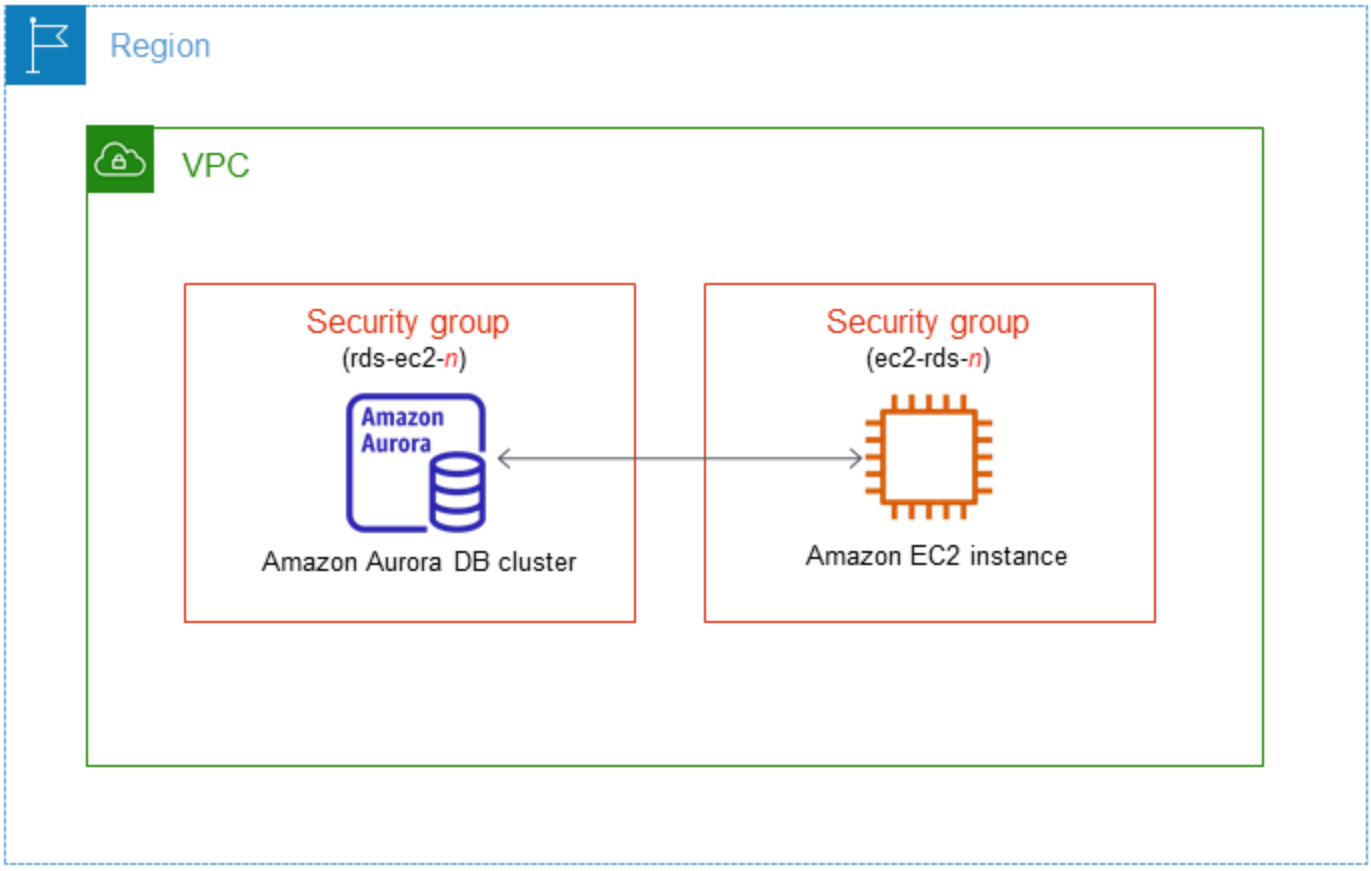
API RDS

Per avviare un cluster Aurora DB utilizzando l'API Amazon RDS, chiama l'DBClusteroperazione Start con il seguente parametro:

- `DBCluster` – il nome del cluster Aurora. Questo nome è sia lo specifico identificatore del cluster scelto durante la sua creazione sia l'identificatore istanze database scelto con `-cluster` aggiunto alla fine.

Connessione automatica di un'EC2istanza e a un cluster Aurora DB

Puoi utilizzare la RDS console Amazon per semplificare la configurazione di una connessione tra un'istanza Amazon Elastic Compute Cloud (AmazonEC2) e un', un cluster Aurora DB. Spesso, il cluster DB si trova in una sottorete privata e l'EC2istanza si trova in una sottorete pubblica all'interno di un. VPC Puoi utilizzare un SQL client sulla tua EC2 istanza per connetterti al cluster DB dell'. L'EC2istanza può anche eseguire server Web o applicazioni che accedono al cluster DB private.



Se desideri connetterti a un'EC2istanza che non si trova nello VPC stesso cluster Aurora DB, consulta gli scenari in. Scenari per l'accesso a un cluster di DB in un VPC

Argomenti

- Panoramica della connettività automatica con un'istanza EC2
- Connessione automatica di un'EC2istanza e di un cluster Aurora DB
- Visualizzazione delle risorse di calcolo connesse
- Connessione a un'istanza database che esegue un motore DB specifico

Panoramica della connettività automatica con un'istanza EC2

Quando configuri una connessione tra un'EC2istanza e un cluster Aurora DB di , Amazon RDSautomatically configura il gruppo di VPC sicurezza per la tua EC2 istanza e per il tuo cluster DB di .

Di seguito sono riportati i requisiti per connettere un'EC2istanza a un Aurora DB:

- L'EC2istanza deve esistere nello VPC stesso cluster DB del .

  Se non esistono EC2 istanze nella stessa istanzaVPC, la console fornisce un collegamento per crearne una.
- Attualmente, il cluster database non può essere un cluster database Aurora Serverless o parte di un database globale Aurora.
- L'utente che configura la connettività deve disporre delle autorizzazioni per eseguire le seguenti EC2 operazioni Amazon:
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Se l'istanza DB e l'EC2istanza si trovano in zone di disponibilità diverse, il tuo account potrebbe sostenere costi tra zone di disponibilità diverse.

Quando configuri una connessione a un'EC2istanza, Amazon RDS agisce in base alla configurazione corrente dei gruppi di sicurezza associati al cluster e all'EC2istanza del DB, come descritto nella tabella seguente.

Configurazione attuale del gruppo di RDS sicurezza	Configurazione attuale del gruppo di EC2 sicurezza	RDSazione
Esistono uno o più gruppi di sicurezza associati al cluster DB del con un nome che corrisponde al modello `rds-ec2-`*n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in entrata con il gruppo di VPC sicurezza dell'EC2istanza come origine.	All'EC2istanza sono associati uno o più gruppi di sicurezza con un nome che corrisponde al modello `ec2-rds-`*n* (dove *n* è un numero). Un gruppo di sicurezza che corrispon de al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in uscita con il gruppo di VPC sicurezza del cluster DB del come origine.	RDSnon esegue alcuna azione. Una connessione era già configurata automaticamente tra l'EC2istanza e il cluster DB del RDS . Poiché esiste già una connessione tra l'EC2ista nza e il RDS database, i gruppi di sicurezza non vengono modificati.
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster DB del con un nome che corrisponda al modello`rds-ec2-`*n*. • Esistono uno o più gruppi di sicurezza associati al cluster DB del con un nome che corrisponde al modello`rds-ec2-`*n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con l'EC2istanza. Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato all'EC2istanza con un nome che corrisponda al modello`ec2-rds-`*n*. • All'EC2istanza sono associati uno o più gruppi di sicurezza con un nome che corrisponde al modello`ec2-rds-`*n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessio ne con il cluster DB del . Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in uscita con il	RDS action: create new security groups

Configurazione attuale del gruppo di RDS sicurezza	Configurazione attuale del gruppo di EC2 sicurezza	RDSazione
entrata con il gruppo di VPC sicurezza dell'EC2istanza come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato. Esempi di modifiche sono l'aggiunta di una regola o la modifica della porta di una regola esistente.	gruppo di VPC sicurezza del cluster DB del come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	
Esistono uno o più gruppi di sicurezza associati al cluster DB del con un nome che corrisponde al modello`rds-ec2-`*n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in entrata con il gruppo di VPC sicurezza dell'EC2istanza come origine.	All'EC2istanza sono associati uno o più gruppi di sicurezza con un nome che corrispondе al modello`ec2-rds-`*n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con il cluster DB del . Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in uscita con il gruppo di VPC sicurezza del cluster DB del come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	RDS action: create new security groups

Configurazione attuale del gruppo di RDS sicurezza	Configurazione attuale del gruppo di EC2 sicurezza	RDSazione
Esistono uno o più gruppi di sicurezza associati al cluster DB del con un nome che corrisponde al modello`rds-ec2-`*n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in entrata con il gruppo di VPC sicurezza dell'EC2istanza come origine.	Esiste un gruppo EC2 di sicurezza valido per la connessione, ma non è associato all'EC2istanza. Questo gruppo di sicurezza ha un nome che corrisponde al modello `ec2-rds-`*n*. Non è stato modificato. Ha solo una regola in uscita con il gruppo di VPC sicurezza del cluster DB del come origine.	RDS action: associate EC2 security group

Configurazione attuale del gruppo di RDS sicurezza	Configurazione attuale del gruppo di EC2 sicurezza	RDSazione
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster DB del con un nome che corrisponda al modello`rds-ec2-`*n*. • Esistono uno o più gruppi di sicurezza associati al cluster DB del con un nome che corrisponde al modello`rds-ec2-`*n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con l'EC2istanza. Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in entrata con il gruppo di VPC sicurezza dell'EC2istanza come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	Esistono uno o più gruppi di sicurezza associati all'EC2istanza con un nome che corrisponde allo schema`ec2-rds-`*n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in uscita con il gruppo di VPC sicurezza del cluster DB del come origine.	RDS action: create new security groups

RDSazione: creare nuovi gruppi di sicurezza

Amazon RDS intraprende le seguenti azioni:

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `rds-ec2-`*n*. Questo gruppo di sicurezza ha una regola in entrata con il gruppo di VPC sicurezza dell'EC2istanza come origine.

Questo gruppo di sicurezza è associato al cluster DB del e consente all'EC2istanza di accedere al cluster DB del .

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `ec2-rds-n`. Questo gruppo di sicurezza ha una regola in uscita con il gruppo di VPC sicurezza del cluster DB del come destinazione. Questo gruppo di sicurezza è associato all'EC2istanza e consente all'EC2istanza di inviare traffico al cluster DB del .

RDSazione: associare un gruppo EC2 di sicurezza

Amazon RDS associa il gruppo di EC2 sicurezza valido ed esistente all'EC2istanza. Questo gruppo di sicurezza consente all'EC2istanza di inviare traffico al cluster DB del .

Connessione automatica di un'EC2istanza e di un cluster Aurora DB

Prima di configurare una connessione tra un'EC2istanza e un cluster di Aurora DB, assicurati di soddisfare i requisiti descritti in. Panoramica della connettività automatica con un'istanza EC2

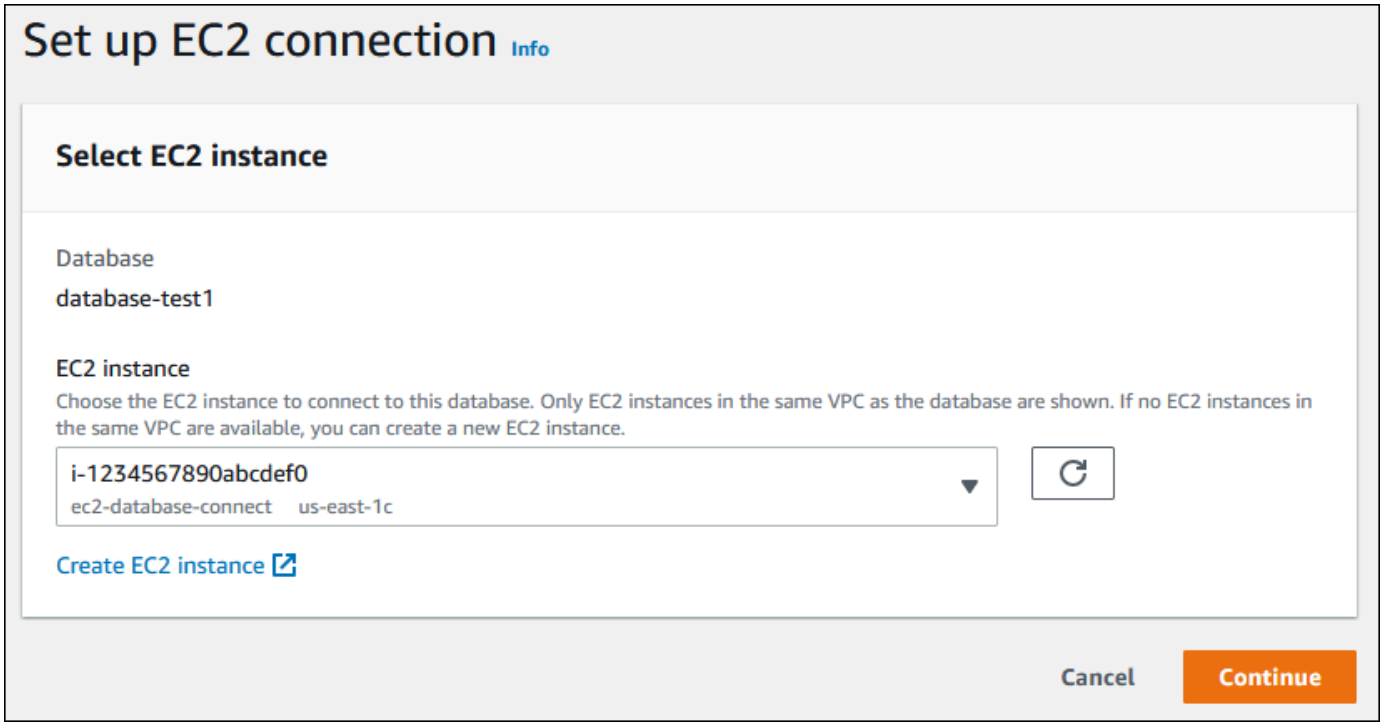
Se si apportano modifiche ai gruppi di sicurezza dopo aver configurato la connettività, le modifiche potrebbero influire sulla connessione tra l'EC2istanza e il cluster di Aurora DB.

> **Note**
>
> È possibile configurare automaticamente una connessione tra un'EC2istanza e un del cluster Aurora DB utilizzando il. AWS Management Console Non è possibile configurare automaticamente una connessione con AWS CLI o RDSAPI.

Per connettere automaticamente un'EC2istanza e un , il cluster Aurora DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi scegli il cluster DB del .
3. Da Azioni, scegli EC2Configura connessione.

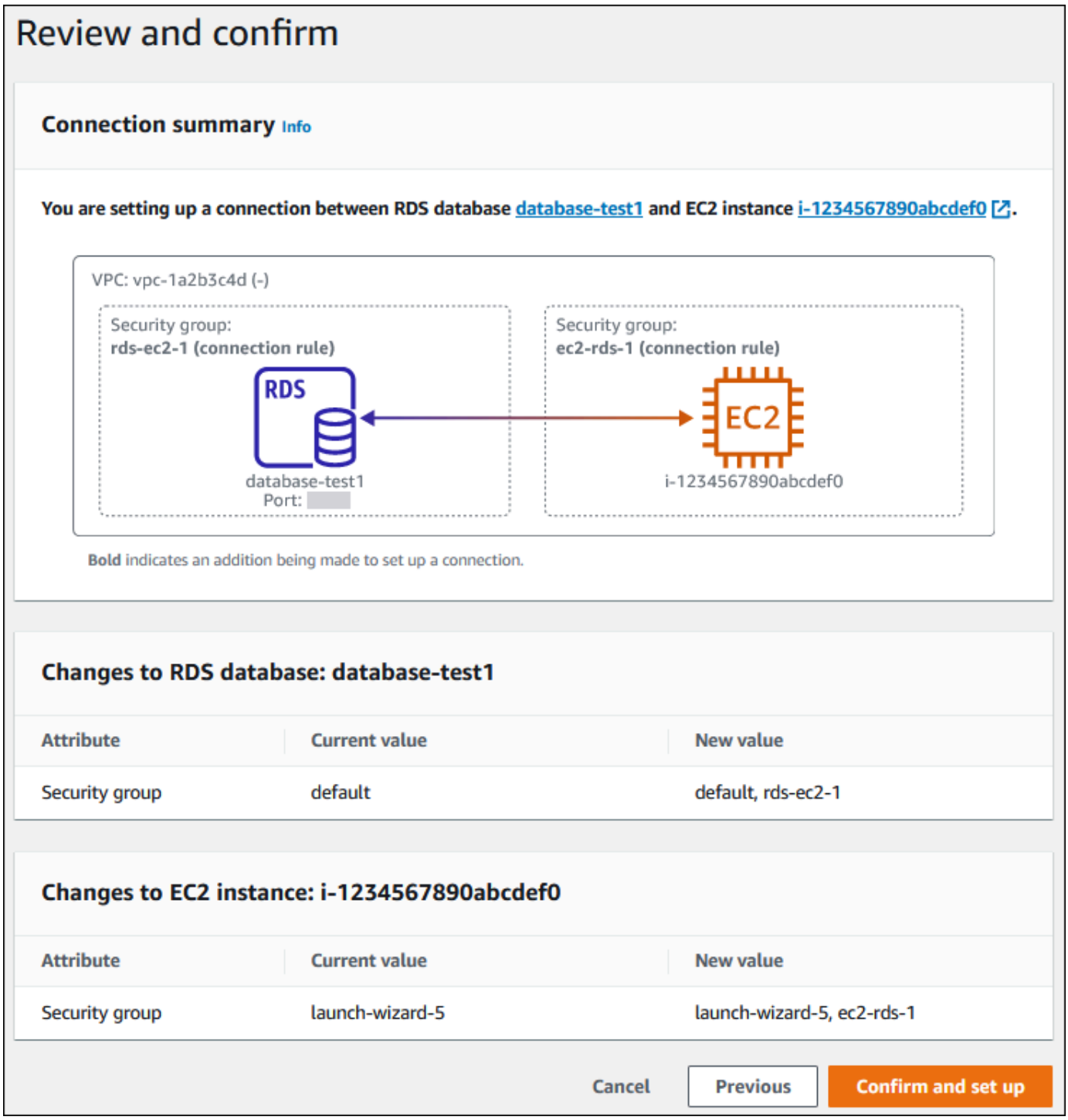
   Viene visualizzata la pagina EC2 Configura connessione.
4. Nella pagina EC2Configura connessione, scegli l'EC2istanza.



Se non esistono EC2 istanze nella stessa istanzaVPC, scegli Crea EC2 istanza per crearne una. In questo caso, assicurati che la nuova EC2 istanza sia nella VPC stessa del cluster DB del .

5. Scegli Continua.

   Viene visualizzata la pagina Review and confirm (Rivedi e conferma).



6. Nella pagina Rivedi e conferma, esamina le modifiche che RDS verranno apportate alla configurazione della connettività con l'EC2istanza.

   Se le modifiche sono corrette, scegli Conferma e configura.

   Se le modifiche non sono corrette, scegli Previous (Precedente) o Cancel (Annulla).

Visualizzazione delle risorse di calcolo connesse

È possibile utilizzare il AWS Management Console per visualizzare le risorse di calcolo connesse a un cluster di Aurora DB. Le risorse mostrate includono le connessioni delle risorse di calcolo configurate automaticamente. È possibile configurare la connettività delle risorse di calcolo automaticamente nei modi seguenti:

- È possibile selezionare la risorsa di calcolo quando si crea il database.

  Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.
- È possibile configurare la connettività tra un database esistente e una risorsa di calcolo.

  Per ulteriori informazioni, consulta Connessione automatica di un'EC2istanza e di un cluster Aurora DB.

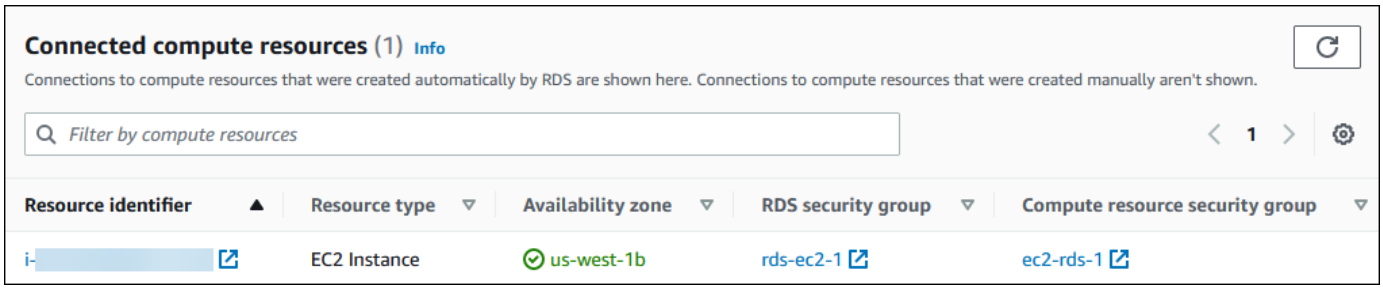
Le risorse di calcolo elencate non includono quelle connesse al database manualmente. Ad esempio, è possibile consentire a una risorsa di calcolo di accedere manualmente a un database aggiungendo una regola al gruppo di VPC sicurezza associato al database.

Per garantire la presenza della risorsa di calcolo nell'elenco, è necessario che siano soddisfatte le condizioni elencate di seguito.

- Il nome del gruppo di sicurezza associato alla risorsa di calcolo corrisponde al modello `ec2-rds-`*n* (dove *n* è un numero).
- Il gruppo di sicurezza associato alla risorsa di calcolo ha una regola in uscita con l'intervallo di porte impostato sulla porta utilizzata dal cluster DB del .
-
- Il nome del gruppo di sicurezza associato al cluster DB del corrisponde al modello `rds-ec2-`*n* (dove *n* è un numero).
- Il gruppo di sicurezza associato al cluster DB del ha una regola in entrata con l'intervallo di porte impostato sulla porta utilizzata dal cluster DB del .
- Il gruppo di sicurezza associato al cluster DB del ha una regola in entrata con l'origine impostata su un gruppo di sicurezza associato alla risorsa di calcolo.

Per visualizzare le risorse di calcolo connesse a un Aurora DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi scegli il nome del cluster DB del .
3. Nella scheda Connectivity & security (Connettività e sicurezza), visualizza le risorse di calcolo in Connected compute resources (Risorse di calcolo connesse).



Connessione a un'istanza database che esegue un motore DB specifico

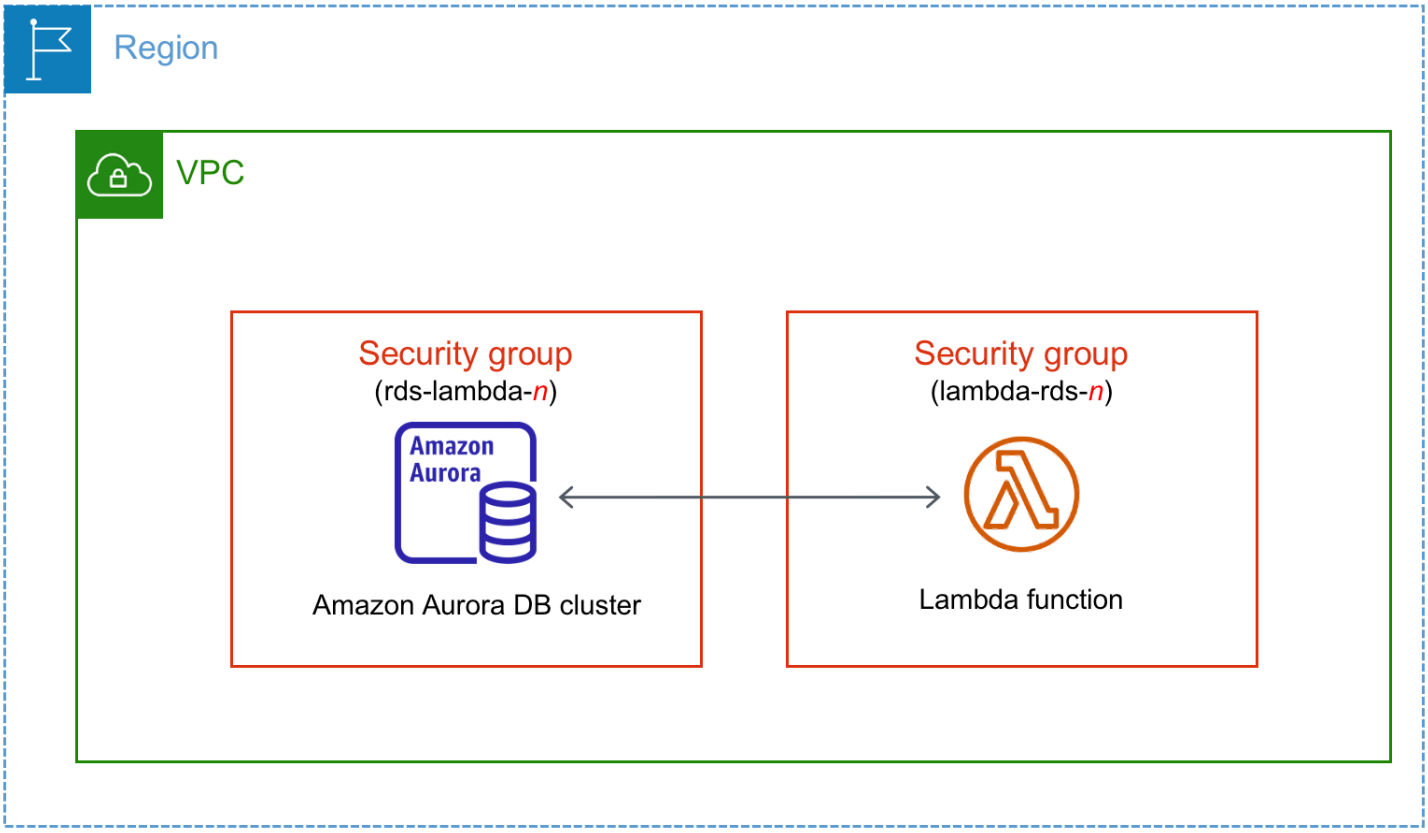
Per informazioni sulla connessione a un'istanza database che esegue un motore DB specifico, seguire le istruzioni per il motore DB:

- Connessione a un cluster di database Amazon Aurora MySQL
- Connessione a un cluster di database Amazon Aurora PostgreSQL

Connessione automatica di una funzione Lambda e di un cluster database Aurora

Puoi utilizzare la RDS console Amazon per semplificare la configurazione di una connessione tra una funzione Lambda e un' Aurora DB. Spesso, il cluster DB si trova in una sottorete privata all'interno di un. VPC La funzione Lambda può essere utilizzata dalle applicazioni per accedere al cluster database privato.

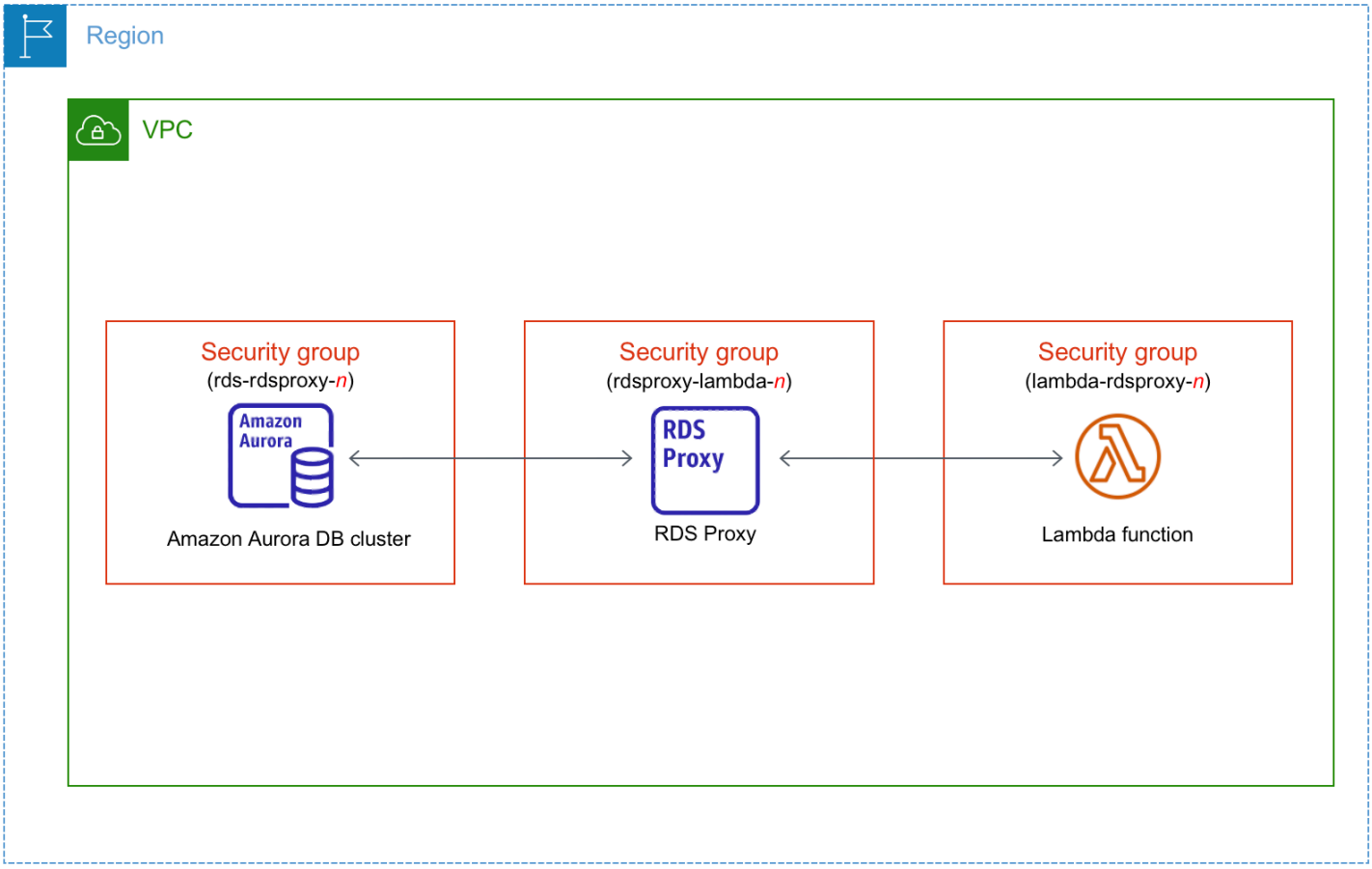
L'immagine seguente mostra una connessione diretta tra il cluster database e la funzione Lambda.



È possibile configurare la connessione tra la funzione Lambda e il cluster DB dell' tramite RDS Proxy per migliorare le prestazioni e la resilienza del database. Spesso, le funzioni Lambda effettuano connessioni al database brevi e frequenti che traggono vantaggio dal pool di connessioni offerto da Proxy. RDS Puoi sfruttare qualsiasi autenticazione AWS Identity and Access Management (IAM) di cui disponi già per le funzioni Lambda, invece di gestire le credenziali del database nel codice dell'applicazione Lambda. Per ulteriori informazioni, consulta Proxy Amazon RDS per Aurora.

Quando usi la console per connetterti a un proxy esistente, Amazon RDS aggiorna il gruppo di sicurezza del proxy per consentire le connessioni dal cluster DB dell' e dalla funzione Lambda dell'istanza DB.

È anche possibile creare un nuovo proxy dalla stessa pagina della console. Quando crei un proxy nella console, per accedere al cluster DB, devi inserire le credenziali del database o selezionare un AWS Secrets Manager segreto.



Tip

Per connettere rapidamente una funzione Lambda a cluster Aurora DB, puoi anche utilizzare la procedura guidata integrata nella console. Per aprire la procedura guidata, procedi come segue:

1. Aprire la pagina Funzioni della console Lambda.
2. Seleziona la funzione a cui vuoi connettere un database.
3. Nella scheda Configurazione, seleziona RDSdatabase.
4. Scegli Connect to RDS database.

Dopo aver collegato la funzione a un database, puoi creare un proxy scegliendo Aggiungi proxy.

Argomenti

- Panoramica della connettività automatica a una funzione Lambda
- Connessione automatica di una funzione Lambda e un cluster di database
- Visualizzazione delle risorse di calcolo connesse

Panoramica della connettività automatica a una funzione Lambda

Di seguito sono riportati i requisiti per connettere una funzione Lambda con un cluster di istanze DB:

- La funzione Lambda deve esistere nello VPC stesso cluster DB.
- Attualmente, il cluster DB non può essere un Aurora Serverless Cluster DB o parte di un database globale Aurora.
- L'utente che configura la connettività deve disporre delle autorizzazioni per eseguire le seguenti operazioni AmazonRDS, AmazonEC2, Lambda, Secrets Manager e: IAM
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBClusters`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBCluster`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`
    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`

  - `ec2:RevokeSecurityGroupEgress`
  - `ec2:RevokeSecurityGroupIngress`
- Lambda
  - `lambda:CreateFunctions`
  - `lambda:ListFunctions`
  - `lambda:UpdateFunctionConfiguration`
- Secrets Manager
  - `secretsmanager:CreateSecret`
  - `secretsmanager:DescribeSecret`
- IAM
  - `iam:AttachPolicy`
  - `iam:CreateRole`
  - `iam:CreatePolicy`
- AWS KMS
  - `kms:describeKey`

> **Note**
>
> Se il cluster database e la funzione Lambda si trovano in zone di disponibilità diverse, potrebbero essere addebitati costi tra zone di disponibilità all'account.

Se utilizzi RDS Proxy, Amazon configura RDS anche il gruppo VPC di sicurezza per il proxy. Amazon RDS agisce in base alla configurazione corrente dei gruppi di sicurezza associati al cluster DB, alla funzione Lambda e al proxy, come descritto nella tabella seguente.

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Esistono uno o più gruppi di sicurezza associati al cluster	Esistono uno o più gruppi di sicurezza associati alla funzione	Esistono uno o più gruppi di sicurezza associati al proxy	Amazon non RDS intraprende alcuna azione.

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
DB con un nome che corrisponde allo schema `rds-lambda-` *n* o, se un proxy è già connesso al cluster DB dell' `TargetHealth` dell'istanza DB, RDS verifica se il proxy associato lo èAVAILABLE . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine.	Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in uscita con il gruppo di VPC sicurezza del cluster di database o il proxy come destinazione.	con un nome che corrisponde al modello `rdsproxy-lambda-` *n* (dove *n* è un numero). Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha regole in entrata e in uscita con i gruppi di VPC sicurezza della funzione Lambda e del cluster DB di istanze .	Una connessione era già stata configurata automaticamente tra la funzione Lambda, il proxy (opzionale) e il cluster database. Poiché esiste già una connessione tra la funzione, il proxy e il database, i gruppi di sicurezza non vengono modificati.

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . • Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Tuttavia, nessuno di questi gruppi di sicurezza può essere utilizzato per la connessione alla funzione Lambda.	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. • Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con il cluster DB. Amazon non RDS può utilizzare un	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. • Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con il cluster DB o la funzione Lambda. Amazon non RDS può utilizzare un gruppo di sicurezza che non	RDS action: create new security groups

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato. Esempi di modifiche sono l'aggiunta di una regola o la modifica della porta di una regola esistente.	gruppo di sicurezza che non ha una regola in uscita con il gruppo di VPC sicurezza del cluster o proxy DB dell'istanza DB come destinazione. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	ha regole in entrata e in uscita con il gruppo di VPC sicurezza del cluster DB di e la funzione Lambda. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificat o.	

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine.	Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rd s-` *n* o `lambda-rd sproxy-` *n*. Tuttavia, Amazon non RDS può utilizzar e nessuno di questi gruppi di sicurezza per la connessione con il cluster DB. Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in uscita con il gruppo di VPC sicurezza del cluster o proxy DB dell'istanza DB come destinazione. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrisponde al modello `rdsproxy-lambda-` *n*. Tuttavia, Amazon non RDS può utilizzar e nessuno di questi gruppi di sicurezza per la connessione con il cluster DB o la funzione Lambda. Amazon non RDS può utilizzare un gruppo di sicurezza che non ha regole in entrata e in uscita con il gruppo di VPC sicurezza del cluster DB di e la funzione Lambda. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificat o.	RDS action: create new security groups

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine.	Esiste un gruppo di sicurezza Lambda valido per la connessione, ma non è associato alla funzione Lambda. Questo gruppo di sicurezza ha un nome che corrisponde al modello `lambda-rd s-` *n* o `lambda-rd sproxy-` *n*. Non è stato modificato. Ha una sola regola in uscita con il gruppo di VPC sicurezza del cluster o proxy DB come destinazione.	Esiste un gruppo di sicurezza proxy valido per la connessione, ma non è associato al proxy. Questo gruppo di sicurezza ha un nome che corrisponde al modello `rdsproxy-lambda-` *n*. Non è stato modificato. Dispone di regole in entrata e in uscita con il gruppo di VPC sicurezza del cluster e la funzione Lambda.	RDS action: associate Lambda security group

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster database con un nome che corrisponde al modello `rds-lambda-` *n* o se `TargetHealth` di un proxy associato è `AVAILABLE` . • Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrisponde al modello `rds-lambda-` *n* o se `TargetHealth` di un proxy associato è `AVAILABLE` . Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con la	Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha una sola regola in uscita con il gruppo di VPC sicurezza dell' o del proxy dell'istanza DB come destinazione.	Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrisponde al modello `rdsproxy-lambda-` *n*. Un gruppo di sicurezza che corrisponde al modello non è stato modificato. Questo gruppo di sicurezza ha regole in entrata e in uscita con il gruppo di VPC sicurezza del cluster e la funzione Lambda.	RDS action: create new security groups

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
funzione o il proxy Lambda. Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificat o.			

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
Si applica una delle due condizioni seguenti: • Non esiste alcun gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . • Esiste almeno un gruppo di sicurezza associato al cluster database con un nome che corrispon de al modello `rds-lambda-` *n* o se `TargetHea lth` di un proxy associato è `AVAILABLE` . Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con la	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. • Esistono uno o più gruppi di sicurezza associati alla funzione Lambda con un nome che corrisponde al modello `lambda-rds-` *n* o `lambda-rdsproxy-` *n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con il cluster DB. Amazon non RDS può utilizzare un gruppo di	Si applica una delle due condizioni seguenti: • Non esiste un gruppo di sicurezza associato al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. • Esistono uno o più gruppi di sicurezza associati al proxy con un nome che corrispon de al modello `rdsproxy-lambda-` *n*. Tuttavia, Amazon non RDS può utilizzare nessuno di questi gruppi di sicurezza per la connessione con il cluster DB o la funzione Lambda. Amazon non RDS può utilizzare un gruppo di sicurezza che non	RDS action: create new security groups

Configurazione attuale del gruppo RDS di sicurezza	Configurazione del gruppo di sicurezza Lambda corrente	Configurazione del gruppo di sicurezza proxy corrente	Operazione RDS
funzione o il proxy Lambda. Amazon non RDS può utilizzare un gruppo di sicurezza che non ha una regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificat o.	sicurezza che non ha una regola in uscita con il gruppo di VPC sicurezza del cluster o proxy DB dell'ista nza di database come origine. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificato.	ha regole in entrata e in uscita con il gruppo di VPC sicurezza del cluster DB di e la funzione Lambda. Amazon RDS inoltre non può utilizzare un gruppo di sicurezza che è stato modificat o.	

RDSazione: creare nuovi gruppi di sicurezza

Amazon RDS intraprende le seguenti azioni:

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `rds-lambda-`*n* o `rds-rdsproxy-`*n* (se scegli di utilizzare RDS Proxy). Questo gruppo di sicurezza ha una regola in entrata con il gruppo di VPC sicurezza della funzione o del proxy Lambda come origine. Questo gruppo di sicurezza è associato al cluster database e consente alla funzione o al proxy di accedere al cluster database.
- Crea un nuovo gruppo di sicurezza che corrisponde al modello `lambda-rds-`*n* o `lambda-rdsproxy-`*n*. Questo gruppo di sicurezza ha una regola in uscita con il gruppo di VPC sicurezza del cluster o proxy DB come destinazione. Questo gruppo di sicurezza è associato alla funzione Lambda e consente alla funzione di inviare traffico al cluster database o inviare traffico tramite un proxy.

- Crea un nuovo gruppo di sicurezza che corrisponde al modello `rdsproxy-lambda-`*n*. Questo gruppo di sicurezza ha regole in entrata e in uscita con il gruppo di VPC sicurezza del cluster e la funzione Lambda.

RDSazione: associare il gruppo di sicurezza Lambda

Amazon RDS associa il gruppo di sicurezza Lambda valido ed esistente alla funzione Lambda. Questo gruppo di sicurezza consente alla funzione di inviare traffico al cluster database o inviare traffico tramite un proxy.

Connessione automatica di una funzione Lambda e un cluster di database

Puoi utilizzare la RDS console Amazon per connettere automaticamente una funzione Lambda al cluster DB dell'. Ciò semplifica il processo di configurazione di una connessione tra queste risorse.

Puoi anche usare RDS Proxy per includere un proxy nella tua connessione. Le funzioni Lambda effettuano connessioni brevi e frequenti al database che traggono vantaggio dal pool di connessioni offerto da Proxy. RDS Puoi anche utilizzare qualsiasi IAM autenticazione che hai già configurato per la tua funzione Lambda, invece di gestire le credenziali del database nel codice dell'applicazione Lambda.

È possibile connettere un cluster database esistente a funzioni Lambda nuove ed esistenti utilizzando la pagina Configurazione della connessione Lambda. Il processo di configurazione consente di impostare automaticamente i gruppi di sicurezza richiesti.

Prima di configurare una connessione tra una funzione Lambda e un cluster database, assicurati che:

- La funzione Lambda e il cluster DB si trovano nello stesso cluster. VPC
- Disponi delle autorizzazioni corrette per l'account utente. Per ulteriori informazioni sui requisiti, consulta Panoramica della connettività automatica a una funzione Lambda.

Se si modificano i gruppi di sicurezza dopo la configurazione della connettività, le modifiche potrebbero influenzare la connessione tra la funzione Lambda e il cluster database.

> **Note**
>
> È possibile configurare automaticamente una connessione tra un cluster DB e una funzione Lambda solo in. AWS Management Console Per connettere una funzione Lambda, tutte le istanze nel cluster database devono essere nello stato Disponibile.

Per connettere automaticamente una funzione Lambda e un cluster database

<result>

Dopo aver confermato la configurazione, Amazon RDS avvia il processo di connessione della funzione Lambda, del RDS proxy (se hai utilizzato un proxy) e del cluster DB dell'. La console mostra la finestra di dialogo Dettagli di connessione, in cui sono elencate le modifiche al gruppo di sicurezza che consentono le connessioni tra le risorse.
</result>

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi seleziona il cluster database che desideri connettere a una funzione Lambda.
3. Per Operazioni, scegli Configura connessione Lambda.
4. Nella pagina Configurazione della connessione Lambda, in Seleziona la funzione Lambda, effettua una delle seguenti operazioni:
   - Se disponi di una funzione Lambda esistente nello VPC stesso cluster DB, scegli Scegli funzione esistente, quindi scegli la funzione.
   - Se non hai una funzione Lambda nella stessa funzioneVPC, scegli Crea nuova funzione, quindi inserisci il nome di una funzione. Il runtime predefinito è impostato su Nodejs.18. Puoi modificare le impostazioni per la nuova funzione Lambda nella console Lambda dopo aver completato la configurazione della connessione.
5. (Facoltativo) In RDSProxy, seleziona Connetti tramite RDS proxy, quindi esegui una delle seguenti operazioni:
   - Se disponi di un proxy esistente che desideri utilizzare, seleziona Scegli un proxy esistente, quindi seleziona il proxy.
   - Se non disponi di un proxy e desideri che Amazon ne RDS crei uno automaticamente per te, scegli Crea nuovo proxy. Quindi, per Credenziali del database, esegui una delle seguenti operazioni:

a. Seleziona Nome utente e password del database, quindi inserisci Nome utente e Password per il cluster database.

b. Seleziona Segreti Secrets Manager. Quindi, per Seleziona segreto, scegli un AWS Secrets Manager segreto. Se non disponi di un segreto di Secrets Manager, seleziona Crea nuovo segreto di Secrets Manager per creare un nuovo segreto. Dopo aver creato il segreto, per Seleziona segreto, scegli il nuovo segreto.

Dopo aver creato il nuovo proxy, seleziona Scegli un proxy esistente, quindi seleziona il proxy. Tieni presente che prima che il proxy sia disponibile per la connessione, potrebbe trascorrere del tempo.

6. (Facoltativo) Espandi Riepilogo della connessione e verifica gli aggiornamenti evidenziati per le risorse.
7. Scegliere Set up (Configura).

Visualizzazione delle risorse di calcolo connesse

È possibile utilizzare il AWS Management Console per visualizzare le funzioni Lambda connesse al cluster DB dell'. Le risorse mostrate includono connessioni alle risorse di calcolo che Amazon RDS configura automaticamente.

Le risorse di elaborazione elencate non includono quelle connesse manualmente al cluster database. Ad esempio, puoi consentire a una risorsa di calcolo di accedere manualmente al cluster DB aggiungendo una regola al gruppo VPC di sicurezza associato al database.

Affinché la console elenchi una funzione Lambda, devono essere soddisfatte le seguenti condizioni:

- Il nome del gruppo di sicurezza associato alla risorsa di calcolo corrisponde al modello `lambda-rds-`*n* o `lambda-rdsproxy-`*n* (dove *n* è un numero).
- Il gruppo di sicurezza associato alla risorsa di calcolo dispone di una regola in uscita con l'intervallo di porte impostato sulla porta del cluster database o un proxy associato. La destinazione della regola in uscita deve essere impostata su un gruppo di sicurezza associato al cluster database o a un proxy associato.
- Se la configurazione include un proxy, il nome del gruppo di sicurezza collegato al proxy associato al database corrisponde al modello `rdsproxy-lambda-`*n* (dove *n* è un numero).

- Il gruppo di sicurezza associato alla funzione dispone di una regola in uscita con l'intervallo di porte impostato sulla porta utilizzata dal cluster database o da un proxy associato. La destinazione deve essere impostata su un gruppo di sicurezza associato al cluster database o a un proxy associato.

Per visualizzare le risorse di elaborazione connesse automaticamente a un cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Database e seleziona il cluster database.
3. Nella scheda Connettività e sicurezza, visualizza le risorse di calcolo in Risorse di calcolo connesse.

Modifica di un cluster database Amazon Aurora

È possibile modificare le impostazioni di un cluster di database per completare attività come la modifica del periodo di retention dei backup o della porta del database. È anche possibile modificare le istanze database in un cluster di database per completare attività come la modifica della classe dell'istanza database o l'abilitazione di Performance Insights. In questo argomento viene illustrato come modificare un cluster di database Aurora, le istanze database e le relative impostazioni.

È consigliabile testare eventuali modifiche in un cluster di database o in un'istanza database di prova prima di modificare un cluster di database o un'istanza database di produzione, per comprendere pienamente l'impatto di ogni modifica. Ciò è importante soprattutto in caso di aggiornamento delle versioni di database.

Argomenti

- Modifica del cluster di database tramite la console, la CLI e l'API
- Modifica di un'istanza database in un cluster database
- Modifica della password per l'utente master del database
- Impostazioni per Amazon Aurora
- Impostazioni che non si applicano ai cluster di database Amazon Aurora
- Impostazioni che non si applicano alle istanze database Amazon Aurora

Modifica del cluster di database tramite la console, la CLI e l'API

È possibile modificare un cluster DB utilizzando l'API AWS Management Console AWS CLI, the o RDS.

> **Note**
>
> La maggior parte delle modifiche può essere applicata immediatamente o durante la successiva finestra di manutenzione programmata. Alcune modifiche, come l'attivazione della protezione da eliminazione, vengono applicate immediatamente, indipendentemente da quando scegli di applicarle.
> La modifica della password principale in AWS Management Console viene sempre applicata immediatamente.

Se utilizzi endpoint SSL e modifichi l'identificatore del cluster database, interrompi e riavvia il cluster database per aggiornare gli endpoint SSL. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.

Console

Per modificare un cluster di database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare il cluster di database che si desidera modificare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. Modificare le impostazioni desiderate. Per informazioni su ciascuna impostazione, consulta Impostazioni per Amazon Aurora.

   > **Note**
   >
   > In AWS Management Console, alcune modifiche a livello di istanza si applicano solo all'istanza DB corrente, mentre altre si applicano all'intero cluster DB. Per sapere se un'impostazione si applica all'istanza database o al cluster di database, vedi l'ambito relativo all'impostazione in Impostazioni per Amazon Aurora. Per modificare un'impostazione che modifica l'intero cluster DB a livello di istanza nel AWS Management Console, segui le istruzioni riportate inModifica di un'istanza database in un cluster database.

5. Quando tutte le modifiche sono come le desideri, seleziona Continue (Continua) e controlla il riepilogo delle modifiche.
6. Per applicare le modifiche immediatamente, selezionare Apply Immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify cluster (Modifica cluster) per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per modificare un cluster DB utilizzando il AWS CLI, chiamate il modify-db-clustercomando. Specifica l'identificatore del cluster di database e i valori per le impostazioni da modificare. Per informazioni su ciascuna impostazione, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano solo alle istanze database. Per modificare queste impostazioni, seguire le istruzioni in Modifica di un'istanza database in un cluster database.

Example

Il comando seguente modifica `mydbcluster` impostando il periodo di retention dei backup su 1 settimana (7 giorni).

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 7
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 7
```

API RDS

Per modificare un cluster DB utilizzando l'API Amazon RDS, chiama l'DBClusteroperazione Modify. Specifica l'identificatore del cluster di database e i valori per le impostazioni da modificare. Per informazioni su ciascun parametro, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano solo alle istanze database. Per modificare queste impostazioni, seguire le istruzioni in Modifica di un'istanza database in un cluster database.

Modifica di un'istanza database in un cluster database

Puoi modificare un'istanza DB in un cluster DB utilizzando l' AWS Management Console API AWS CLI, the o RDS.

Quando modifichi un'istanza database, puoi applicare le modifiche immediatamente. Per applicare immediatamente le modifiche, seleziona l'opzione Applica immediatamente in AWS Management Console, usi il `--apply-immediately` parametro quando chiami AWS CLI o imposti il `ApplyImmediately` parametro su `true` quando usi l'API Amazon RDS.

Se non si sceglie di applicare immediatamente le modifiche, le modifiche vengono rinviate alla finestra di manutenzione successiva. Durante la finestra di manutenzione successiva, vengono applicate tutte queste modifiche rinviate. Se si sceglie di applicare immediatamente le modifiche, le nuove modifiche verranno applicate insieme alle eventuali modifiche precedentemente rinviate.

Per visualizzare le modifiche in sospeso per la prossima finestra di manutenzione, usa il describe-db-clusters AWS CLI comando e controlla il campo. `PendingModifiedValues`

> **Important**
>
> Se nessuna delle modifiche rinviate richiede tempi di inattività, la selezione Apply immediately (Applica immediatamente) può provocare tempi di inattività imprevisti per l'istanza database. Non sono previsti tempi di inattività per le altre istanze database nel cluster di database.
> Le modifiche rinviate non sono elencate nell'output del comando CLI `describe-pending-maintenance-actions`. Le azioni di manutenzione includono solo gli aggiornamenti di sistema pianificati per la finestra di manutenzione successiva.

Console

Per modificare un'istanza database in un cluster di database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare l'istanza database che si desidera modificare.
3. Per Actions (Operazioni), scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).

4. Modificare le impostazioni desiderate. Per informazioni su ciascuna impostazione, consulta Impostazioni per Amazon Aurora.

   > **Note**
   >
   > Alcune impostazioni sono valide per l'intero cluster di database e devono essere modificate a livello di cluster. Per modificare queste impostazioni, seguire le istruzioni in Modifica del cluster di database tramite la console, la CLI e l'API.
   > In AWS Management Console, alcune modifiche a livello di istanza si applicano solo all'istanza DB corrente, mentre altre si applicano all'intero cluster DB. Per sapere se un'impostazione si applica all'istanza database o al cluster di database, vedi l'ambito relativo all'impostazione in Impostazioni per Amazon Aurora.

5. Quando tutte le modifiche sono come le desideri, seleziona Continue (Continua) e controlla il riepilogo delle modifiche.
6. Per applicare le modifiche immediatamente, selezionare Apply Immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, seleziona Modifica istanza database per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per modificare un'istanza DB in un cluster DB utilizzando il AWS CLI, chiamate il modify-db-instancecomando. Specifica l'identificatore istanze DB e i valori per le impostazioni da modificare. Per informazioni su ciascun parametro, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano all'intero cluster di database. Per modificare queste impostazioni, seguire le istruzioni in Modifica del cluster di database tramite la console, la CLI e l'API.

Example

Il codice seguente modifica `mydbinstance` impostando la classe di istanza database su `db.r4.xlarge`. Le modifiche vengono applicate durante la prossima finestra di manutenzione utilizzando `--no-apply-immediately`. Utilizza `--apply-immediately` per applicare immediatamente le modifiche.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r4.xlarge \
    --no-apply-immediately
```

In Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.r4.xlarge ^
    --no-apply-immediately
```

API RDS

Per modificare un'istanza DB utilizzando l'API Amazon RDS, chiama l'DBInstanceoperazione Modify. Specifica l'identificatore istanze DB e i valori per le impostazioni da modificare. Per informazioni su ciascun parametro, consulta Impostazioni per Amazon Aurora.

> **Note**
>
> Alcune impostazioni si applicano all'intero cluster di database. Per modificare queste impostazioni, seguire le istruzioni in Modifica del cluster di database tramite la console, la CLI e l'API.

Modifica della password per l'utente master del database

È possibile utilizzare AWS Management Console o AWS CLI per modificare la password dell'utente principale.

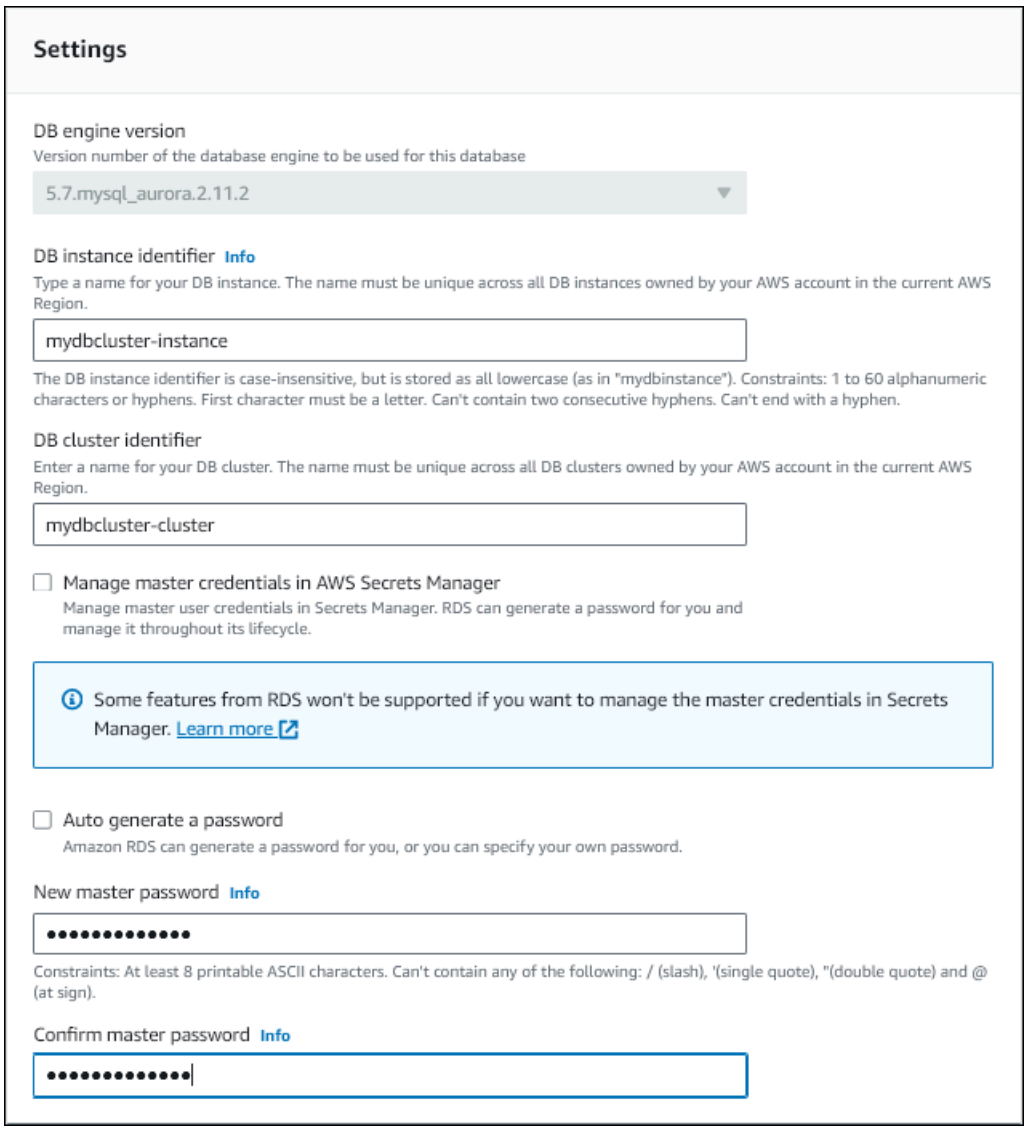
Console

È possibile modificare l'istanza DB di Writer per cambiare la password dell'utente principale utilizzando AWS Management Console.

Per modificare la password dell'utente principale

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare l'istanza database che si desidera modificare.
3. Per Actions (Operazioni), scegliere Modify (Modifica).

   Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
4. Inserisci una nuova password principale.
5. Per Conferma la password principale, inserisci la stessa nuova password.



6. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.

> **Note**
>
> Le modifiche alla password vengono sempre applicate immediatamente.

7. Nella pagina di conferma scegli Modify DB instance (Modifica istanza database).

CLI

Si chiama il modify-db-clustercomando per modificare la password dell'utente principale utilizzando il AWS CLI. Specificate l'identificatore del cluster DB e la nuova password, come illustrato negli esempi seguenti.

Non è necessario specificare`--apply-immediately|--no-apply-immediately`, poiché le modifiche alla password vengono sempre applicate immediatamente.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --master-user-password mynewpassword
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --master-user-password mynewpassword
```

Impostazioni per Amazon Aurora

La tabella seguente contiene informazioni dettagliate su quali impostazioni è possibile modificare, sui metodi per modificare l'impostazione e sull'ambito dell'impostazione. L'ambito determina se l'impostazione si applica all'intero cluster di database o se può essere impostata solo per specifiche istanze database.

> **Note**
>
> Sono disponibili impostazioni aggiuntive se si modifica un Aurora Serverless v1 oppure Aurora Serverless v2 Cluster DB. Per informazioni su queste impostazioni, consulta Modificare un Aurora Serverless v1 cluster di database e Gestione di Aurora Serverless v2 Cluster database.
> Alcune impostazioni non sono disponibili per Aurora Serverless v1 e Aurora Serverless v2 a causa delle loro limitazioni. Per ulteriori informazioni, consultare Limitazioni di Aurora Serverless v1 e Requisiti e limitazioni per Aurora Serverless v2.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Auto minor version upgrade (Aggiornamento automatico della versione secondaria) Specifica se l'istanza database deve ricevere automaticamente gli aggiornamenti della versione secondaria del motore preferiti quando diventano disponibili. Gli aggiornamenti vengono installati solo durante la finestra di manutenzione pianificata. Per ulteriori informazioni sugli aggiornamenti del motore, consulta Aggiornamenti del motore di database per Amazon Aurora Postgre SQL e Aggiornamenti del motore di database per Amazon Aurora My SQL. Per ulteriori informazioni sull'impostazione di Auto Minor Version	**Note** Questa impostazione è abilitata per impostazione predefinita. Per ogni nuovo cluster, scegli il valore appropriato per questa impostazione in base all'importanza, alla durata prevista e alla quantità di test di verifica eseguiti dopo ogni aggiornamento. Quando modifichi questa impostazione, esegui la modifica per ogni istanza database nel cluster Aurora. Se un'istanza database del cluster ha questa impostazione disattiva	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. Le interruzioni si verificano durante le finestre di manutenzione future quando Aurora applica gli aggiornamenti automatici.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Upgrade (Aggiornamento automatico minore della versione) per Aurora MySQL, consulta Abilitazione degli aggiornamenti automatici tra versioni minori di Aurora MySQL.	ta, il cluster non viene aggiornato automaticamente. Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui modify-db-instance e imposta l'`--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` opzione. Utilizzando l'API RDS, chiamate Modify DBInstance e impostate il `AutoMinorVersionUpgrade` parametro.		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Backup retention period (Periodo di retention dei backup) Numero di giorni durante i quali vengono conservati i backup automatici. Il valore minimo è 1. Per ulteriori informazioni, consulta Backup.	Utilizzando il AWS Management Console,Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui modify-db-clustere imposta l'`--backup-retention-period`opzione. Utilizzando l'API RDS, chiamate Modify DBCluster e impostate il `BackupRetentionPeriod` parametro.	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Finestra di backup (Ora di inizio) Intervallo di tempo in cui vengono eseguiti i backup automatici del database. L'ora di inizio della finestra di backup è indicata in base al formato Universal Coordinated Time (UTC) e la sua durata è in ore. I backup di Aurora sono continui e incrementali, tuttavia si utilizza una finestra di backup per creare un backup giornaliero del sistema che viene conservato per tutto il periodo di conservazione del backup. È anche possibile copiarlo per conservarlo al di fuori del periodo di conservazione. La finestra di manutenzione e la finestra di backup per il cluster di	Utilizzando il AWS Management Console,Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui modify-db-clustere imposta l'`--preferred-backup-window`opzione. Utilizzando l'API RDS, chiamate Modify DBCluster e impostate il `PreferredBackupWindow` parametro.	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
database non possono sovrapporsi. Per ulteriori informazioni, consulta Finestra di backup.			
Impostazioni della capacità Le proprietà di ridimensionamento di un Aurora Serverless v1 Cluster DB. Le proprietà di dimensionamento possono essere modificate solo per i cluster di database in modalità di motore DB `serverless` . Per informazioni su Aurora Serverless v1, consulta Usare Amazon Aurora Serverless v1.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui modify-db-cluster e imposta l'`--scaling-configuration` opzione. Utilizzando l'API RDS, chiamate Modify DBCluster e impostate il `ScalingConfiguration` parametro.	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. La modifica avviene immediatamente. Questa impostazione ignora l'impostazione Applica immediatamente.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Autorità di certificazione L'autorità di certificazione (CA) per il certificato del server utilizzato dall'istanza database.	Utilizzando il AWS Management Console,Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui modify-db-instancee imposta l'`--ca-certificate-identifier` opzione. Utilizzando l'API RDS, chiamate Modify DBInstance e impostate il `CACertificateIdentifier` parametro.	Solo l'istanza database specificata	Un'interruzione si verifica solo se il motore database non supporta la rotazione senza riavvio. È possibile utilizzare il describe-db-engine-versions AWS CLI comando per determinare se il motore DB supporta la rotazione senza riavvio.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Configurazione dell'archiviazione del cluster Il tipo di archiviazione per il cluster DB: Aurora I/O-Optimized o Aurora Standard. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui modify-db-cluster e imposta l'`--storage-type` opzione. Utilizzando l'API RDS, chiamate Modify DBCluster e impostate il `StorageType` parametro.	L'intero cluster di database	La modifica del tipo di archiviazione di un cluster Aurora PostgreSQL DB con classi di istanze Optimized Reads causa un'interruzione. Ciò non si verifica quando si modifica o i tipi di archiviazione per i cluster con altri tipi di classi di istanze. Per ulteriori informazioni sui tipi di classi di istanze DB, vedere Tipi di classi di istanza database.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Copy tags to snapshots (Copia tag in snapshot) Seleziona questa opzione per specificare di copiare i tag definiti per questo cluster di database in snapshot DB create da tale cluster di database. Per ulteriori informazioni, consulta Etichettatura di Amazon Aurora e risorse Amazon RDS.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando l'opzione AWS CLI, https://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-cluster.html esegui e imposta l'`--no-copy-tags-to-snapshot` opzione `--copy-tags-to-snapshot` or. Tramite l'API RDS, chiama https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_ModifyDBCluster.html e imposta il parametro `CopyTagsToSnapshot` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Data API (API dati). Puoi accedere Aurora Serverless v1 con applicazioni basate su servizi Web, tra cui e. AWS Lambda AWS AppSync Questa impostazione si applica solo a un Aurora Serverless v1 Cluster DB. Per ulteriori informazioni, consulta Utilizzo dell'API dati Amazon RDS.	Utilizzando AWS Management Console,Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--enable-http-endpoint` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `EnableHttpEndpoint` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Database authentication (Autenticazione del database) L'autenticazione del database da utilizzare. Per MySQL: • Scegliere Password authentication (Autenticazione tramite password) per autenticare gli utenti di database solo con le password del database. • Seleziona Autenticazione database tramite password e IAM per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli IAM. Per ulteriori informazioni, consulta Autenticazione del database IAM .	Utilizzando AWS Management Console,Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui modify-db-clustere imposta le seguenti opzioni: • Per l'autenticazione IAM, imposta l'opzione `--enable-iam-database-authentication|--no-enable-iam-database-authentication` . • Per l'autenticazione Kerberos imposta le opzioni `--domain` e `--domain-iam-role-name` . Utilizzando l'API RDS, chiamate Modify DBCluster e	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Per PostgreSQL: • Scegli IAM database authentication (Autenticazione database IAM) per autenticare gli utenti del database con password del database e credenziali utente tramite utenti e ruoli. Per ulteriori informazioni, consulta Autenticazione del database IAM . • Seleziona Autenticazione Kerberos per autenticare le password del database e le credenziali utente utilizzando l'autenticazione Kerberos. Per ulteriori informazioni, consulta Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL.	impostate i seguenti parametri: • Per l'autenticazione IAM, imposta il parametro `EnableIAMDatabaseAuthentication` . • Per l'autenticazione Kerberos, imposta i parametri `Domain` e `DomainIAMRoleName` .		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Database port (Porta del database) Porta da utilizzare per accedere al cluster di database.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--port` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `Port`.	L'intero cluster di database	Durante questa modifica si verifica un'interruzione. Tutte le istanze database nel cluster di database vengono riavviate immediatamente.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB Cluster Identifier (Identificatore cluster DB) L'identificatore del cluster di database. Questo valore è archiviato come stringa in caratteri minuscoli. Quando si modifica l'identificatore del cluster database, cambiano anche gli endpoint del cluster database. Gli endpoint delle istanze database nel cluster database non cambiano.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--new-db-cluster-identifier` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `NewDBClusterIdentifier`.	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB cluster parameter group (Gruppo di parametri del cluster database) Gruppo di parametri del cluster di database da associare al cluster di database. Per ulteriori informazioni, consulta .	Utilizzando AWS Management Console,Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--db-cluster-parameter-group-name` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `DBClusterParameterGroupName` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. Quando modifichi il gruppo di parametri, le modifiche apportate ad alcuni parametri vengono applicate alle istanze database nel cluster di database immediatamente senza un riavvio. Le modifiche apportate ad altri parametri vengono applicate solo dopo che le istanze database nel cluster di database vengono riavviate.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB instance class (Classe istanza database) Classe dell'istanza database da utilizzare. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--db-instance-class` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `DBInstanceClass`.	Solo l'istanza database specificata	Durante questa modifica si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB instance identifier (Identificatore istanze DB) Identificatore istanze DB. Questo valore è archiviato come stringa in caratteri minuscoli.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--new-db-instance-identifier` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `NewDBInstanceIdentifier`.	Solo l'istanza database specificata	Durante questa modifica si verifica un'interruzione. RDS riavvia l'istanza DB per aggiornare quanto segue: • Aurora MySQL — colonna nella tabella `SERVER_ID` `information_schema.replica_host_status` • Aurora PostgreSQL — colonna nella funzione `server_id` `aurora_replica_status()`

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
DB parameter group (Gruppo di parametri database) Gruppo di parametri database da associare all'istanza database. Per ulteriori informazioni, consulta .	Utilizzando,. AWS Management ConsoleModifica di un'istanza database in un cluster database Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--db-parameter-group-name`opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `DBParameterGroupName` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione. Quando si associa un nuovo gruppo parametri del database a un'istanza database, i parametri statici e dinamici modificati vengono applicati solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio. Per ulteriori informazioni, consulta e Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Deletion protection (Protezione da eliminazione) L'opzione Enable deletion protection (Abilita protezione da eliminazione) permette di impedire l'eliminazione del cluster di database. Per ulteriori informazioni, consulta Protezione dall'eliminazione per i cluster Aurora.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--deletion-protection|--no-deletion-protection` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `DeletionProtection` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Versione del motore La versione del motore di database da utilizzare. Prima di eseguire l'aggiornamento del cluster di database di produzione, è consigliabile testare il processo di aggiornamento in un cluster di database di test per verificarne la durata e convalidare le applicazioni.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--engine-version` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `EngineVersion` .	L'intero cluster di database	Durante questa modifica si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Enhanced Monitoring (Monitoraggio avanzato) L'opzione Enable enhanced monitoring (Abilita monitoraggio avanzato) permette di raccogliere i parametri in tempo reale per il sistema operativo in cui viene eseguita l'istanza database. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta le `--monitoring-interval` opzioni `--monitoring-role-arn` and. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta i parametri `MonitoringRoleArn` e `MonitoringInterval` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Log exports (Esportazioni log) Seleziona i tipi di log da pubblicare su Amazon CloudWatch Logs. Per ulteriori informazioni, consulta File di log del database Aurora MySQL.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--cloudwatch-logs-export-configuration` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `CloudwatchLogsExportConfiguration`.	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Maintenance window (Finestra di manutenzione) Intervallo di tempo durante cui viene eseguita la manutenzione del sistema. La manutenzione del sistema include gli aggiornamenti, se applicabili. L'ora di inizio della finestra di manutenzione è indicata in base al formato Universal Coordinated Time (UTC) e la sua durata è in ore. Se imposti la finestra sull'ora corrente, vi devono essere almeno 30 minuti tra l'ora corrente e la fine della finestra per assicurare che tutte le modifiche in sospeso vengano applicate. Puoi impostare la finestra di manutenzione in modo indipendente per il	Per modificare la finestra di manutenzione per il cluster DB utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Per modificare la finestra di manutenzione di un'istanza DB utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Per modificare la finestra di manutenzione per il cluster DB utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--preferred-maintenance-window` opzione. Per modificare la finestra di manutenzione di un'istanza DB utilizzando AWS CLI, esegui `modify-db-`	L'intero cluster di database o un'unica istanza database	Se sono presenti una o più operazioni in sospeso che causano un'interruzione e la finestra di manutenzione viene modificata per includere l'ora corrente, tali operazioni in sospeso vengono applicate immediatamente e si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
cluster di database e per ogni istanza database nel cluster di database. Quando l'ambito di una modifica è l'intero cluster di database, la modifica viene eseguita durante la finestra di manutenzione del cluster di database. Quando l'ambito di una modifica è un'istanza database, la modifica viene eseguita durante la finestra di manutenzione di tale istanza database. La finestra di manutenzione e la finestra di backup per il cluster di database non possono sovrapporsi. Per ulteriori informazioni, consulta Finestra di manutenzione di Amazon RDS.	`instance` e imposta l'`--preferred-maintenance-window` opzione. Per cambiare la finestra di manutenzione per il cluster di database tramite l'API RDS, chiamare `ModifyDBCluster` e impostare il parametro `PreferredMaintenanceWindow`. Per cambiare la finestra di manutenzione per un'istanza database tramite l'API RDS, chiamare `ModifyDBInstance` e impostare il parametro `PreferredMaintenanceWindow`.		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Gestisci le credenziali principali in AWS Secrets Manager Seleziona Manage master credentials in AWS Secrets Manager (Gestione credenziali master in AWS Secrets Manager) per gestire la password dell'utente master in un segreto di Secrets Manager. Facoltativamente, scegli la chiave KMS da utilizzare per proteggere il segreto. Scegliere tra le chiavi KMS presenti nell'account o inserire la chiave da un altro account. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager. Se Aurora sta già gestendo la password	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta le `--master-user-secret-kms-key-id` opzioni `--manage-master-user-password | --no-manage-master-user-password` and. Per ruotare immediatamente la password dell'utente master, imposta l'opzione `--rotate-master-user-password` . Tramite l'API RDS, chiama `ModifyDBCluster` e imposta i parametri `ManageMasterUserPassword` e `MasterUserSecretKmsKeyId` . Per ruotare immediata	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
dell'utente master per il cluster database, puoi ruotare la password dell'utente master scegliendo Rotate secret immediately (Ruota il segreto immediatamente). Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager.	mente la password dell'utente master, imposta il parametro `RotateMasterUserPassword` su `true`.		

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Tipo di rete I protocolli di indirizzo IP supportati dal cluster database. IPv4per specifica re che le risorse possono comunicare con il cluster DB solo tramite il protocollo di IPv4 indirizzamento. Modalità dual-stack per specificare che le risorse possono comunicare con il cluster DB tramite o IPv4 entrambi IPv6. Utilizza la modalità dual-stack se disponi di risorse che devono comunicar e con il cluster DB tramite il protocoll o di indirizzamento. IPv6 Per utilizzar e la modalità dual-stack, assicurati che almeno due sottoreti si estendano su due zone di disponibi lità che supportino sia il protocollo di	Utilizzando il,. AWS Management ConsoleModifica del cluster di database tramite la console, la CLI e l'API Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--network-type`opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `NetworkType` .	L'intero cluster di database	Durante questa modifica non si verifica un'interr uzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
rete che il protocollo di rete. IPv4 IPv6 Inoltre, assicurati di associare un blocco IPv6 CIDR alle sottoreti del gruppo di sottoreti DB specificato. Per ulteriori informazioni, consulta Assegnazione di indirizzi IP in Amazon Aurora.			

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
New master password (Nuova password master) Password dell'utente master. • Per Aurora MySQL, la password deve contenere da 8 a 41 caratteri ASCII stampabili. • Per Aurora PostgreSQL, deve contenere da 8 a 99 caratteri ASCII stampabili. • Non può contenere /, ", @ o uno spazio.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--master-user-password` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `MasterUserPassword` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Approfondimenti sulle prestazioni Specifica se abilitare Performance Insights, uno strumento che monitora il carico di istanze database per consentire di analizzare e risolvere i problemi di performance del database. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--enable-performance-insights|--no-enable-performance-insights` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `EnablePerformanceInsights` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Approfondimenti sulle prestazioni AWS KMS key L' AWS KMS key identificatore per la crittografia dei dati di Performance Insights. L'identificatore della chiave KMS è il nome della risorsa Amazon Resource Name (ARN), l'identificatore della chiave o l'alias della chiave per la chiave KMS. Per ulteriori informazioni, consulta Attivazione e disattivazione di Performance Insights per Aurora.	Utilizzando AWS Management Console,Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--performance-insights-kms-key-id` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PerformanceInsightsKMSKeyId` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Performance Insights retention period (Periodi di retention di Performance Insights) Quantità di tempo, espressa in giorni, per cui conservare i dati di Performance Insights. L'impostazione del periodo di conservazione nel livello gratuito è Default (7 days) (Impostazione predefinita (7 giorni)). Per mantenere i dati sulle prestazioni più a lungo, specifica da 1 a 24 mesi. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights. Per ulteriori informazioni, consulta Attivazione e disattivazione di Performance Insights per Aurora.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--performance-insights-retention-period` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PerformanceInsightsRetentionPeriod` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Promotion tier (Livello di promozione) Valore che specifica l'ordine di promozione di una replica di Aurora a istanza primaria in un cluster di database dopo un errore dell'istanza primaria esistente. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.	Utilizzando AWS Management Console, Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--promotion-tier` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PromotionTier` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Accesso pubblico Publicly accessibl e (Accessibile pubblicamente) per assegnare all'istanza database un indirizzo IP pubblico, ovvero renderla accessibile al di fuori del VPC. Per essere accessibi le pubblicamente, l'istanza database deve anche trovarsi in una sottorete pubblica nel VPC. Not publicly accessibl e (Non accessibi le pubblicamente) per rendere l'istanza database accessibile solo dall'interno del VPC. Per ulteriori informazi oni, consulta Nascondere cluster database in un VPC da Internet. Per connettersi a un'istanza DB dall'esterno Amazon VPC, l'istanza DB	Utilizzando AWS Management Console,Modifica di un'istanza database in un cluster database. Utilizzando AWS CLI, esegui `modify-db-instance` e imposta l'`--publicly-accessible|--no-publicly-accessibl e` opzione. Tramite l'API RDS, chiama `ModifyDBInstance` e imposta il parametro `PubliclyAccessible` .	Solo l'istanza database specificata	Durante questa modifica non si verifica un'interr uzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
deve essere accessibile pubblicamente, l'accesso deve essere concesso utilizzando le regole in ingresso del gruppo di sicurezza dell'istanza DB e devono essere soddisfatti altri requisiti. Per ulteriori informazioni, consulta Impossibile connettersi all'istanza database di Amazon RDS. Se la tua istanza DB non è accessibile al pubblico, puoi anche utilizzare una connessione AWS Site-to-Site VPN o una AWS Direct Connect connessione per accedervi da una rete privata. Per ulteriori informazioni, consulta Riservatezza del traffico Internet.			

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Serverless v2 impostazioni di capacità La capacità del database di un Aurora Serverless v2 Cluster DB, misurato in unità di capacità Aurora ()ACUs. Per ulteriori informazioni, consulta Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.	Utilizzando AWS Management Console,Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--serverless-v2-scaling-configuration` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `ServerlessV2ScalingConfiguration` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione. La modifica avviene immediatamente. Questa impostazione ignora l'impostazione Applica immediatamente.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Gruppo di sicurezza Gruppo di sicurezza da associare al cluster di database. Per ulteriori informazioni, consulta Controllo dell'accesso con i gruppi di sicurezza.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--vpc-security-group-ids` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `VpcSecurityGroupIds` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazione e descrizione	Metodo	Ambito	Note sui tempi di inattività
Target Backtrack window (Finestra di backtrack di destinazione) Quantità di tempo per cui è possibile eseguire il backtrack del cluster di database, espresso in secondi. Questa impostazione è disponibile solo per Aurora MySQL e solo se il cluster di database è stato creato con la funzionalità di backtrack abilitata.	Utilizzando AWS Management Console, Modifica del cluster di database tramite la console, la CLI e l'API. Utilizzando AWS CLI, esegui `modify-db-cluster` e imposta l'`--backtrack-window` opzione. Tramite l'API RDS, chiama `ModifyDBCluster` e imposta il parametro `BacktrackWindow` .	L'intero cluster di database	Durante questa modifica non si verifica un'interruzione.

Impostazioni che non si applicano ai cluster di database Amazon Aurora

Le seguenti impostazioni nel AWS CLI comando `modify-db-cluster`e nell'operazione dell'API RDS `ModifyDBCluster`non si applicano ai cluster Amazon Aurora DB.

> **Note**
>
> Non è possibile utilizzare il AWS Management Console per modificare queste impostazioni per i cluster Aurora DB.

AWS CLI impostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`

Impostazioni che non si applicano alle istanze database Amazon Aurora

Le seguenti impostazioni nel AWS CLI comando `modify-db-instance` e nell'operazione dell'API RDS `ModifyDBInstance` non si applicano alle istanze DB di Amazon Aurora.

> **Note**
>
> Non è possibile utilizzare il AWS Management Console per modificare queste impostazioni per le istanze Aurora DB.

AWS CLI impostazione	Impostazione API RDS
`--allocated-storage`	`AllocatedStorage`
`--allow-major-version-upgrade|--no-allow-major-version-upgrade`	`AllowMajorVersionUpgrade`
`--copy-tags-to-snapshot|--no-copy-tags-to-snapshot`	`CopyTagsToSnapshot`
`--domain`	`Domain`
`--db-security-groups`	`DBSecurityGroups`
`--db-subnet-group-name`	`DBSubnetGroupName`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--multi-az|--no-multi-az`	`MultiAZ`
`--iops`	`Iops`
`--license-model`	`LicenseModel`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--processor-features`	`ProcessorFeatures`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--use-default-processor-features|--no-use-default-processor-features`	`UseDefaultProcessorFeatures`

Aggiunta di repliche di Aurora a un cluster di database

Un cluster di database di Aurora con repliche ha un'istanza database primaria e fino a 15 repliche di Aurora. L'istanza database primaria supporta operazioni di lettura e scrittura ed esegue tutte le modifiche ai dati nel volume del cluster. Le repliche di Aurora si connettono allo stesso volume di storage dell'istanza database primaria ma supportano solo le operazioni di lettura. È possibile utilizzare le repliche di Aurora per effettuare l'offload dei carichi di lavoro in lettura dall'istanza database primaria. Per ulteriori informazioni, consulta Repliche di Aurora.

Le repliche di Amazon Aurora presentano le seguenti limitazioni:

- Non è possibile creare una replica Aurora per un Aurora Serverless v1 Cluster DB. Aurora Serverless v1 dispone di una singola istanza DB che si ridimensiona automaticamente verso l'alto e verso il basso per supportare tutte le operazioni di lettura e scrittura del database.

  Tuttavia, puoi aggiungere istanze di lettura a Aurora Serverless v2 Cluster DB. Per ulteriori informazioni, consulta Aggiungere un Aurora Serverless v2 reader.

Consigliamo di distribuire l'istanza primaria e le repliche di Aurora del cluster di database Aurora in più zone di disponibilità, in modo da migliorare la disponibilità del cluster di database. Per ulteriori informazioni, consulta Disponibilità nelle regioni.

Per rimuovere una replica di Aurora da un cluster di database Aurora, eliminare la replica di Aurora seguendo le istruzioni in Eliminazione di un'istanza database da un cluster database Aurora.

> **Note**
>
> Amazon Aurora supporta anche la replica con un database esterno, ad esempio un'istanza DB. RDS L'istanza RDS DB deve trovarsi nella stessa AWS Regione come Amazon Aurora. Per ulteriori informazioni, consulta Replica con Amazon Aurora.

È possibile aggiungere repliche Aurora a un cluster DB utilizzando AWS Management Console, il AWS CLI, o il RDSAPI.

Console

Per aggiungere una replica di Aurora a un cluster di database

1. Accedere a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database), quindi selezionare il cluster di database a cui aggiungere la nuova istanza database.
3. Assicurarsi che sia il cluster che l'istanza primaria siano nello stato Available (Disponibile) . Se il cluster DB o l'istanza primaria si trovano in uno stato di transizione, ad esempio Creating (Creazione), non è possibile aggiungere una replica.

   Se il cluster non dispone di un'istanza principale, creane una utilizzando create-db-instance AWS CLI comando. Questa situazione può verificarsi se si è utilizzato il CLI per ripristinare uno snapshot del cluster DB e quindi visualizzare il cluster nel AWS Management Console.
4. In Actions (Operazioni), scegliere Add reader (Aggiungi lettore).

   Viene visualizzata la pagina Add reader (Aggiungi lettore).
5. Nela pagina Add reader (Aggiungi lettore), specificare le opzioni per la replica Aurora. La tabella riportata di seguito mostra le impostazioni di una replica di Aurora.

Per questa opzione	Eseguire questa operazione
Availability zone (Zona di disponibilità)	Stabilire se si desidera specificare una zona di disponibilità particolare. L'elenco include solo le zone di disponibilità mappate al gruppo di sottoreti del database scelto al momento della creazione del cluster database. Per ulteriori informazioni sulle zone di disponibilità, consultare Regioni e zone di disponibilità.
Accessibile pubblicamente	Selezionare `Yes` per assegnare alla replica di Aurora un indirizzo IP pubblico. Altrimenti, selezionare `No`. Per ulteriori informazioni su come nascondere le repliche di Aurora per impedire l'accesso pubblico, consultare Nascondere cluster database in un VPC da Internet.

Per questa opzione	Eseguire questa operazione
Encryption (Crittografia)	Selezionare `Enable encryption` per abilitare la crittografia dei dati inattivi per questa replica di Aurora. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
DB instance class (Classe istanza database)	Selezionare una classe di istanze database che definisca i requisiti di elaborazione e di memoria per replica di Aurora. Per ulteriori informazioni sulle opzioni di classe di istanza database, consulta Classi di istanze DB Amazon Aurora.
Aurora replica source (Origine replica di)	Selezionare l'identificatore dell'istanza primaria per cui creare una replica di Aurora.
DB instance identifier (Identificatore istanze DB)	Inserisci un nome per l'istanza che sia unico per il tuo account nel AWS Regione selezionata. Potresti scegliere di aggiungere alcune informazioni al nome, ad esempio includere il AWS La regione e il motore DB selezionati, ad esempio**`aurora-read-instance1`** .
Priorità	Scegliere una priorità di failover per l'istanza. Se non si specifica alcun valore, l'impostazione predefinita è tier-1 (livello 1). Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.
Database port (Porta del database)	La porta di una replica di Aurora corrisponde a quella per il cluster di database.

Per questa opzione	Eseguire questa operazione
DB parameter group (Gruppo di parametri database)	Selezionare un gruppo di parametri. Si può utilizzare gruppo di parametri predefiniti forniti da Aurora oppure creare un gruppo di parametri personalizzato. Per ulteriori informazioni sui gruppi di parametri, consultare .
Approfondimenti sulle prestazioni	La casella di controllo Turn on Performance Insights (Attiva Performance Insights) è selezionata per impostazione predefinita. Il valore non viene ereditato dall'istanza di scrittura. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
Enhanced Monitoring (Monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Scegli il IAM ruolo che hai creato per consentire RDS ad Amazon di comunicare con Amazon CloudWatch Logs per te oppure scegli Default per RDS creare un ruolo a te assegnato. `rds-monitoring-role` Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.

Per questa opzione	Eseguire questa operazione
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Selezionare Enable auto minor version upgrade (Abilita aggiornamenti automatici versioni minori) per abilitare il cluster di database Aurora in modo che riceva automaticamente gli aggiornamenti minori della versione del motore di database non appena diventano disponibili. L'impostazione di aggiornamento automatico della versione secondaria si applica ai cluster Aurora SQL Postgre e Aurora My DB. SQL Per i cluster Aurora My SQL 2.x, questa impostazione aggiorna i cluster a una versione massima di 2.07.2. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora SQL Postgre, vedere. Aggiornamenti del motore di database per Amazon Aurora Postgre SQL Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consulta. Aggiornamenti del motore di database per Amazon Aurora My SQL

6. Scegliere Add reader (Aggiungi lettore) per creare la replica Aurora.

AWS CLI

Per creare una replica Aurora nel tuo cluster DB, esegui create-db-instance AWS CLI comando. Includere il nome del cluster di database come opzione `--db-cluster-identifier`. È anche possibile specificare una zona di disponibilità per la replica di Aurora utilizzando il parametro `--availability-zone`, come mostrato negli esempi seguenti.

Ad esempio, il comando seguente crea una nuova Replica Aurora SQL compatibile con My 5.7 denominata. `sample-instance-us-west-2a`

In Linux, macOS, oppure Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
```

```
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

In Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

Il comando seguente crea una nuova Replica Aurora SQL compatibile con My 5.7 denominata. `sample-instance-us-west-2a`

In Linux, macOS, oppure Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

In Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

Il comando seguente crea una nuova Replica Aurora SQL compatibile con Postgre denominata. `sample-instance-us-west-2a`

In Linux, macOS, oppure Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large \
    --availability-zone us-west-2a
```

In Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
```

```
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large ^
    --availability-zone us-west-2a
```

RDS API

Per creare una replica Aurora nel tuo cluster DB, chiama l'operazione C. reateDBInstance
Includere il nome del cluster di database come parametro `DBClusterIdentifier`. È anche possibile specificare una zona di disponibilità per la replica di Aurora utilizzando il parametro `AvailabilityZone`.

Per informazioni sull'Auto Scaling di Amazon Aurora con repliche Aurora, consulta le seguenti sezioni.

Argomenti

- Auto Scaling di Amazon Aurora con repliche Aurora
- Aggiungere una policy di scalabilità automatica a un cluster Amazon Aurora DB
- Modifica di una politica di scalabilità automatica per un cluster Amazon Aurora DB
- Eliminazione di una politica di scalabilità automatica dal cluster Amazon Aurora DB

Auto Scaling di Amazon Aurora con repliche Aurora

Per soddisfare i requisiti di connettività e carico di lavoro, il dimensionamento automatico Aurora modifica dinamicamente il numero di repliche Aurora (istanze database di lettura) assegnato a un cluster di database Aurora. Aurora Auto Scaling è disponibile per Aurora MySQL e Aurora PostgreSQL. Aurora Auto Scaling consente al cluster di database Aurora di gestire gli aumenti improvvisi di connettività o del carico di lavoro. Quando la connettività o il carico di lavoro diminuiscono, Aurora Auto Scaling rimuove le repliche Aurora inutili così da evitare di pagare per le istanze database non utilizzate.

Si definisce e si applica una politica di ridimensionamento a un cluster del database Aurora. La policy di dimensionamento definisce il numero minimo e massimo di repliche Aurora che Aurora Auto Scaling riesce a gestire. In base alla policy, Aurora Auto Scaling aumenta o diminuisce il numero di repliche Aurora in risposta ai carichi di lavoro effettivi, determinati utilizzando i parametri e i valori target di Amazon. CloudWatch

> **Note**
>
> Aurora Auto Scaling non si applica al carico di lavoro sull'istanza Writer DB. Aurora Auto Scaling aiuta con il carico di lavoro solo sulle istanze del lettore.

È possibile utilizzare il AWS Management Console per applicare una politica di scalabilità basata su una metrica predefinita. In alternativa, puoi utilizzare l'API Aurora Auto Scaling per applicare una politica di scalabilità basata su una metrica predefinita AWS CLI o personalizzata.

Argomenti

- Prima di iniziare
- Policy di dimensionamento automatico Aurora
- Istanza IDs DB e tagging
- Dimensionamento automatico Aurora e Approfondimenti sulle prestazioni

Prima di iniziare

Prima di poter utilizzare il dimensionamento automatico Aurora con il cluster di database Aurora è necessario creare un cluster di database Aurora con un'istanza database (di scrittura) primaria.

Per ulteriori informazioni sulla creazione di un cluster di database Aurora, consulta Creazione di un cluster database Amazon Aurora.

Il dimensionamento automatico Aurora dimensiona un cluster di database solo se è nello stato disponibile.

Quando Aurora Auto Scaling aggiunge una nuova replica Aurora, la nuova replica Aurora è la stessa classe di istanze database di quella utilizzata dall'istanza primaria. Per altre informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora. Inoltre, il livello di promozione per le nuove repliche Aurora è impostato sull'ultima priorità, che per impostazione predefinita è 15. Ciò significa che durante un failover una replica con una priorità più alta, ad esempio una creata manualmente, sarà promossa per prima. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.

Aurora Auto Scaling rimuove solo le repliche Aurora che ha creato.

Per sfruttare Aurora Auto Scaling, le applicazioni devono supportare le connessioni a nuove repliche Aurora. Per farlo, consigliamo di utilizzare l'endpoint di lettura Aurora. È possibile utilizzare un driver come il driver JDBC. AWS Per ulteriori informazioni, consulta Connessione a un cluster database Amazon Aurora.

> **Note**
>
> I database globali Aurora attualmente non supportano l'Auto Scaling Aurora per i cluster di database secondari.

Policy di dimensionamento automatico Aurora

Aurora Auto Scaling utilizza una policy di dimensionamento per regolare il numero di repliche Aurora in un cluster di database Aurora. Aurora Auto Scaling ha le seguenti componenti:

- Un ruolo collegato al servizio
- UN parametro target
- Capacità minima e massima
- Un periodo di attesa

Argomenti

- Ruolo legato al servizio
- Parametro target
- Capacità minima e massima
- Periodo di attesa
- Abilitazione o disabilitazione delle attività di riduzione
- Aggiungere, modificare o eliminare le politiche di ridimensionamento automatico

Ruolo legato al servizio

Aurora Auto Scaling utilizza il ruolo collegato al servizio `AWSServiceRoleForApplicationAutoScaling_RDSCluster`. Per ulteriori informazioni, consulta Ruoli collegati ai servizi per Application Auto Scaling nella Guida per l'utente di Application Auto Scaling.

Parametro target

In questo tipo di policy, vengono specificati una metrica personalizzata o predefinita e un valore di destinazione per la metrica in una configurazione della policy di dimensionamento di monitoraggio delle destinazioni. Aurora Auto Scaling crea e CloudWatch gestisce allarmi che attivano la politica di scalabilità e calcola la regolazione della scalabilità in base alla metrica e al valore target. La policy di dimensionamento aggiunge o rimuove le repliche Aurora come richiesto per mantenere il parametro al valore di destinazione specificato o vicino a esso. Oltre a mantenere il parametro vicino al valore di destinazione, una policy di dimensionamento del monitoraggio di destinazione si adatta anche alle oscillazioni del parametro dovute a un carico di lavoro mutevole. Tale policy riduce anche le fluttuazioni rapide nel numero di repliche Aurora disponibili per il cluster di database.

Ad esempio, prendi una policy di dimensionamento che usa la metrica di utilizzo della CPU media predefinita. Tale policy può mantenere l'utilizzo della CPU a una percentuale specifica di utilizzo, come il 40 percento, o vicino ad essa.

> **Note**
>
> Per ogni cluster di database Aurora, è possibile creare solo una policy di Auto Scaling per ogni parametro di destinazione.

Quando configuri Aurora Auto Scaling, il valore della metrica di destinazione viene calcolato come media di tutte le istanze di lettura nel cluster. Questo calcolo viene eseguito come segue:

- Include tutte le istanze di lettura nel cluster Aurora, indipendentemente dal fatto che siano gestite da Auto Scaling o aggiunte manualmente.
- Include istanze associate a endpoint personalizzati. Gli endpoint personalizzati non influiscono sul calcolo delle metriche target.
- Non include l'istanza writer del cluster.

Le metriche derivano dall' CloudWatch utilizzo delle seguenti dimensioni:

- `DBClusterIdentifier`
- `Role=READER`

Ad esempio, considera un cluster Aurora MySQL con la seguente configurazione:

- Istanze manuali (non controllate da Auto Scaling):
  - Writer con utilizzo della CPU al 50%
  - Reader 1 (endpoint personalizzato:`custom-reader-1`) con un utilizzo della CPU del 90%
  - Reader 2 (endpoint personalizzato:`custom-reader-2`) con utilizzo della CPU al 90%
- Istanza Auto Scaling:
  - Reader 3 (aggiunto utilizzando Auto Scaling) con un utilizzo della CPU del 10%

In questo scenario, la metrica target per la policy Auto Scaling viene calcolata come segue:

```
Target metric = (CPU utilization of reader 1 + reader 2 + reader 3) / total number of
 readers

Target metric = (90 + 90 + 10) / 3 = 63.33%
```

La politica Auto Scaling utilizza questo valore per valutare se scalare in avanti o in orizzontale in base alla soglia definita.

Considera i seguenti aspetti:

- Sebbene gli endpoint personalizzati determinino il modo in cui il traffico viene indirizzato a lettori specifici, non escludono i lettori dal calcolo della metrica.
- Le istanze manuali sono sempre incluse nei calcoli delle metriche target.

- Per evitare comportamenti di scalabilità imprevisti, assicurati che la configurazione Auto Scaling tenga conto di tutte le istanze di lettura nel cluster.
- Se un cluster non ha lettori, la metrica non viene calcolata e gli allarmi della policy Auto Scaling rimangono inattivi. Affinché la policy Auto Scaling funzioni in modo efficace, deve essere sempre presente almeno un lettore.

Capacità minima e massima

È possibile specificare il numero massimo di repliche Aurora che deve essere gestito da Application Auto Scaling. Questo valore deve essere impostato su 0–15 e deve essere uguale o maggiore rispetto al valore specificato per il numero minimo di repliche Aurora.

È anche possibile specificare il numero minimo di repliche Aurora che deve essere gestito da Application Auto Scaling. Questo valore deve essere impostato su 0–15 e deve essere uguale o minore rispetto al valore specificato per il numero massimo di repliche Aurora.

È necessario che sia presente almeno un'istanza Reader DB affinché Aurora Auto Scaling funzioni. Se il cluster DB non ha un'istanza di lettura e imposti la capacità minima su 0, Aurora Auto Scaling non funzionerà.

> **Note**
>
> Sono impostate le capacità minima e massima per un cluster di database Aurora. I valori specificati vengono applicati a tutte le policy associate con quel cluster di database Aurora.

Periodo di attesa

È possibile sintonizzare i tempi di risposta di una policy di dimensionamento di monitoraggio della destinazione aggiungendo dei periodi di attesa che influiscano sul ridimensionamento del cluster di database Aurora in entrata e in uscita. Un periodo di attesa blocca le richieste di riduzione o aumento ulteriori finché il periodo non scade. Questi blocchi rallentano le eliminazioni delle repliche Aurora nel cluster di database Aurora per le richieste di riduzione e la creazione di repliche Aurora per le richieste di aumento.

Puoi specificare i seguenti periodi di attesa:

- Un'attività di riduzione riduce il numero di repliche Aurora nel cluster di database Aurora. Un periodo di attesa di riduzione specifica la quantità di tempo che deve passare, in secondi, tra il completamento di un'attività di riduzione e l'inizio di un'altra attività di questo tipo.
- Un'attività di aumento aumenta il numero di repliche Aurora nel cluster di database Aurora. Un periodo di attesa di aumento specifica la quantità di tempo che deve passare, in secondi, tra il completamento di un'attività di aumento e l'inizio di un'altra attività di questo tipo.

> **Note**
>
> Un tempo di raffreddamento di aumento orizzontale viene ignorato se una successiva richiesta di aumento orizzontale è relativa a un numero maggiore di repliche Aurora rispetto alla prima richiesta.

Se non si imposta un tempo di raffreddamento di riduzione orizzontale o aumento orizzontale, il valore predefinito per ciascuno è pari a 300 secondi.

Abilitazione o disabilitazione delle attività di riduzione

Puoi abilitare o disabilitare le attività di riduzione per una policy. Abilitare queste attività di riduzione consente alla policy di dimensionamento di eliminare le repliche Aurora. Quando le attività di riduzione sono abilitate, il periodo di attesa della riduzione nella policy di dimensionamento si applica alle attività di riduzione. Disabilitare le attività di riduzione evita alla policy di dimensionamento di eliminare le repliche Aurora.

> **Note**
>
> Le attività di aumento sono sempre abilitate, in modo che la policy di dimensionamento possa creare repliche Aurora in base alle esigenze.

Aggiungere, modificare o eliminare le politiche di ridimensionamento automatico

È possibile aggiungere, modificare o eliminare le policy di auto scaling utilizzando o l' AWS Management Console API AWS CLI Application Auto Scaling. Per ulteriori informazioni sull'aggiunta, la modifica o l'eliminazione delle politiche di ridimensionamento automatico, vedere le seguenti sezioni.

- Aggiungere una policy di scalabilità automatica a un cluster Amazon Aurora DB

- Modifica di una politica di scalabilità automatica per un cluster Amazon Aurora DB
- Eliminazione di una politica di scalabilità automatica dal cluster Amazon Aurora DB

Istanza IDs DB e tagging

Quando una replica viene aggiunta da Aurora Auto Scaling, il relativo ID istanza database è preceduto da `application-autoscaling-`, ad esempio `application-autoscaling-61aabbcc-4e2f-4c65-b620-ab7421abc123`.

Il tag seguente viene aggiunto automaticamente all'istanza di database. È possibile visualizzarlo nella scheda Tag della pagina dei dettagli dell'istanza di database.

Tag	Valore
application-autoscaling:resourceId	cluster:mynewcluster-cluster

Per ulteriori informazioni sui tag delle risorse di Amazon RDS, consulta Etichettatura di Amazon Aurora e risorse Amazon RDS.

Dimensionamento automatico Aurora e Approfondimenti sulle prestazioni

È possibile utilizzare Approfondimenti sulle prestazioni per monitorare le repliche aggiunte da Dimensionamento automatico Aurora, come con qualsiasi istanza database di lettura Aurora.

Per ulteriori informazioni sull'Approfondimenti sulle prestazioni per monitorare i cluster di database Aurora, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Aggiungere una policy di scalabilità automatica a un cluster Amazon Aurora DB

È possibile aggiungere una politica di scalabilità utilizzando Application Auto Scaling AWS CLI, o Application Auto API Scaling. AWS Management Console

> **Note**
>
> Per un esempio che aggiunge una politica di scalabilità utilizzando AWS CloudFormation, consulta Dichiarazione di una politica di scalabilità per un cluster Aurora DB nella Guida per l'utente.AWS CloudFormation

Console

È possibile aggiungere una politica di scalabilità a un cluster Aurora DB utilizzando. AWS Management Console

Per aggiungere una policy di Auto Scaling a un cluster di database Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegliere il cluster di database Aurora a cui si desidera aggiungere una policy.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Nella sezione Auto scaling policies (Policy di Auto Scaling), scegliere Add (Aggiungi).

   Appare la finestra di dialogo Add Auto Scaling policy (Aggiungi policy di Auto Scaling).
6. Per Policy name (Nome policy), digitare il nome della policy.
7. Per il parametro di destinazione, scegliere in uno dei seguenti modi:

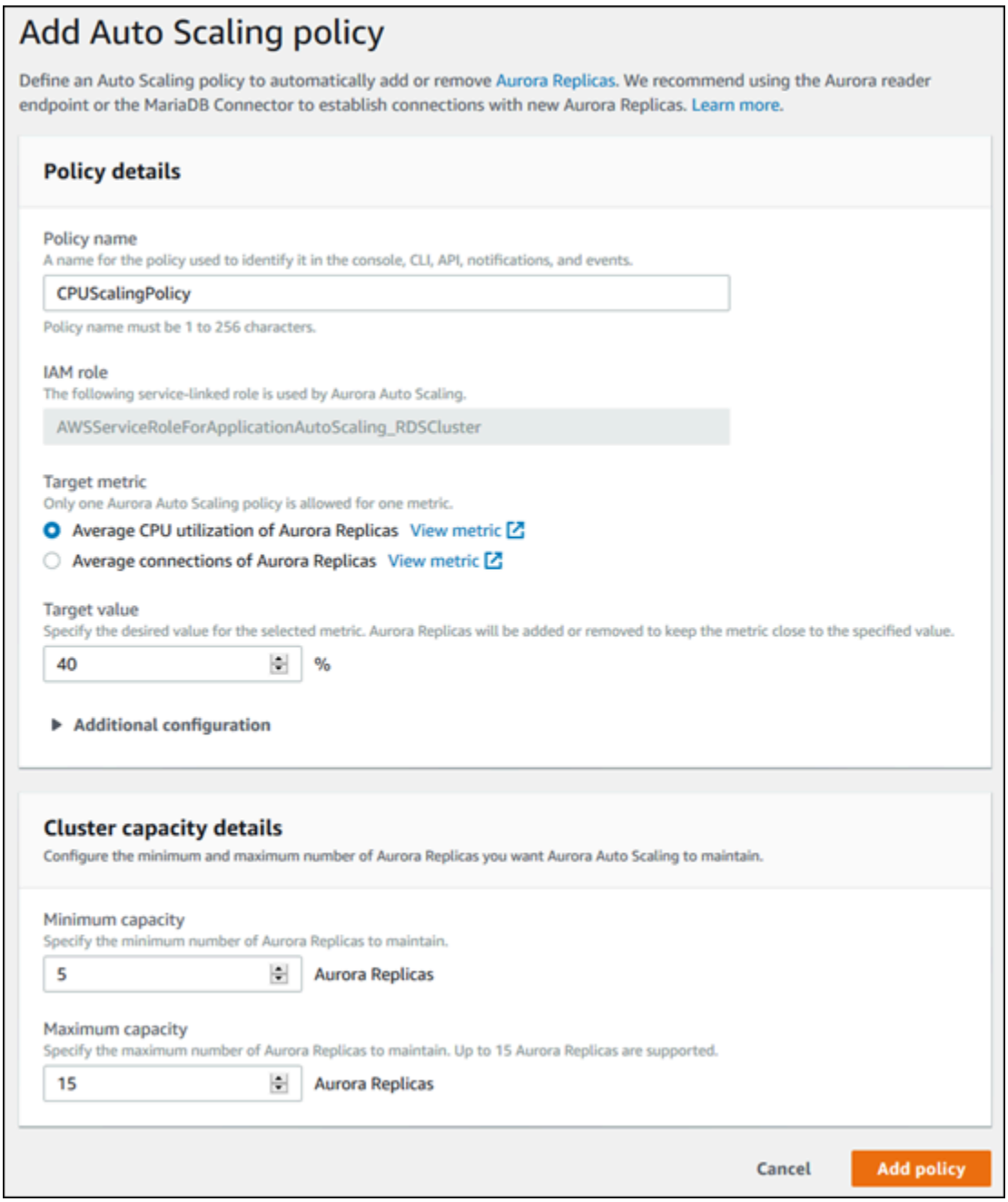
   - CPUUtilizzo medio di Aurora Replicas per creare una policy basata sull'utilizzo medio. CPU
   - Average connections of Aurora Replicas (Connessioni medie delle repliche Aurora) per creare una policy basata sul numero medio di connessioni alle repliche Aurora.
8. Per il valore di destinazione, digitare uno dei seguenti modi:

   - Se hai scelto CPUUtilizzo medio delle repliche Aurora nel passaggio precedente, digita la percentuale CPU di utilizzo che desideri mantenere sulle repliche Aurora.
   - Se si sceglie Average connections of Aurora Replicas (Connessioni medie delle repliche Aurora), digitare il numero di connessioni da mantenere.

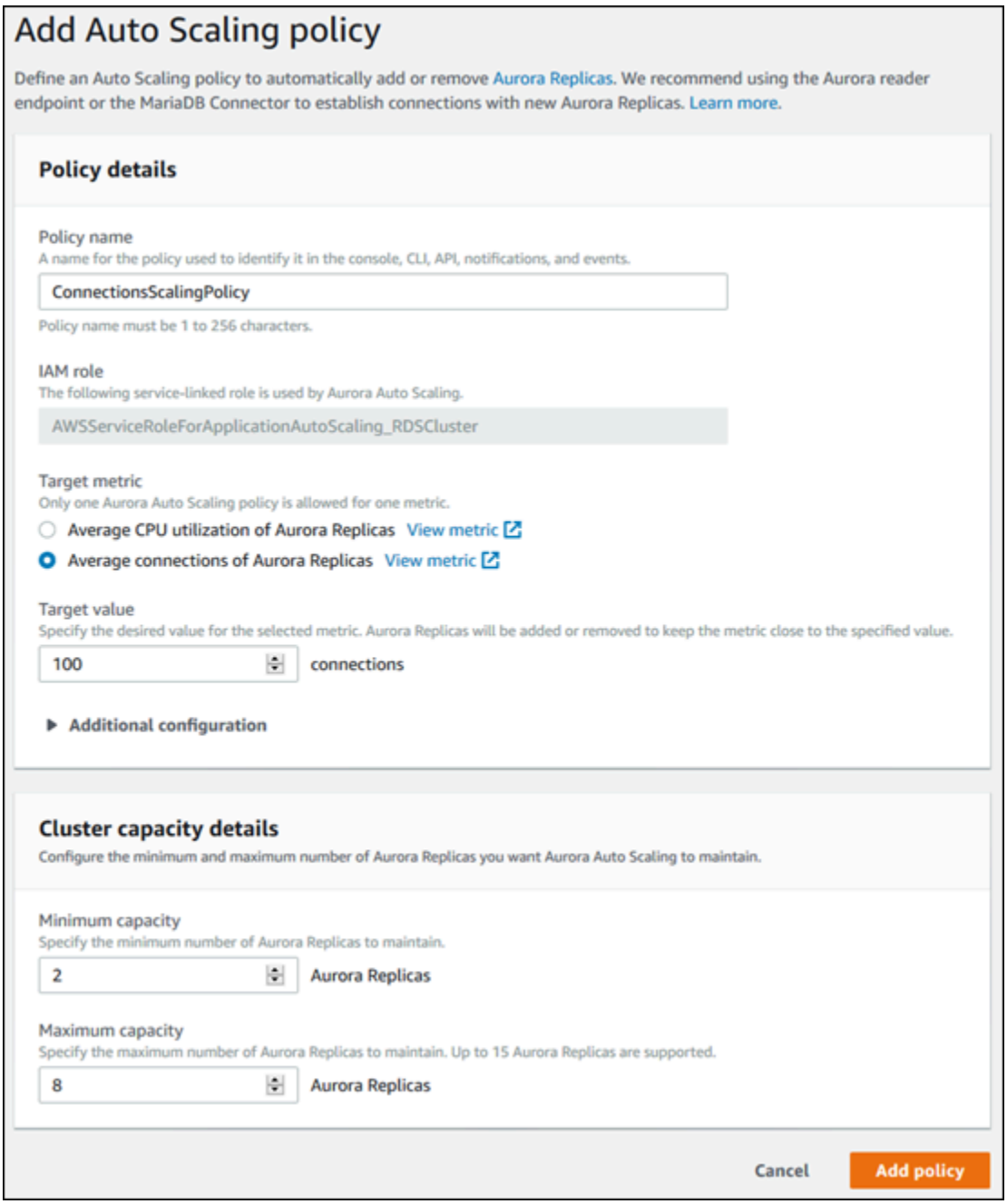
   Le repliche Aurora vengono aggiunte o rimosse per tenere il parametro vicino al valore specificato.

9. (Facoltativo) Espandere Configurazione aggiuntiva per creare un tempo di raffreddamento di riduzione orizzontale o aumento orizzontale.
10. Per Minimum capacity (Capacità minima), digitare il numero minimo di repliche Aurora che la policy di Aurora Auto Scaling deve mantenere.
11. Per Maximum capacity (Capacità massima), digitare il numero massimo di repliche Aurora che la policy di Aurora Auto Scaling deve mantenere.
12. Scegliere Add policy (Aggiungi policy).

La seguente finestra di dialogo crea una politica di Auto Scaling basata su un CPU utilizzo medio del 40 percento. La policy specifica un minimo di 5 repliche Aurora e un massimo di 15 repliche Aurora.



La seguente finestra di dialogo crea una policy di Auto Scaling basata su un numero di connessioni pari a 100. La policy specifica un minimo di due repliche Aurora e un massimo di otto repliche Aurora.



AWS CLI o Application Auto Scaling API

Puoi applicare una policy di dimensionamento basata un parametro di default o personalizzato. A tale scopo, è possibile utilizzare AWS CLI o l'Application Auto ScalingAPI. La prima fase consiste nel registrare il cluster di database Aurora con Application Auto Scaling.

Registrazione di un cluster di database Aurora

Prima di poter utilizzare Aurora Auto Scaling con un cluster di database Aurora, registrare il cluster di database Aurora con Application Auto Scaling. Questa operazione consente di definire la dimensione e i limiti di dimensionamento da applicare al cluster. Application Auto Scaling ridimensiona dinamicamente il cluster di database Aurora lungo la dimensione scalabile `rds:cluster:ReadReplicaCount`, che rappresenta il numero di repliche Aurora.

Per registrare il cluster Aurora DB, è possibile utilizzare AWS CLI o l'Application Auto Scaling. API

AWS CLI

Per registrare il tuo cluster Aurora DB, usa il `register-scalable-target` AWS CLI comando con i seguenti parametri:

- `--service-namespace` – Impostare questo valore su `rds`.
- `--resource-id` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `--scalable-dimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `--min-capacity` – Il numero minimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `--min-capacity`, `--max-capacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.
- `--max-capacity` – Il numero massimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `--min-capacity`, `--max-capacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.

Example

Nell'esempio seguente viene registrato un cluster di database Aurora denominato `myscalablecluster`. La registrazione indica che il cluster di database deve essere dimensionato dinamicamente per avere da uno a otto repliche Aurora.

Per LinuxmacOS, oUnix:

```
aws application-autoscaling register-scalable-target \
    --service-namespace rds \
    --resource-id cluster:myscalablecluster \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

```
    --min-capacity 1 \
    --max-capacity 8 \
```

Per Windows:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace rds ^
    --resource-id cluster:myscalablecluster ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --min-capacity 1 ^
    --max-capacity 8 ^
```

Application Auto Scaling API

Per registrare il cluster Aurora DB con Application Auto Scaling, utilizzate l'operazione Application Auto `RegisterScalableTarget`Scaling con i seguenti API parametri:

- `ServiceNamespace` – Impostare questo valore su `rds`.
- `ResourceID` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `ScalableDimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `MinCapacity` – Il numero minimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `MinCapacity`, `MaxCapacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.
- `MaxCapacity` – Il numero massimo di istanze database del lettore da gestire con Application Auto Scaling. Per informazioni sulla relazione tra `MinCapacity`, `MaxCapacity` e il numero di istanze database nel cluster, consulta Capacità minima e massima.

Example

Nell'esempio seguente, si registra un cluster Aurora DB denominato `myscalablecluster` Application Auto Scaling. API Questa registrazione indica che il cluster di database deve essere dimensionato dinamicamente per avere da uno a otto repliche Aurora.

```
POST / HTTP/1.1
```

```
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "MinCapacity": 1,
    "MaxCapacity": 8
}
```

Definizione di una politica di dimensionamento per un cluster di database Aurora

Una configurazione di policy di scaling target-tracking è rappresentata da un JSON blocco in cui sono definite le metriche e i valori di destinazione. È possibile salvare una configurazione della politica di scalabilità come JSON blocco in un file di testo. Si utilizza quel file di testo quando si richiama il AWS CLI o l'Application Auto API Scaling. Per ulteriori informazioni sulla sintassi di configurazione delle policy, vedere `TargetTrackingScalingPolicyConfiguration`Application Auto API Scaling Reference.

Le seguenti opzioni sono disponibili per definire una configurazione di una policy di dimensionamento di monitoraggio dei target.

Argomenti

- Utilizzo di un parametro predefinito
- Utilizzo di un parametro personalizzato
- Utilizzo di periodi di attesa
- Disabilitazione dell'attività di riduzione

Utilizzo di un parametro predefinito

Utilizzando dei parametri predefiniti, si può definire rapidamente una policy di dimensionamento per il monitoraggio della destinazione per un cluster di database Aurora che funziona bene sia con il monitoraggio della destinazione sia con il dimensionamento dinamico in Aurora Auto Scaling.

Attualmente, Aurora supporta i seguenti parametri predefiniti in Aurora Auto Scaling:

- RDSReaderAverageCPUUtilization— Il valore medio della `CPUUtilization` metrica in CloudWatch tutte le repliche Aurora nel cluster Aurora DB.
- RDSReaderAverageDatabaseConnections— Il valore medio della `DatabaseConnections` metrica in CloudWatch tutte le repliche Aurora nel cluster Aurora DB.

Per ulteriori informazioni sui parametri `CPUUtilization` e `DatabaseConnections`, consultare CloudWatch Parametri Amazon per Amazon Aurora.

Per utilizzare un parametro di default nella policy di dimensionamento, crea una configurazione di monitoraggio degli obiettivi per la policy di dimensionamento. La configurazione deve includere un `PredefinedMetricSpecification` per il parametro predefinito e un `TargetValue` per il valore di destinazione del parametro.

Example

L'esempio seguente descrive una tipica configurazione di policy per il dimensionamento di monitoraggio della destinazione per un cluster di database Aurora. In questa configurazione, la metrica `RDSReaderAverageCPUUtilization` predefinita viene utilizzata per regolare il cluster Aurora DB in base a un CPU utilizzo medio del 40 percento su tutte le repliche Aurora.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    }
}
```

Utilizzo di un parametro personalizzato

Utilizzando dei parametri personalizzati, è possibile definire una policy di dimensionamento di monitoraggio della destinazione che soddisfi i requisiti personalizzati. È possibile definire un parametro personalizzato in base a qualsiasi parametro Aurora che si modifichi in proporzione al dimensionamento.

Non tutti i parametri Aurora funzionano per il monitoraggio della destinazione. Il parametro deve essere un parametro di utilizzo valido e deve descrivere quanto è impegnata un'istanza. Il valore

del parametro deve aumentare o diminuire in proporzione al numero di repliche Aurora nel cluster di database Aurora. Questo aumento o riduzione proporzionale è necessario per utilizzare i dati del parametro per aumentare o diminuire in modo proporzionale il numero di repliche Aurora.

Example

Il seguente esempio descrive una configurazione di monitoraggio della destinazione per una policy di dimensionamento. In questa configurazione, una metrica personalizzata regola un cluster Aurora DB in base a un CPU utilizzo medio del 50 percento su tutte le repliche Aurora in un cluster Aurora DB denominato. `my-db-cluster`

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/RDS",
        "Dimensions": [
            {"Name": "DBClusterIdentifier","Value": "my-db-cluster"},
            {"Name": "Role","Value": "READER"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

Utilizzo di periodi di attesa

È possibile specificare un valore, in secondi, per `ScaleOutCooldown` per aggiungere un periodo di attesa per l'aumento del cluster di database Aurora. Allo stesso modo, è possibile aggiungere un valore, in secondi, per `ScaleInCooldown` per aggiungere un periodo di attesa per la riduzione del cluster di database Aurora. Per ulteriori informazioni su `ScaleInCooldown` e`ScaleOutCooldown`, vedere `TargetTrackingScalingPolicyConfiguration`Application Auto Scaling API Reference.

Example

Il seguente esempio descrive una configurazione di monitoraggio degli obiettivi per una policy di dimensionamento. In questa configurazione, la metrica `RDSReaderAverageCPUUtilization` predefinita viene utilizzata per regolare un cluster Aurora DB in base a un CPU utilizzo medio del 40 percento su tutte le repliche Aurora in quel cluster Aurora DB. La configurazione fornisce un tempo

di raffreddamento di riduzione orizzontale di 10 minuti e un tempo di raffreddamento di aumento di 5 minuti.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Disabilitazione dell'attività di riduzione

Puoi prevenire la configurazione della policy di dimensionamento di monitoraggio della destinazione riducendo il cluster di database Aurora disabilitando l'attività di riduzione. La disabilitazione delle attività di riduzione impedisce alla policy di dimensionamento di eliminare le repliche Aurora, consentendo tuttavia alla policy di dimensionamento di crearle in base alle esigenze.

È possibile specificare un valore booleano per `DisableScaleIn` per abilitare o disabilitare l'attività di riduzione per il cluster di database Aurora. Per ulteriori informazioni `TargetTrackingScalingPolicyConfiguration`in merito`DisableScaleIn`, vedere Application Auto Scaling API Reference.

Example

Il seguente esempio descrive una configurazione di monitoraggio degli obiettivi per una policy di dimensionamento. In questa configurazione, la metrica `RDSReaderAverageCPUUtilization` predefinita regola un cluster Aurora DB in base a un CPU utilizzo medio del 40 percento su tutte le repliche Aurora in quel cluster Aurora DB. La configurazione disabilita l'attività di riduzione per la policy di dimensionamento.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Applicazione di una policy di dimensionamento a un cluster di database Aurora

Dopo la registrazione del cluster di database Aurora con Application Auto Scaling e la definizione di una policy di dimensionamento, applicare la policy di dimensionamento al cluster di database Aurora registrato. Per applicare una politica di scalabilità a un cluster Aurora DB, puoi utilizzare o Application AWS CLI Auto Scaling. API

AWS CLI

Per applicare una politica di scalabilità al cluster Aurora DB, usa `put-scaling-policy` AWS CLI il comando con i seguenti parametri:

- `--policy-name` – Il nome della policy di dimensionamento.
- `--policy-type` – Impostare questo valore su `TargetTrackingScaling`.
- `--resource-id` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `--service-namespace` – Impostare questo valore su `rds`.
- `--scalable-dimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `--target-tracking-scaling-policy-configuration` – La configurazione di una policy di dimensionamento per il monitoraggio della destinazione da utilizzare per il cluster di database Aurora.

Example

Nell'esempio seguente si applica una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` a un cluster di database Aurora denominato `myscalablecluster` con Application Auto Scaling. Per fare ciò, usa la configurazione della policy salvata in un file denominato `config.json`.

PerLinux, omacOS: Unix

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

```
    --target-tracking-scaling-policy-configuration file://config.json
```

Per Windows:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --target-tracking-scaling-policy-configuration file://config.json
```

Application Auto Scaling API

Per applicare una policy di scalabilità al cluster Aurora DB con Application Auto Scaling, utilizza l'APIoperazione Application [PutScalingPolicy]Auto Scaling con i seguenti API parametri:

- `PolicyName` – Il nome della policy di dimensionamento.
- `ServiceNamespace` – Impostare questo valore su `rds`.
- `ResourceID` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `ScalableDimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.
- `PolicyType` – Impostare questo valore su `TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration` – La configurazione di una policy di dimensionamento per il monitoraggio della destinazione da utilizzare per il cluster di database Aurora.

Example

Nell'esempio seguente si applica una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` a un cluster di database Aurora denominato `myscalablecluster` con Application Auto Scaling. Si utilizza una policy di configurazione in base al parametro predefinito `RDSReaderAverageCPUUtilization`.

```
POST / HTTP/1.1
```

```
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
        }
    }
}
```

Modifica di una politica di scalabilità automatica per un cluster Amazon Aurora DB

È possibile modificare una politica di scalabilità utilizzando Application Auto Scaling AWS CLI, o Application Auto API Scaling. AWS Management Console

Console

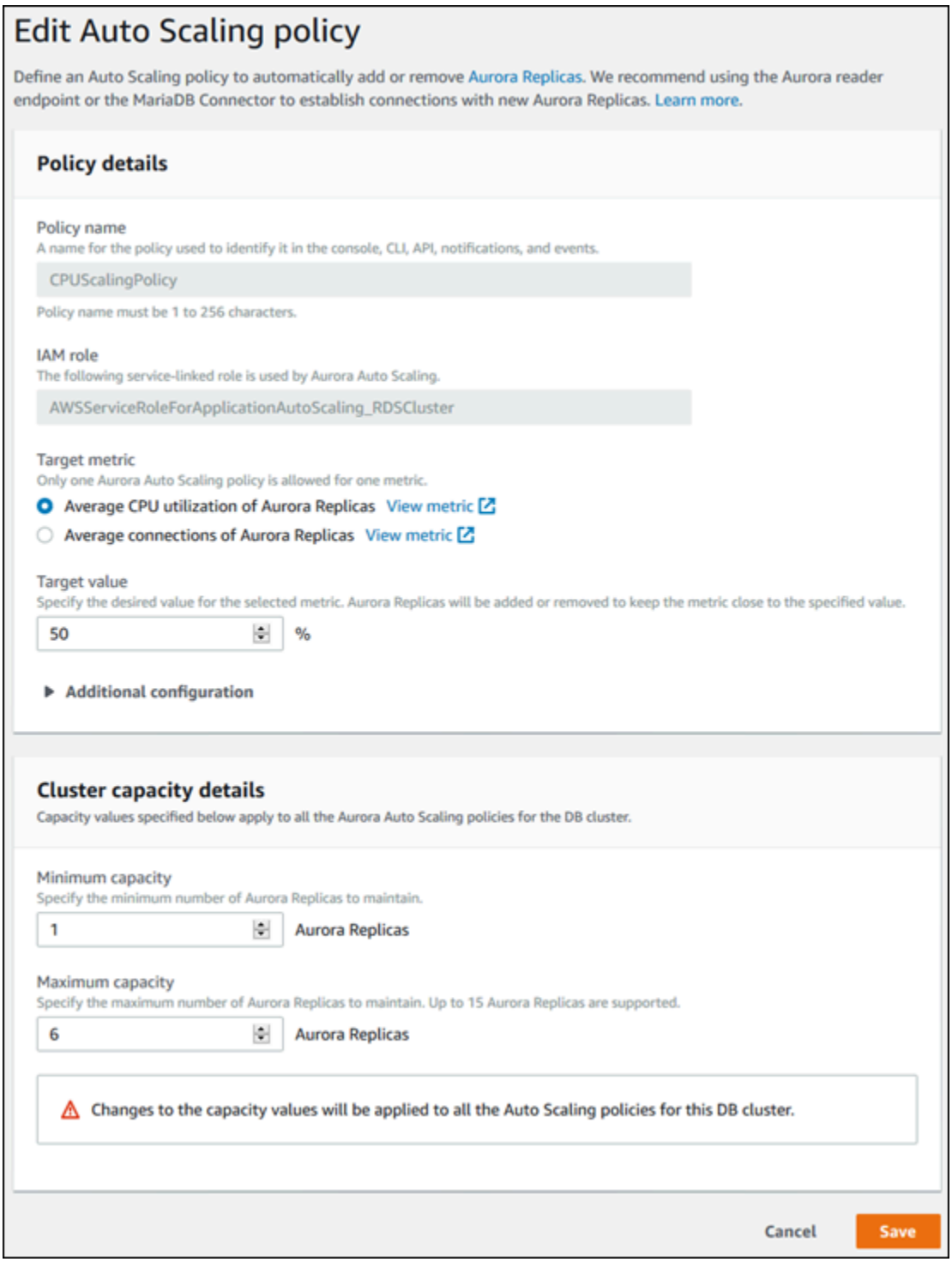
È possibile modificare una politica di ridimensionamento utilizzando. AWS Management Console

Per modificare una policy di Auto Scaling per un cluster di database Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegliere il cluster di database Aurora di cui si desidera modificare la policy di Auto Scaling.
4. Scegliere la scheda Logs & events (Log ed eventi).

5. Nella sezione Auto Scaling Policies (Policy di Auto Scaling), scegliere la policy di Auto Scaling e successivamente Edit (Modifica).
6. Apportare modifiche alla policy.
7. Scegliere Save (Salva).

Quella che segue è una finestra di dialogo di esempio Edit Scaling policy (Modifica policy di dimensionamento).



AWS CLI o Application Auto Scaling API

È possibile utilizzare AWS CLI o Application Auto Scaling API per modificare una policy di scaling nello stesso modo in cui si applica una policy di scaling:

- Quando si utilizza il AWS CLI, specificare il nome della politica che si desidera modificare nel parametro. `--policy-name` Specifica i nuovi valori per i parametri che desideri modificare.
- Quando si utilizza Application Auto ScalingAPI, specificare il nome della politica che si desidera modificare nel `PolicyName` parametro. Specifica i nuovi valori per i parametri che desideri modificare.

Per ulteriori informazioni, consulta Applicazione di una policy di dimensionamento a un cluster di database Aurora.

Eliminazione di una politica di scalabilità automatica dal cluster Amazon Aurora DB

È possibile eliminare una politica di scalabilità utilizzando il AWS Management Console AWS CLI, o Application Auto API Scaling.

Console

Puoi eliminare una policy di dimensionamento usando la AWS Management Console.

Per eliminare una policy di Auto Scaling per un cluster di database Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegliere il cluster di database Aurora con la policy di Auto Scaling che si desidera eliminare.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Nella sezione Auto Scaling Policies (Policy di Auto Scaling), scegliere la policy di Auto Scaling e successivamente Delete (Elimina).

AWS CLI

Per eliminare una politica di scalabilità dal cluster Aurora DB, usa `delete-scaling-policy` AWS CLI il comando con i seguenti parametri:

- `--policy-name` – Il nome della policy di dimensionamento.

- `--resource-id` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `--service-namespace` – Impostare questo valore su `rds`.
- `--scalable-dimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.

Example

Nell'esempio seguente, eliminare una policy di dimensionamento per il monitoraggio della destinazione denominata `myscalablepolicy` da un cluster di database Aurora denominato `myscalablecluster`.

PerLinux, omacOS: Unix

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

Per Windows:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
```

Application Auto Scaling API

Per eliminare una policy di scalabilità dal cluster Aurora DB, utilizza `DeleteScalingPolicy`l'operazione Application API Auto Scaling con i seguenti parametri:

- `PolicyName` – Il nome della policy di dimensionamento.
- `ServiceNamespace` – Impostare questo valore su `rds`.

- `ResourceID` – L'identificatore della risorsa per il cluster di database Aurora. Per questo parametro, il tipo di risorsa è `cluster` e l'identificatore univoco è il nome del cluster di database Aurora, ad esempio `cluster:myscalablecluster`.
- `ScalableDimension` – Impostare questo valore su `rds:cluster:ReadReplicaCount`.

Example

Nell'esempio seguente, si elimina una policy di scaling di tracciamento della destinazione denominata da `myscalablepolicy` un cluster Aurora DB denominato Application `myscalablecluster` Auto Scaling. API

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount"
}
```

Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora

Puoi utilizzare le seguenti opzioni per gestire le prestazioni e il dimensionamento dei cluster DB e le istanze database Aurora:

Argomenti

- Dimensionamento dello storage
- Dimensionamento delle istanze
- Dimensionamento della lettura
- Gestione delle connessioni
- Gestione dei piani di esecuzione delle query

Dimensionamento dello storage

Lo storage Aurora viene automaticamente dimensionato con i dati del volume dei cluster. Man mano che i dati aumentano, l'archiviazione del volume del cluster si espande fino a un massimo di 128 tebibyte (TiB) o 64 TiB. La dimensione massima dipende dalla versione del motore di database. Per informazioni sui tipi di dati inclusi nel volume del cluster, consulta Archiviazione Amazon Aurora. Per informazioni sulla dimensione massima per una versione specifica, consulta Limiti di dimensione Amazon Aurora.

Le dimensioni del volume del cluster vengono valutate ogni ora, per stabilire i costi di storage. Per informazioni sui prezzi, consulta la pagina dei prezzi di Aurora.

Anche se un volume del cluster Aurora può aumentare le dimensioni fino a molti tebibyte, solo lo spazio utilizzato nel volume viene addebitato. Il meccanismo per determinare lo spazio di storage fatturato dipende dalla versione del cluster Aurora.

- Quando i dati Aurora vengono rimossi dal volume del cluster, lo spazio fatturato complessivo diminuisce di una quantità comparabile. Questo comportamento di ridimensionamento dinamico si verifica quando i tablespace sottostanti vengono eliminati o riorganizzati per richiedere meno spazio. Pertanto, puoi ridurre le spese di archiviazione eliminando le tabelle e i database che non sono più necessari. Il ridimensionamento dinamico si applica a determinate versioni di Aurora. Di seguito sono riportate le versioni di Aurora in cui il volume del cluster viene ridimensionato dinamicamente durante la rimozione dei dati:

Motore del database	Versioni con ridimensionamento dinamico
Aurora Mia SQL	• Versione 3 (compatibile con My SQL 8.0): tutte le versioni supportate • Versione 2 (compatibile con My SQL 5.7): 2.11 e successive
Aurora Postger SQL	Tutte le versioni supportate
Aurora Serverless v2	Tutte le versioni supportate
Aurora Serverless v1	Non disponibile

- Nelle versioni di Aurora precedenti a quelle dell'elenco precedente, il volume del cluster può riutilizzare lo spazio liberato quando si rimuovono i dati, ma le dimensioni del volume stesso non diminuiscono mai.

Il ridimensionamento dinamico si applica alle operazioni che rimuovono o ridimensionano fisicamente i tablespace all'interno del volume del cluster. Pertanto, si applica a SQL istruzioni come`DROP TABLE`,, `DROP DATABASE` e. `TRUNCATE TABLE` `ALTER TABLE ... DROP PARTITION` Non si applica all'eliminazione di righe con l'istruzione `DELETE`. Se si elimina un numero elevato di righe da una tabella, è possibile eseguire l'SQL`OPTIMIZE TABLE`istruzione Aurora My o utilizzare l'SQL`pg_repack`estensione Aurora Postgre in seguito per riorganizzare la tabella e ridimensionare dinamicamente il volume del cluster.

Per Aurora MySQL, valgono le seguenti considerazioni:

- Dopo aver aggiornato il cluster DB a una versione del motore DB che supporti il ridimensionamento dinamico e quando la funzionalità è abilitata in quella specifica area Regione AWS, lo spazio che viene successivamente liberato da determinate SQL istruzioni, ad esempio, è recuperabile. `DROP TABLE`

  Se la funzionalità è esplicitamente disabilitata in un determinato caso, lo spazio potrebbe essere riutilizzabile Regione AWS, e non recuperabile, solo nelle versioni che supportano il ridimensionamento dinamico.

La funzionalità è stata abilitata per versioni specifiche del motore DB (1.23.0—1.23.4, 2.09.0—2.09.3 e 2.10.0) tra novembre 2020 e marzo 2022 ed è abilitata per impostazione predefinita su tutte le versioni successive.

- Una tabella viene archiviata internamente in uno o più frammenti contigui di dimensioni diverse. Durante l'esecuzione `TRUNCATE TABLE` delle operazioni, lo spazio corrispondente al primo frammento è riutilizzabile e non recuperabile. Gli altri frammenti sono recuperabili. Durante `DROP TABLE` le operazioni, lo spazio corrispondente all'intero tablespace è recuperabile.
- Il `innodb_file_per_table` parametro influisce sul modo in cui è organizzata la memorizzazione delle tabelle. Quando le tabelle fanno parte del tablespace di sistema, l'eliminazione della tabella non riduce le dimensioni del tablespace di sistema. Pertanto, assicurati di impostare su 1 `innodb_file_per_table` per i cluster Aurora My SQL DB per sfruttare appieno il ridimensionamento dinamico.
- Nella versione 2.11 e successive, il tablespace temporaneo InnoDB viene eliminato e ricreato al riavvio. In questo modo lo spazio occupato dal tablespace temporaneo viene rilasciato al sistema e il volume del cluster viene ridimensionato. Per sfruttare appieno la funzionalità di ridimensionamento dinamico, consigliamo di aggiornare il cluster DB alla versione 2.11 o successiva.

> **Note**
>
> La funzionalità di ridimensionamento dinamico non recupera spazio immediatamente quando le tabelle nelle tablespace vengono eliminate, ma gradualmente, a una velocità di circa 10 TB al giorno. Lo spazio nel tablespace di sistema non viene recuperato, perché il tablespace di sistema non viene mai rimosso. Lo spazio libero non reclamato in uno spazio tabella viene riutilizzato quando un'operazione richiede spazio in tale spazio tabella. La funzionalità di ridimensionamento dinamico può recuperare spazio di archiviazione solo quando il cluster è in uno stato disponibile.

Puoi verificare la quantità di spazio di storage utilizzata da un cluster monitorando il parametro `VolumeBytesUsed` in CloudWatch. Per ulteriori informazioni sulla fatturazione dello storage, consulta Come viene fatturato lo storage dei dati Aurora.

- In AWS Management Console, è possibile visualizzare questa figura in un grafico visualizzando la `Monitoring` scheda nella pagina dei dettagli del cluster.

- Con AWS CLI, è possibile eseguire un comando simile al seguente esempio di Linux. Sostituisci i tuoi valori per l'ora di inizio e di fine e il nome del cluster.

```
aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '6 hours ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" \
  --statistics Average Maximum Minimum \
  --dimensions Name=DBClusterIdentifier,Value=my_cluster_identifier
```

  Questo comando genera un output simile al seguente.

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T21:25:00+00:00",
            "Average": 182871982080.0,
            "Minimum": 182871982080.0,
            "Maximum": 182871982080.0,
            "Unit": "Bytes"
        }
    ]
}
```

Gli esempi seguenti mostrano come è possibile tenere traccia dell'utilizzo dello storage per un cluster Aurora nel tempo utilizzando AWS CLI i comandi su un sistema Linux. I parametri `--start-time` e `--end-time` definiscono l'intervallo di tempo complessivo come un giorno. Il parametro `--period` richiede le misurazioni a intervalli di un'ora. Non ha senso scegliere un valore `--period` piccolo perché i parametri vengono raccolti a intervalli, non in modo continuo. Inoltre, le operazioni di archiviazione Aurora a volte continuano per qualche tempo in background dopo il completamento dell'SQListruzione pertinente.

Il primo esempio restituisce l'output nel JSON formato predefinito. I punti dati vengono restituiti in ordine arbitrario, non in base al timestamp. È possibile importare questi JSON dati in uno strumento di creazione di grafici per eseguire l'ordinamento e la visualizzazione.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id
```

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T19:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T00:40:00+00:00",
            "Maximum": 198573719552.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T05:40:00+00:00",
            "Maximum": 206827454464.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-04T17:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
... output omitted ...
```

In questo esempio vengono restituiti gli stessi dati del precedente. Il parametro `--output` rappresenta i dati in formato testo normale compatto. Il comando `aws cloudwatch` invia il suo output al comando `sort`. Il `-k` parametro del `sort` comando ordina l'output in base al terzo campo, che è il timestamp in formato Universal Coordinated Time (). UTC

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id \
  --output text | sort -k 3
VolumeBytesUsed
DATAPOINTS  182872522752.0  2020-08-04T17:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T18:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T19:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T20:41:00+00:00 Bytes
DATAPOINTS  187667791872.0  2020-08-04T21:41:00+00:00 Bytes
DATAPOINTS  190981029888.0  2020-08-04T22:41:00+00:00 Bytes
DATAPOINTS  195587244032.0  2020-08-04T23:41:00+00:00 Bytes
```

```
DATAPOINTS  201048915968.0  2020-08-05T00:41:00+00:00 Bytes
DATAPOINTS  205368492032.0  2020-08-05T01:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T02:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T03:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T04:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T05:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T06:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T07:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T08:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T09:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T10:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T11:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T12:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T13:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T14:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T15:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T16:41:00+00:00 Bytes
```

L'output ordinato mostra quanto spazio di storage è stato utilizzato all'inizio e alla fine del periodo di monitoraggio. Puoi inoltre trovare i punti durante quel periodo quando Aurora ha allocato più spazio di storage per il cluster. Nell'esempio seguente vengono utilizzati i comandi Linux per riformattare i valori `VolumeBytesUsed` iniziali e finali come gigabyte (GB) e gibibyte (GiB). I gigabyte rappresentano unità misurate in potenze di 10 e sono comunemente utilizzati nelle discussioni di storage per dischi rigidi rotazionali. I gibibyte rappresentano le unità misurate in potenze di 2. Le misurazioni e i limiti di archiviazione di Aurora sono generalmente indicati nelle unità di potenza di 2, come gibibyte e tebibyte.

```
$ GiB=$((1024*1024*1024))
$ GB=$((1000*1000*1000))
$ echo "Start: $((182872522752/$GiB)) GiB, End: $((206833664000/$GiB)) GiB"
Start: 170 GiB, End: 192 GiB
$ echo "Start: $((182872522752/$GB)) GB, End: $((206833664000/$GB)) GB"
Start: 182 GB, End: 206 GB
```

Il parametro `VolumeBytesUsed` indica la quantità di spazio di storage nel cluster che sta causando costi. Quindi, è meglio ridurre al minimo questo numero quando possibile. Tuttavia, questo parametro non include alcune risorse di storage che Aurora utilizza internamente nel cluster senza addebitare alcun costo. Se il cluster si sta avvicinando al limite di storage e potrebbe esaurire lo spazio, è più utile monitorare il parametro `AuroraVolumeBytesLeftTotal` e cercare di massimizzare

tale numero. Nell'esempio seguente viene eseguito un calcolo simile a quello precedente, ma per `AuroraVolumeBytesLeftTotal` invece di `VolumeBytesUsed`.

```
$ aws cloudwatch get-metric-statistics --metric-name "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_old_cluster_id \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS      140530528288768.0       2023-02-23T19:25:00+00:00       Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((69797067915264 / $TB)) TB remaining for this cluster"
69 TB remaining for this cluster
$ echo "$((69797067915264 / $TiB)) TiB remaining for this cluster"
63 TiB remaining for this cluster
```

Per un cluster che esegue Aurora My SQL versione 2.09 o successiva o Aurora PostgreSQL, la dimensione gratuita riportata da `VolumeBytesUsed` aumenta quando i dati vengono aggiunti e diminuisce quando i dati vengono rimossi. L'esempio seguente mostra come. Questo report mostra le dimensioni massime e minime di storage per un cluster a intervalli di 15 minuti man mano che vengono create ed eliminate le tabelle con dati temporanei. Il report elenca il valore massimo prima del valore minimo. Pertanto, per capire come l'utilizzo dello storage è cambiato nell'intervallo di 15 minuti, interpreta i numeri da destra a sinistra.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Maximum Minimum --dimensions
 Name=DBClusterIdentifier,Value=my_new_cluster_id
  --output text | sort -k 4
VolumeBytesUsed
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T20:49:00+00:00 Bytes
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T21:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 14545305600.0 2020-08-05T21:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 15614263296.0 2020-08-05T23:19:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-05T23:49:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-06T00:19:00+00:00 Bytes
```

L'esempio seguente mostra come in un cluster che esegue Aurora My SQL versione 2.09 o successiva o Aurora PostgreSQL, la dimensione libera riportata da riflette il limite di dimensione di 128 TiB. `AuroraVolumeBytesLeftTotal`

```
$ aws cloudwatch get-metric-statistics --region us-east-1 --metric-name
 "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBClusterIdentifier,Value=pq-57 \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS 140515818864640.0 2020-08-05T20:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:26:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:56:00+00:00 Count
DATAPOINTS 140514866757632.0 2020-08-05T22:26:00+00:00 Count
DATAPOINTS 140511020580864.0 2020-08-05T22:56:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:26:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-06T00:26:00+00:00 Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((140515818864640 / $TB)) TB remaining for this cluster"
140 TB remaining for this cluster
$ echo "$((140515818864640 / $TiB)) TiB remaining for this cluster"
127 TiB remaining for this cluster
```

Dimensionamento delle istanze

Puoi dimensionare il cluster database Aurora in base alle necessità, modificando la classe di istanza database per ogni istanza database nel cluster database. Aurora supporta diverse classi di istanze database ottimizzate per Aurora, a seconda della compatibilità del motore di database.

Motore di database	Dimensionamento delle istanze
Amazon Aurora My SQL	Consulta la sezione Dimensionamento delle istanze database Aurora MySQL
Amazon Aurora Poster SQL	Per informazioni, consulta Dimensionamento delle istanze database Aurora PostgreSQL

Dimensionamento della lettura

Puoi ottenere il dimensionamento della lettura per il cluster di database Aurora creando fino a 15 repliche Aurora nel cluster di database. Ogni replica Aurora restituisce gli stessi dati dal volume del cluster con ritardo di replica minimo—, in genere di molto inferiore a 100 millisecondi dopo che l'istanza primaria ha scritto un aggiornamento. Man mano che il traffico di lettura aumenta, puoi creare ulteriori repliche Aurora ed effettuare direttamente una connessione alle stesse per distribuire il carico di lettura del cluster DB. Non è necessario che le repliche Aurora appartengano alla stessa classe di istanze database dell'istanza primaria.

Per informazioni sull'aggiunta di repliche di Aurora a un cluster di database, consulta Aggiunta di repliche di Aurora a un cluster di database.

Gestione delle connessioni

Il numero massimo di connessioni consentite a un'istanza database di Aurora è determinato dal parametro `max_connections` nel gruppo di parametri a livello di istanza per l'istanza database. Il valore predefinito di tale parametro varia in base alla classe di istanze database utilizzata per l'istanza database e alla compatibilità del motore di database.

Motore di database	Valore predefinito di max_connections
Amazon Aurora My SQL	Consulta la sezione Numero massimo di connessioni a un'istanz a database Aurora MySQL
Amazon Aurora Poster SQL	Consulta la sezione Numero massimo di connessioni a un'istanz a database Aurora PostgreSQL

> **Tip**
>
> Se le tue applicazioni aprono e chiudono spesso connessioni o mantengono aperte un gran numero di connessioni di lunga durata, ti consigliamo di utilizzare Amazon RDS Proxy. RDS Proxy è un proxy di database completamente gestito e ad alta disponibilità che utilizza il pool di connessioni per condividere le connessioni al database in modo sicuro ed efficiente. Per ulteriori informazioni su RDS Proxy, consulta. Proxy Amazon RDS per Aurora

Gestione dei piani di esecuzione delle query

Se si utilizza la gestione dei piani di query per Aurora PostgreSQL, si ottiene il controllo sui piani eseguiti dall'ottimizzatore. Per ulteriori informazioni, consulta Gestione dei piani di esecuzione delle query per Aurora PostgreSQL.

Clonazione di un volume per un cluster di database Amazon Aurora

La clonazione di Aurora ti consente di creare un nuovo cluster che abbia inizialmente le stesse pagine di dati dell'originale, ma sia un volume separato e indipendente. Il processo è progettato per essere veloce e conveniente. Il nuovo cluster con il relativo volume di dati associato è noto come clone. La creazione di un clone è più veloce ed efficiente in termini di spazio rispetto alla copia fisica dei dati utilizzando una tecnica diversa, ad esempio con il ripristino di uno snapshot.

Argomenti

- Panoramica sulla clonazione Aurora
- Limitazioni della clonazione Aurora
- Come funziona la clonazione Aurora
- Creazione di un clone Amazon Aurora
- VPCClonazione incrociata con Amazon Aurora
- Clonazione tra account con AWS RAM e Amazon Aurora

Panoramica sulla clonazione Aurora

Aurora utilizza un copy-on-write protocollo per creare un clone. Questo meccanismo utilizza uno spazio aggiuntivo minimo per creare un clone iniziale. Quando il clone viene creato per la prima volta, Aurora conserva una singola copia dei dati utilizzati dal cluster database Aurora di origine e dal nuovo cluster database Aurora (clonato). L'archiviazione aggiuntiva viene allocata solo quando vengono apportate modifiche ai dati (sul volume di archiviazione Aurora) dal cluster database Aurora di origine o dal clone del cluster database Aurora. Per ulteriori informazioni sul copy-on-write protocollo, consulta. Come funziona la clonazione Aurora

La clonazione Aurora è particolarmente utile per configurare rapidamente ambienti di test utilizzando i dati di produzione, senza rischiare il danneggiamento dei dati. È possibile utilizzare i cloni per molti tipi di applicazioni di breve durata, ad esempio:

- Sperimenta potenziali cambiamenti (modifiche allo schema e modifiche ai gruppi di parametri, ad esempio) per valutare tutti gli impatti.
- Esegui operazioni che utilizzano in modo intensivo i carichi di lavoro, come l'esportazione di dati o l'esecuzione di query analitiche sul clone.
- Creare una copia del cluster database di produzione per lo sviluppo, il test o altri scopi.

È possibile creare più di un clone dallo stesso cluster database Aurora. È anche possibile creare più cloni da un altro clone.

Dopo aver creato un clone Aurora, è possibile configurare le istanze database Aurora in modo diverso dal cluster database Aurora di origine. Ad esempio, potrebbe non essere necessario un clone per scopi di sviluppo per soddisfare gli stessi requisiti di alta disponibilità del cluster database Aurora di produzione di origine. In questo caso, è possibile configurare il clone con una singola istanza database Aurora anziché con più istanze database utilizzate dal cluster database Aurora.

Quando si crea un clone utilizzando una configurazione di distribuzione diversa da quella di origine, il clone viene creato utilizzando l'ultima versione secondaria del motore Aurora DB della sorgente.

Quando crei cloni dai tuoi cluster Aurora DB, i cloni vengono creati nel AWS tuo account, lo stesso account che possiede il cluster Aurora DB di origine. Tuttavia, puoi anche condividere Aurora Serverless v2 e ha fornito cluster e cloni Aurora DB con altri account. AWS Per ulteriori informazioni, consulta Clonazione tra account con AWS RAM e Amazon Aurora.

Una volta terminato di utilizzare il clone per test, sviluppo o altri scopi, è possibile eliminarlo.

Limitazioni della clonazione Aurora

La clonazione Aurora presenta le seguenti limitazioni:

- Puoi creare tutti i cloni che desideri, fino al numero massimo di cluster database consentito nella Regione AWS.
- È possibile creare fino a 15 cloni con protocollo. copy-on-write Dopo che hai raggiunto 15 cloni, il clone successivo che crei è una copia completa. Il protocollo Full-Copy funziona come un point-in-time ripristino.
- Non è possibile creare un clone in una AWS regione diversa dal cluster Aurora DB di origine.
- Non è possibile creare un clone da un cluster database Aurora senza la funzionalità di query parallela a un cluster che utilizza query parallela. Per inserire dati in un cluster che usa la query parallela, crea uno snapshot del cluster originale ed eseguine il ripristino nel cluster in cui è abilitata l'opzione per la query parallela.
- Non è possibile creare un clone da un cluster database Aurora che non ha istanze database. È possibile clonare solo cluster database Aurora con almeno un'istanza database.
- È possibile creare un clone in un cloud privato virtuale (VPC) diverso da quello del cluster database Aurora. In tal caso, le sottoreti del sistema VPCs devono essere mappate alle stesse zone di disponibilità.

- È possibile creare un clone con provisioning Aurora da un cluster database Aurora con provisioning.
- Cluster con Aurora Serverless v2 le istanze seguono le stesse regole dei cluster con provisioning.
- In Aurora Serverless v1:
  - È possibile creare un clone di cui è stato effettuato il provisioning da un Aurora Serverless v1 Cluster DB.
  - È possibile creare un Aurora Serverless v1 clone da un Aurora Serverless v1 o un cluster DB predisposto.
  - Non è possibile creare un Aurora Serverless v1 clone da un cluster Aurora DB non crittografato e fornito.
  - La clonazione tra account attualmente non supporta la clonazione Aurora Serverless v1 Cluster DB. Per ulteriori informazioni, consulta Restrizioni della clonazione tra più account.
  - Un clonato Aurora Serverless v1 Il cluster DB ha lo stesso comportamento e le stesse limitazioni di qualsiasi altro Aurora Serverless v1 Cluster DB. Per ulteriori informazioni, consulta Usare Amazon Aurora Serverless v1.
  - Aurora Serverless v1 I cluster database sono sempre crittografati. Quando cloni un Aurora Serverless v1 Cluster DB in un cluster Aurora DB di cui è stato eseguito il provisioning, il cluster Aurora DB fornito è crittografato. Puoi scegliere la chiave di crittografia, ma non è possibile disabilitare la crittografia. Per clonare da un cluster Aurora DB fornito a un Aurora Serverless v1, è necessario iniziare con un cluster Aurora DB con provisioning crittografato.

Come funziona la clonazione Aurora

La clonazione Aurora funziona a livello di archiviazione di un cluster database Aurora. Utilizza un copy-on-writeprotocollo veloce ed efficiente in termini di supporti durevoli sottostanti che supportano il volume di archiviazione Aurora. Per ulteriori informazioni, consulta la sezione relativa ai volumi cluster Aurora in Panoramica dell'archiviazione di Amazon Aurora.

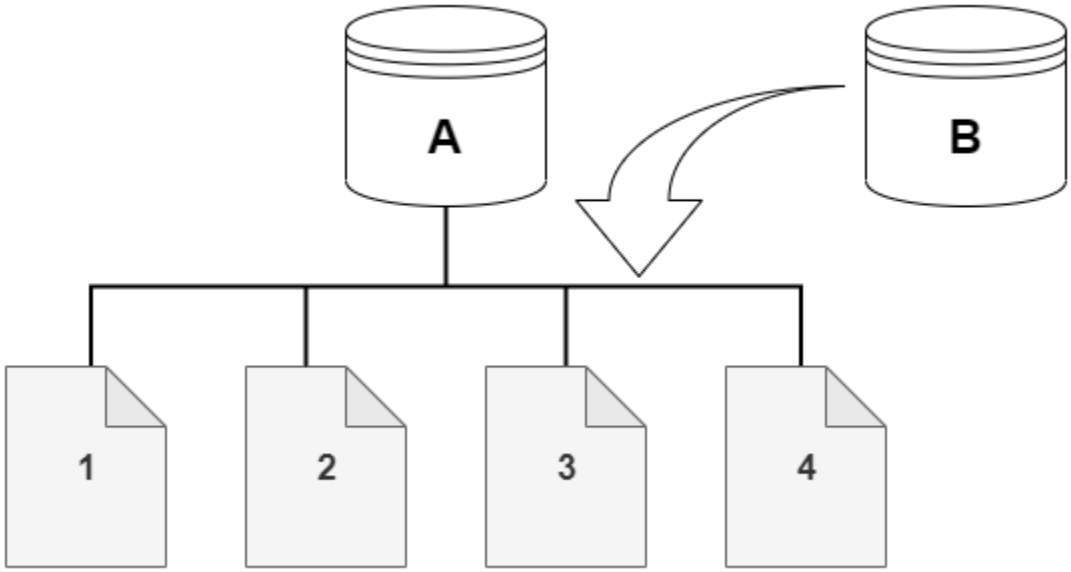
Argomenti

- Comprensione del protocollo copy-on-write
- Eliminazione di un volume cluster di origine

Comprensione del protocollo copy-on-write

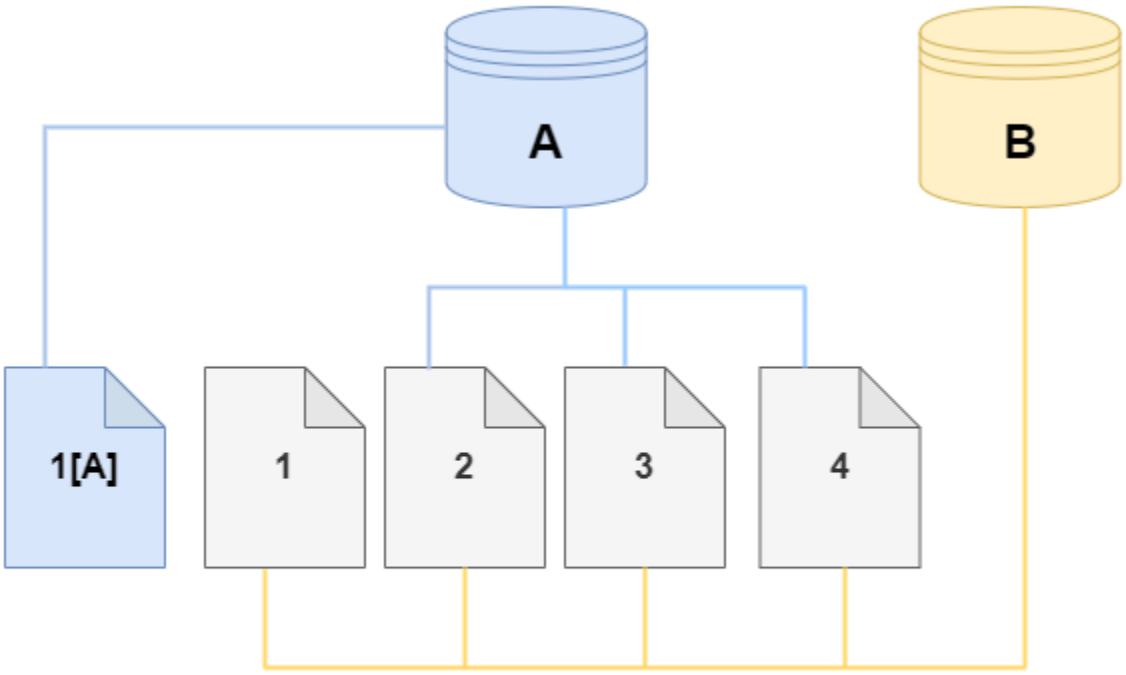
Un cluster database Aurora memorizza i dati nelle pagine del volume di archiviazione Aurora sottostante.

Ad esempio, nel diagramma seguente puoi vedere un cluster database Aurora (A) con quattro pagine dati, 1, 2, 3 e 4. Immagina che un clone, B, venga creato dal cluster database Aurora. Quando viene creato il clone, non viene copiato alcun dato. Piuttosto, il clone punta allo stesso insieme di pagine del cluster database Aurora di origine.

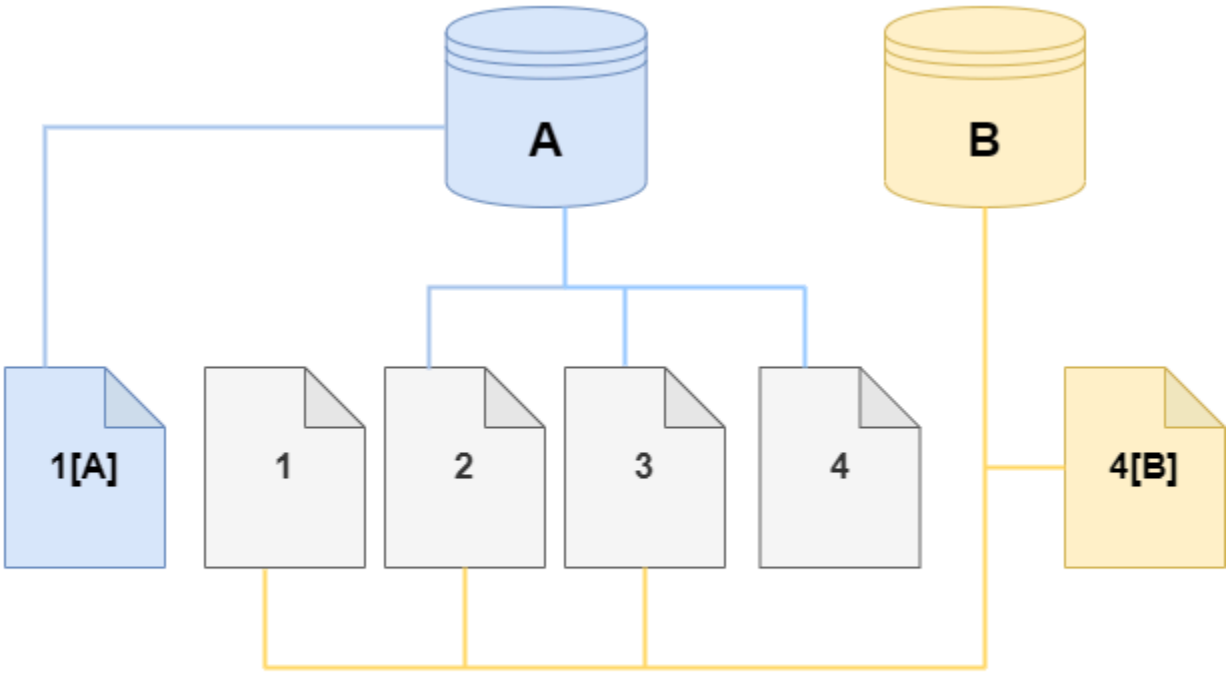


Quando viene creato il clone, in genere non è necessario alcuno spazio di archiviazione aggiuntivo. Il copy-on-write protocollo utilizza lo stesso segmento sul supporto di archiviazione fisico del segmento di origine. Lo spazio di archiviazione aggiuntivo è necessario solo se la capacità del segmento di origine non è sufficiente per l'intero segmento di clone. In questo caso, il segmento di origine viene copiato su un altro dispositivo fisico.

Nei diagrammi seguenti, è possibile trovare un esempio del copy-on-write protocollo in azione che utilizza lo stesso cluster A e il suo clone, B, come mostrato in precedenza. Supponiamo che si apporti una modifica al cluster database Aurora (A) che si traduce in una modifica ai dati contenuti a pagina 1. Invece di scrivere sulla pagina originale 1, Aurora crea una nuova pagina 1[A]. Il volume del cluster database Aurora per cluster (A) punta ora alla pagina 1[A], 2, 3 e 4, mentre il clone (B) fa ancora riferimento alle pagine originali.



Sul clone, viene apportata una modifica a pagina 4 sul volume di archiviazione. Invece di scrivere sulla pagina originale 4, Aurora crea una nuova pagina 4[B]. Il clone punta ora alle pagine 1, 2, 3 e alla pagina 4[B], mentre il cluster (A) continua a puntare a 1[A], 2, 3 e 4.



Quando nel corso del tempo si verificano altre modifiche sia nel volume del cluster database Aurora di origine che nel clone, avrai bisogno di più spazio di archiviazione per acquisire e archiviare tali modifiche.

Eliminazione di un volume cluster di origine

Inizialmente, il volume clone condivide le stesse pagine di dati del volume originale da cui viene creato il clone. Finché esiste il volume originale, il volume clone viene considerato solo il proprietario delle pagine che il clone ha creato o modificato. Pertanto, la `VolumeBytesUsed` metrica per il volume clone all'inizio è piccola e cresce solo quando i dati divergono tra il cluster originale e il clone. Per le pagine identiche tra il volume di origine e il clone, i costi di archiviazione si applicano solo al cluster originale. Per ulteriori informazioni sul parametro `VolumeBytesUsed`, consulta Parametri a livello di cluster per Amazon Aurora.

Quando si elimina un volume del cluster di origine a cui sono associati uno o più cloni, i dati nei volumi del cluster dei cloni non vengono modificati. Aurora conserva le pagine che in precedenza erano di proprietà del volume del cluster di origine. Aurora ridistribuisce la fatturazione dello spazio di

archiviazione per le pagine di proprietà del cluster eliminato. Ad esempio, supponiamo che un cluster originale avesse due cloni e che quindi il cluster originale sia stato eliminato. La metà delle pagine di dati di proprietà del cluster originale ora sarebbe di proprietà di un clone. L'altra metà delle pagine sarebbe di proprietà dell'altro clone.

Se elimini il cluster originale, man mano che crei o elimini altri cloni, Aurora continua a ridistribuire la proprietà delle pagine di dati tra tutti i cloni che condividono le stesse pagine. Pertanto, è possibile che il valore della `VolumeBytesUsed` metrica cambi per il volume del cluster di un clone. Il valore della metrica può diminuire man mano che vengono creati più cloni e la proprietà della pagina viene distribuita su più cluster. Il valore della metrica può inoltre aumentare man mano che i cloni vengono eliminati e la proprietà della pagina viene assegnata a un numero inferiore di cluster. Per informazioni su come le operazioni di scrittura influiscono sulle pagine di dati sui volumi clonati, consulta. Comprensione del protocollo copy-on-write

Quando il cluster originale e i cloni sono di proprietà dello stesso AWS account, tutti i costi di archiviazione per tali cluster si applicano allo stesso account. AWS Se alcuni cluster sono cloni con più account, l'eliminazione del cluster originale può comportare costi di archiviazione aggiuntivi per gli account che possiedono i cloni su più AWS account.

Ad esempio, supponiamo che un volume del cluster contenga 1000 pagine di dati utilizzate prima di creare qualsiasi clone. Quando clonate quel cluster, inizialmente il volume clone ha zero pagine utilizzate. Se il clone apporta modifiche a 100 pagine di dati, solo quelle 100 pagine vengono memorizzate nel volume clone e contrassegnate come utilizzate. Le altre 900 pagine invariate del volume principale sono condivise da entrambi i cluster. In questo caso, il cluster principale prevede costi di archiviazione per 1000 pagine e il volume clone per 100 pagine.

Se si elimina il volume di origine, i costi di archiviazione per il clone includono le 100 pagine modificate, più le 900 pagine condivise del volume originale, per un totale di 1000 pagine.

Creazione di un clone Amazon Aurora

È possibile creare un clone nello stesso AWS account del cluster Aurora DB di origine. A tale scopo, è possibile utilizzare AWS Management Console o AWS CLI le procedure seguenti.

Per consentire a un altro AWS account di creare un clone o di condividere un clone con un altro AWS account, utilizzare le procedure riportate in. Clonazione tra account con AWS RAM e Amazon Aurora

Console

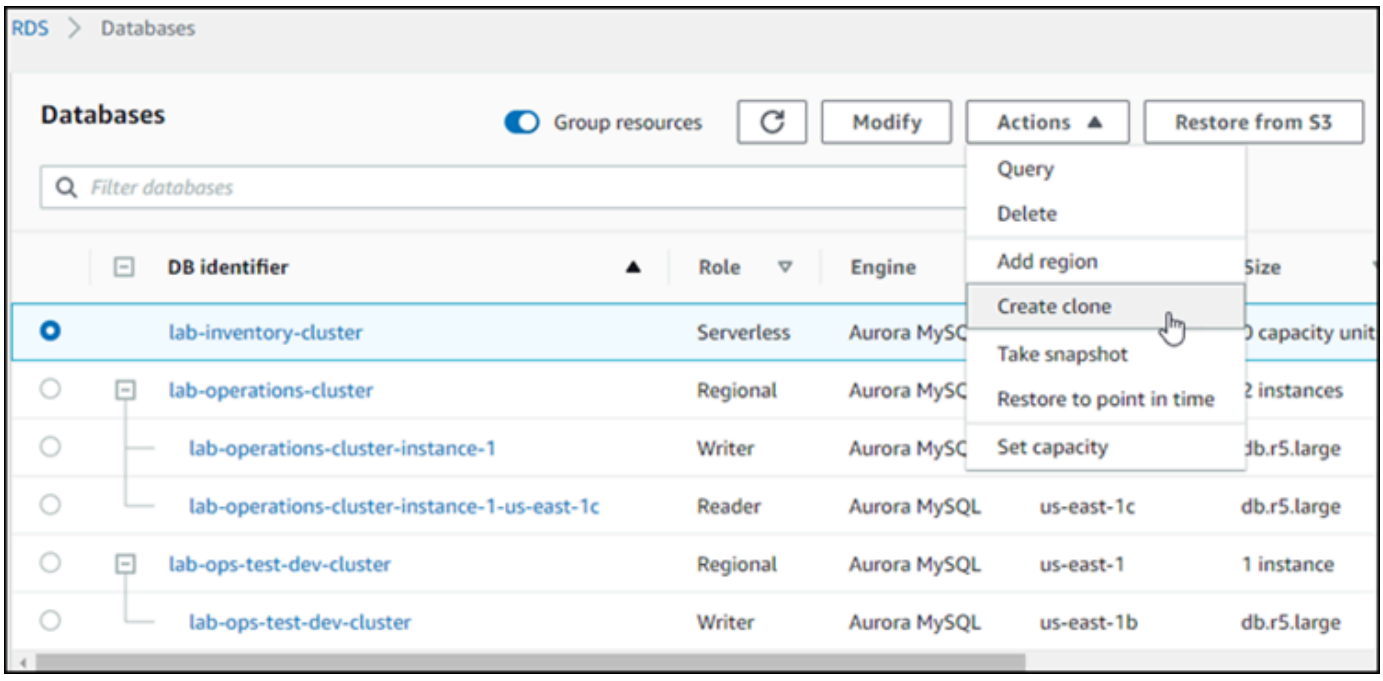
La procedura seguente descrive come clonare un cluster database Aurora utilizzando la AWS Management Console.

Creazione di un clone utilizzando i AWS Management Console risultati in un cluster Aurora DB con un'istanza Aurora DB.

Queste istruzioni si applicano ai cluster DB di proprietà dello stesso AWS account che sta creando il clone. Se il cluster DB è di proprietà di un AWS account diverso, vedi Clonazione tra account con AWS RAM e Amazon Aurora invece.

Per creare un clone di un cluster DB di proprietà del tuo AWS account utilizzando il AWS Management Console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Seleziona il cluster database Aurora dall'elenco, quindi in Operazioni, seleziona Crea clone.



   Verrà visualizzata la pagina Crea clone, dove è possibile configurare le opzioni Impostazioni, Connettività e altre opzioni per il clone del cluster database Aurora.
4. Per Identificatore istanza database, immettere il nome che si desidera assegnare al cluster database Aurora clonato.

5. In Aurora Serverless v1 Cluster DB, scegli Provisioned o Serverless per il tipo di capacità.

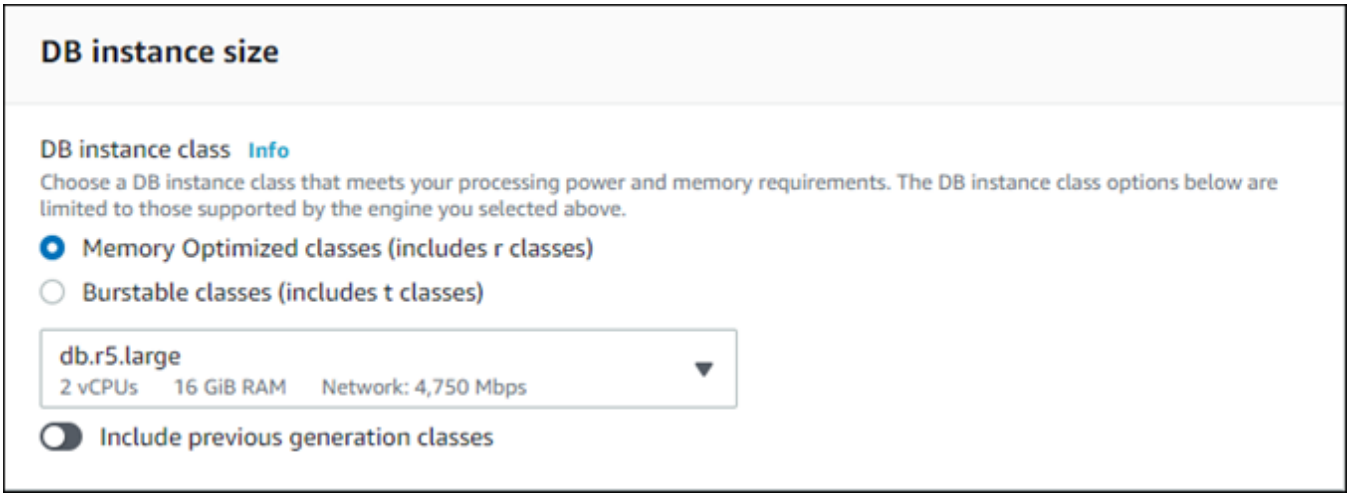
   Puoi scegliere Serverless solo se il cluster Aurora DB di origine è un Aurora Serverless v1 DB cluster o è un cluster Aurora DB fornito e crittografato.

6. In Aurora Serverless v2 o cluster DB con provisioning, scegli uno dei due Aurora I/O-Optimized o Aurora Standardper la configurazione dello storage del cluster.

   Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

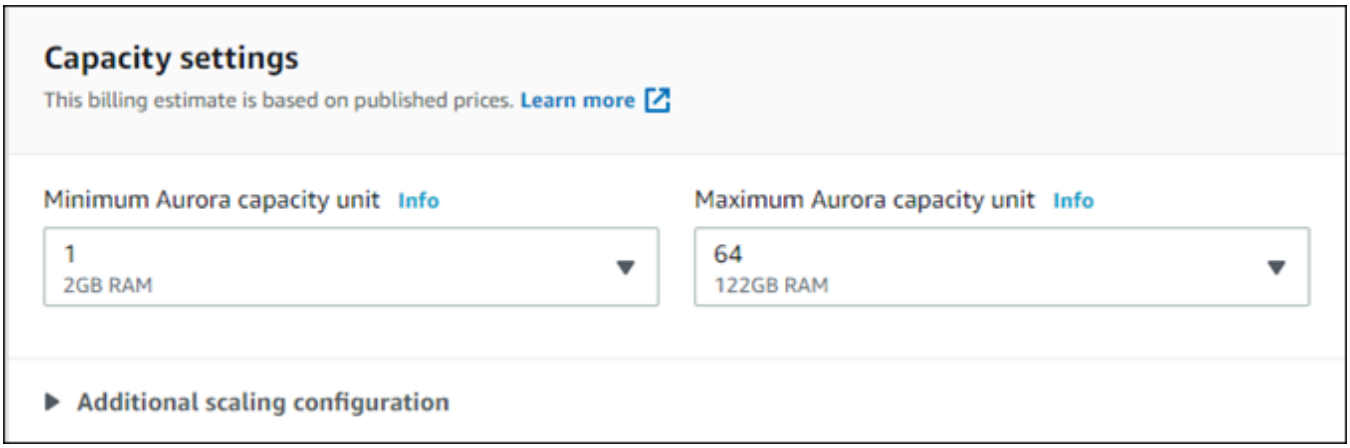
7. Scegliere la dimensione dell'istanza database o la capacità del cluster database:

   - Per un clone predisposto, scegli una classe di istanza DB.



     È possibile accettare l'impostazione predefinita oppure utilizzare una classe di istanza database diversa per il clone.

   - Per un Aurora Serverless v1 oppure Aurora Serverless v2 clone, scegli le impostazioni di capacità.



Puoi accettare le impostazioni fornite o modificarle per il tuo clone.

8. Scegliete le altre impostazioni necessarie per il clone. Per ulteriori informazioni sulle impostazioni di cluster e istanza database Aurora, consulta Creazione di un cluster database Amazon Aurora.
9. Scegli Crea clone.

Quando viene creato il clone, viene elencato con gli altri cluster database Aurora nella sezione Database e se ne visualizza lo stato corrente. Il clone è pronto per l'utilizzo quando lo stato diventa Disponibile.

AWS CLI

L'utilizzo di AWS CLI per la clonazione del cluster Aurora DB prevede passaggi separati per la creazione del cluster clone e l'aggiunta di una o più istanze DB ad esso.

Il `restore-db-cluster-to-point-in-time` AWS CLI comando utilizzato genera un cluster Aurora DB con gli stessi dati di archiviazione del cluster originale, ma nessuna istanza Aurora DB. Le istanze DB vengono create separatamente dopo che il clone è disponibile. È possibile scegliere il numero di istanze DB e le relative classi di istanze per assegnare al clone una capacità di elaborazione maggiore o minore rispetto al cluster originale. Le fasi del processo sono le seguenti:

1. Crea il clone utilizzando il comando restore-db-cluster-to- point-in-time CLI.
2. Crea l'istanza Writer DB per il clone utilizzando il comando create-db-instanceCLI.
3. (Facoltativo) Esegui comandi create-db-instanceCLI aggiuntivi per aggiungere una o più istanze di lettura al cluster clone. L'utilizzo delle istanze Reader aiuta a migliorare gli aspetti relativi all'elevata disponibilità e alla scalabilità di lettura del clone. Potresti saltare questo passaggio se intendi utilizzare il clone solo per lo sviluppo e il test.

Argomenti

- Creazione del clone
- Controllo dello stato e ottenimento deii dettagli del clone
- Creazione dell'istanza database Aurora per il clone
- Parametri da utilizzare per la clonazione

Creazione del clone

Utilizzate il comando `restore-db-cluster-to-point-in-time` CLI per creare il cluster clone iniziale.

Per creare un clone da un cluster Aurora DB di origine

- Utilizza il comando della CLI `restore-db-cluster-to-point-in-time`. Specificate i valori per i seguenti parametri. In questo caso tipico, il clone utilizza la stessa modalità motore del cluster originale, fornito o Aurora Serverless v1.
  - `--db-cluster-identifier`: scegliere un nome significativo per il clone. Assegnate un nome al clone quando utilizzate il comando restore-db-cluster-to- point-in-time CLI. Quindi si passa il nome del clone nel comando create-db-instanceCLI.
  - `--restore-type`: utilizza `copy-on-write` per creare un clone del cluster database di origine. Senza questo parametro, il parametro `restore-db-cluster-to-point-in-time` ripristina il cluster database Aurora anziché creare un clone.
  - `--source-db-cluster-identifier`: utilizza il nome del cluster database Aurora di origine che si desidera clonare.
  - `--use-latest-restorable-time`— Questo valore indica i dati di volume ripristinabili più recenti per il cluster DB di origine. Utilizzatelo per creare cloni.

Nell'esempio seguente viene creato un clone denominato `my-clone` da un cluster denominato `my-source-cluster`.

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

In Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
```

```
    --use-latest-restorable-time
```

Il comando restituisce l'oggetto JSON contenente i dettagli del clone. Prima di provare a creare l'istanza database per il clone, verificare che il cluster database clonato sia disponibile. Per ulteriori informazioni, consulta Controllo dello stato e ottenimento deii dettagli del clone.

Si supponga, ad esempio, di disporre di un cluster denominato `tpch100g` che si desidera clonare. Il seguente esempio di Linux crea un cluster clonato denominato`tpch100g-clone`, an Aurora Serverless v2 istanza writer denominata `tpch100g-clone-instance` e un'istanza reader predisposta denominata `tpch100g-clone-instance-2` per il nuovo cluster.

Non è necessario fornire alcun parametro, come `--master-username` e `--master-user-password`. Aurora determina automaticamente quelli dal cluster originale. È necessario specificare il motore database da utilizzare. Pertanto, l'esempio verifica il nuovo cluster per determinare il valore corretto da utilizzare per il parametro `--engine`.

Questo esempio include anche l'`--serverless-v2-scaling-configuration`opzione per la creazione del cluster clone. In questo modo, puoi aggiungere Aurora Serverless v2 istanze al clone anche se il cluster originale non lo utilizzava Aurora Serverless v2.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier tpch100g \
  --db-cluster-identifier tpch100g-clone \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
  --restore-type copy-on-write \
  --use-latest-restorable-time

$ aws rds describe-db-clusters \
  --db-cluster-identifier tpch100g-clone \
    --query '*[].[Engine]' \
    --output text
aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.serverless \
  --engine aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance-2 \
```

```
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.r6g.2xlarge \
  --engine aurora-mysql
```

Per creare un clone con una modalità motore diversa dal cluster Aurora DB di origine

- Questa procedura si applica solo alle versioni precedenti del motore che supportano Aurora Serverless v1. Supponiamo di avere un Aurora Serverless v1 cluster e desideri creare un clone che sia un cluster fornito. In tal caso, utilizzate il comando `restore-db-cluster-to-point-in-time` CLI e specificate valori di parametro simili a quelli dell'esempio precedente, più questi parametri aggiuntivi:

  - `--engine-mode`— Utilizzate questo parametro solo per creare cloni con una modalità motore diversa dal cluster Aurora DB di origine. Questo parametro si applica solo alle versioni precedenti del motore che supportano Aurora Serverless v1. Scegliete il valore da utilizzare `--engine-mode` come segue:
    - `--engine-mode provisioned`Da utilizzare per creare un clone del cluster Aurora DB con provisioning da un Aurora Serverless Cluster DB.

      > **Note**
      >
      > Se intendi utilizzare Aurora Serverless v2 con un cluster che è stato clonato da Aurora Serverless v1, si specifica comunque la modalità motore per il clone come. `provisioned` Successivamente si eseguono ulteriori passaggi di aggiornamento e migrazione.

    - Utilizzare `--engine-mode serverless` per creare un Aurora Serverless v1 clone da un cluster Aurora DB fornito. Quando si specifica la modalità `serverless` del motore, è anche possibile scegliere. `--scaling-configuration`
  - `--scaling-configuration`— (Facoltativo) Utilizzare with `--engine-mode serverless` per configurare la capacità minima e massima per un Aurora Serverless v1 clonare. Se non si utilizza questo parametro, Aurora crea un Aurora Serverless v1 clone utilizzando l'impostazione predefinita Aurora Serverless v1 valori di capacità per il motore DB.

L'esempio seguente crea un clone fornito denominato`my-clone`, da un Aurora Serverless v1 Cluster DB denominato. `my-source-cluster`

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --engine-mode provisioned \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

In Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --engine-mode provisioned ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Questi comandi restituiscono l'oggetto JSON contenente i dettagli del clone necessari per creare l'istanza database. Non puoi farlo finché lo stato del clone (il cluster database Aurora vuoto) non diventa Disponibile.

> **Note**
>
> Il comando restore-db-cluster-to- point-in-time AWS CLI ripristina solo il cluster DB, non le istanze DB per quel cluster DB. Si esegue il create-db-instancecomando per creare istanze DB per il cluster DB ripristinato. Con questo comando, si specifica l'identificatore del cluster DB ripristinato come parametro. `--db-cluster-identifier` Puoi creare le istanze database solo dopo che il comando `restore-db-cluster-to-point-in-time` è terminato e il cluster database è disponibile.
> Supponiamo di iniziare con un Aurora Serverless v1 cluster e intendete migrarlo verso un Aurora Serverless v2 ammasso. Si crea un clone predisposto di Aurora Serverless v1 cluster come fase iniziale della migrazione. Per la procedura completa, compresi gli eventuali aggiornamenti di versione richiesti, vedereAggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2.

Controllo dello stato e ottenimento deii dettagli del clone

È possibile utilizzare il comando seguente per verificare lo stato del cluster clone appena creato.

```
$ aws rds describe-db-clusters --db-cluster-identifier my-clone --query '*[].[Status]'
 --output text
```

Oppure puoi ottenere lo stato e gli altri valori necessari per creare l'istanza DB per il tuo clone utilizzando la seguente AWS CLI query.

In Linux, macOS, oppure Unix:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}'
```

In Windows:

```
aws rds describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}"
```

Questa query restituisce un output simile al seguente:

```
[
  {
        "Status": "available",
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.04.1",
        "EngineMode": "provisioned"
    }
]
```

Creazione dell'istanza database Aurora per il clone

Usa il comando create-db-instanceCLI per creare l'istanza DB per il tuo Aurora Serverless v2 o clone predisposto. Non si crea un'istanza DB per un Aurora Serverless v1 clonare.

L'istanza DB eredita le `--master-user-password` proprietà `--master-username` and dal cluster DB di origine.

L'esempio seguente crea un'istanza DB per un clone di cui è stato eseguito il provisioning.

In Linux, macOS, oppure Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.r6g.2xlarge \
    --engine aurora-mysql
```

In Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.r6g.2xlarge ^
    --engine aurora-mysql
```

L'esempio seguente crea un Aurora Serverless v2 Istanza DB, per un clone che utilizza una versione del motore che supporta Aurora Serverless v2.

In Linux, macOS, oppure Unix:

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.serverless \
  --engine aurora-postgresql
```

In Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.serverless ^
    --engine aurora-mysql
```

Parametri da utilizzare per la clonazione

La tabella seguente riepiloga i vari parametri utilizzati con `restore-db-cluster-to-point-in-time` per clonare i cluster database Aurora.

Parametro	Descrizione
`--source-db-cluster-identifier`	Utilizza il nome del cluster database Aurora di origine che si desidera clonare.
`--db-cluster-identifier`	Scegli un nome significativo per il tuo clone quando lo crei con il `restore-db-cluster-to-point-in-time` comando. Quindi questo nome viene inviato al comando `create-db-instance` .
`--restore-type`	Specifica `copy-on-write` come `--restore-type` per creare un clone del cluster database di origine anziché ripristinare il cluster database Aurora di origine.
`--use-latest-restorable-time`	Questo valore indica i dati di volume ripristinabili più recenti per il cluster DB di origine. Usalo per creare cloni.
`--serverless-v2-scaling-configuration`	(Versioni più recenti che supportano Aurora Serverless v2) Utilizzare questo parametro per configurare la capacità minima e massima per un Aurora Serverless v2 clonare. Se non specifichi questo parametro, non puoi crearne uno Aurora Serverless v2 istanze nel cluster clone finché non modifichi il cluster per aggiungere questo attributo.
`--engine-mode`	(Versioni precedenti che supportano Aurora Serverless v1 solo) Utilizzate questo parametro per creare cloni di tipo diverso dal cluster Aurora DB di origine, con uno dei seguenti valori: • Utilizzato `provisioned` per creare un clone predisposto da un Aurora Serverless v1 Cluster DB. • Utilizzare `serverless` per creare un Aurora Serverless v1 clone da un sistema predisposto o Aurora Serverless v2 cluster DB. Quando si specifica la modalità `serverless` del motore, è anche possibile scegliere`--scaling-configuration` .
`--scaling-configuration`	(Versioni precedenti che supportano Aurora Serverless v1 solo) Utilizzate questo parametro per configurare la capacità minima e massima per

Parametro	Descrizione
	un Aurora Serverless v1 clonare. Se non specificate questo parametro, Aurora crea il clone utilizzando i valori di capacità predefiniti per il motore DB.

Per informazioni sulla clonazione tra VPC e più account, consulta le seguenti sezioni.

Argomenti

- VPCClonazione incrociata con Amazon Aurora
- Clonazione tra account con AWS RAM e Amazon Aurora

VPCClonazione incrociata con Amazon Aurora

Supponiamo di voler imporre diversi controlli di accesso alla rete al cluster originale e al clone. Ad esempio, è possibile utilizzare la clonazione per creare una copia di un cluster Aurora di produzione in un VPC altro a scopo di sviluppo e test. Oppure potreste creare un clone come parte di una migrazione da sottoreti pubbliche a sottoreti private, per migliorare la sicurezza del database.

Le sezioni seguenti mostrano come configurare la configurazione di rete per il clone in modo che il cluster originale e il clone possano entrambi accedere agli stessi nodi di archiviazione Aurora, anche da sottoreti diverse o diverse. VPCs La verifica preventiva delle risorse di rete può evitare errori durante la clonazione che potrebbero essere difficili da diagnosticare.

Se non conosci il modo in cui Aurora interagisce con le sottoreti e i VPCs gruppi di sottoreti DB, consulta prima. Amazon VPC e Aurora Puoi seguire i tutorial in quella sezione per creare questo tipo di risorse nella AWS console e capire come si integrano tra loro.

Poiché i passaggi prevedono il RDS passaggio da un EC2 servizio all'altro, negli esempi vengono utilizzati AWS CLI comandi che consentono di comprendere come automatizzare tali operazioni e salvare l'output.

Argomenti

- Prima di iniziare
- Raccolta di informazioni sull'ambiente di rete
- Creazione di risorse di rete per il clone

- Creazione di un clone di Aurora con nuove impostazioni di rete
- Spostamento di un cluster da sottoreti pubbliche a sottoreti private
- End-to-end esempio di creazione di un clone incrociato VPC

Prima di iniziare

Prima di iniziare a configurare un VPC cross-clone, assicuratevi di disporre delle seguenti risorse:

- Un cluster Aurora DB da utilizzare come sorgente per la clonazione. Se è la prima volta che crei un cluster Aurora DB, consulta i tutorial in Nozioni di base su Amazon Aurora per configurare un cluster utilizzando il motore di database My SQL o Postgre. SQL
- Un secondoVPC, se intendi creare un clone incrociato. VPC Se non ne hai uno VPC da usare per il clone, vedi Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 o. Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Raccolta di informazioni sull'ambiente di rete

Con la VPC clonazione incrociata, l'ambiente di rete può differire notevolmente tra il cluster originale e il relativo clone. Prima di creare il clone, raccogli e registra le informazioni sulle sottoretiVPC, sul gruppo di sottoreti DB e sulle informazioni utilizzate nel cluster originale. AZs In questo modo, è possibile ridurre al minimo le possibilità di problemi. Se si verifica un problema di rete, non sarà necessario interrompere alcuna attività di risoluzione dei problemi per cercare informazioni diagnostiche. Le sezioni seguenti mostrano CLI esempi per raccogliere questi tipi di informazioni. È possibile salvare i dettagli nel formato più comodo da consultare durante la creazione del clone e la risoluzione dei problemi.

- Fase 1: Verificare le zone di disponibilità del cluster originale
- Fase 2: Controllare il gruppo di sottoreti DB del cluster originale
- Fase 3: Controllare le sottoreti del cluster originale
- Fase 4: Verificare le zone di disponibilità delle istanze DB nel cluster originale
- Passaggio 5: verifica VPCs che puoi utilizzare per il clone

Fase 1: Verificare le zone di disponibilità del cluster originale

Prima di creare il clone, verifica quale tipo di archiviazione viene utilizzato AZs dal cluster originale. Come spiegato inArchiviazione Amazon Aurora, lo storage per ogni cluster Aurora è associato

esattamente a tre. AZs Poiché Cluster database Amazon Aurora sfrutta la separazione tra elaborazione e archiviazione, questa regola è valida indipendentemente dal numero di istanze presenti nel cluster.

Ad esempio, esegui un CLI comando come il seguente, sostituendo il nome del tuo cluster con. *my_cluster* L'esempio seguente produce un elenco ordinato alfabeticamente in base al nome AZ.

```
aws rds describe-db-clusters \
  --db-cluster-identifier my_cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' \
  --output text
```

L'esempio seguente mostra un esempio di output del comando precedente. `describe-db-clusters` Dimostra che lo storage per il cluster Aurora ne utilizza sempre tre. AZs

```
us-east-1c
us-east-1d
us-east-1e
```

Per creare un clone in un ambiente di rete che non dispone di tutte le risorse necessarie per connettersi a tali sottoretiAZs, è necessario creare sottoreti associate ad almeno due di queste e quindi creare un gruppo di sottoreti DB contenente queste AZs due o tre sottoreti. Gli esempi seguenti mostrano come.

Fase 2: Controllare il gruppo di sottoreti DB del cluster originale

Se si desidera utilizzare per il clone lo stesso numero di sottoreti del cluster originale, è possibile ottenere il numero di sottoreti dal gruppo di sottoreti DB del cluster originale. Un gruppo di sottoreti Aurora DB contiene almeno due sottoreti, ciascuna associata a una AZ diversa. Prendi nota a quali sottoreti sono AZs associate.

L'esempio seguente mostra come trovare il gruppo di sottoreti DB del cluster originale e quindi tornare al gruppo corrispondente. AZs Sostituisci il nome del cluster con quello *my_cluster* nel primo comando. Sostituisci il nome del gruppo di sottoreti DB con il secondo *my_subnet* comando.

```
aws rds describe-db-clusters --db-cluster-identifier my_cluster \
  --query '*[].DBSubnetGroup' --output text

aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
```

```
  --query '*[].Subnets[].[SubnetAvailabilityZone.Name]' --output text
```

L'output di esempio potrebbe essere simile al seguente, per un cluster con un gruppo di sottoreti DB contenente due sottoreti. In questo caso, `two-subnets` è un nome specificato durante la creazione del gruppo di sottoreti DB.

```
two-subnets

us-east-1d
us-east-1c
```

Per un cluster in cui il gruppo di sottoreti DB contiene tre sottoreti, l'output potrebbe essere simile al seguente.

```
three-subnets

us-east-1f
us-east-1d
us-east-1c
```

Fase 3: Controllare le sottoreti del cluster originale

Se hai bisogno di maggiori dettagli sulle sottoreti del cluster originale, AWS CLI esegui comandi simili ai seguenti. È possibile esaminare gli attributi della sottorete, ad esempio gli intervalli di indirizzi IP, il proprietario e così via. In questo modo, è possibile determinare se utilizzare diverse sottoreti nella stessa VPC o creare sottoreti con caratteristiche simili in un'altra. VPC

Trova la sottorete IDs di tutte le sottoreti disponibili nel tuo. VPC

```
aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc \
  --query '*[].[SubnetId]' --output text
```

Trova le sottoreti esatte utilizzate nel tuo gruppo di sottoreti DB.

```
aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetIdentifier]' --output text
```

Quindi specifica le sottoreti che desideri esaminare in un elenco, come nel comando seguente. Sostituisci i nomi delle tue sottoreti e così via. *my_subnet_1*

```
aws ec2 describe-subnets \
  --subnet-ids '["my_subnet_1","my_subnet2","my_subnet3"]'
```

L'esempio seguente mostra l'output parziale di tale comando. `describe-subnets` L'output mostra alcuni degli attributi importanti che è possibile visualizzare per ogni sottorete, ad esempio la zona Z associata e VPC quella di cui fa parte.

```
{
    'Subnets': [
        {
            'AvailabilityZone': 'us-east-1d',
            'AvailableIpAddressCount': 54,
            'CidrBlock': '10.0.0.64/26',
            'State': 'available',
            'SubnetId': 'subnet-000a0bca00e0b0000',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
        },
        {
            'AvailabilityZone': 'us-east-1c',
            'AvailableIpAddressCount': 55,
            'CidrBlock': '10.0.0.0/26',
            'State': 'available',
            'SubnetId': 'subnet-4b4dbfe4d4a4fd4c4',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
```

Fase 4: Verificare le zone di disponibilità delle istanze DB nel cluster originale

È possibile utilizzare questa procedura per comprendere le istanze DB AZs utilizzate nel cluster originale. In questo modo, puoi configurare esattamente lo stesso AZs per le istanze DB nel clone. È inoltre possibile utilizzare più o meno istanze DB nel clone a seconda che il clone venga utilizzato per la produzione, lo sviluppo e il test e così via.

Per ogni istanza del cluster originale, esegui un comando come il seguente. Assicurati che l'istanza abbia terminato la creazione e che si trovi prima nello `Available` stato in cui si trova. Sostituite l'identificatore di istanza con. *`my_instance`*

```
aws rds describe-db-instances --db-instance-identifier my_instance \
  --query '*[].AvailabilityZone' --output text
```

L'esempio seguente mostra l'output dell'esecuzione del comando precedente`describe-db-instances`. Il cluster Aurora dispone di quattro istanze di database. Pertanto, eseguiamo il comando quattro volte, sostituendo ogni volta un identificatore di istanza DB diverso. L'output mostra come tali istanze DB sono distribuite su un massimo di tre. AZs

```
us-east-1a
us-east-1c
us-east-1d
us-east-1a
```

Dopo aver creato il clone e avervi aggiunto delle istanze DB, potete specificare gli stessi nomi AZ nei comandi. `create-db-instance` È possibile farlo per configurare le istanze DB nel nuovo cluster configurato esattamente AZs come nel cluster originale.

Passaggio 5: verifica VPCs che puoi utilizzare per il clone

Se intendi creare il clone in un modo VPC diverso dall'originale, puoi ottenere un elenco di quelli VPC IDs disponibili per il tuo account. È possibile eseguire questo passaggio anche se è necessario creare eventuali sottoreti aggiuntive nello VPC stesso cluster originale. Quando si esegue il comando per creare una sottorete, si specifica l'VPCID come parametro.

Per elencare tutti i dati VPCs relativi al tuo account, esegui il CLI comando seguente:

```
aws ec2 describe-vpcs --query '*[].[VpcId]' --output text
```

L'esempio seguente mostra un esempio di output del `describe-vpcs` comando precedente. L'output dimostra che ce ne sono quattro VPCs nell' AWS account corrente che possono essere usati come origine o destinazione per la clonazione incrociata. VPC

```
vpc-fd111111
vpc-2222e2cd2a222f22e
vpc-33333333a33333d33
vpc-4ae4d4de4a4444dad
```

È possibile utilizzare la VPC stessa destinazione del clone o un'altra. VPC Se il cluster originale e il clone si trovano nello stesso gruppoVPC, è possibile riutilizzare lo stesso gruppo di sottoreti DB per il clone. È inoltre possibile creare un gruppo di sottoreti DB diverso. Ad esempio, il nuovo gruppo di sottoreti DB potrebbe utilizzare sottoreti private, mentre il gruppo di sottoreti DB del cluster originale potrebbe utilizzare sottoreti pubbliche. Se crei il clone in un altro clusterVPC, assicurati che ci siano

abbastanza sottoreti nel nuovo cluster VPC e che le sottoreti siano associate al lato destro del cluster originale. AZs

Creazione di risorse di rete per il clone

Se durante la raccolta delle informazioni di rete avete scoperto che per il clone sono necessarie risorse di rete aggiuntive, potete creare tali risorse prima di provare a configurare il clone. Ad esempio, potrebbe essere necessario creare più sottoreti, sottoreti associate a sottoreti specifiche o un nuovo gruppo di sottoreti AZs DB.

- Fase 1: Creare le sottoreti per il clone
- Fase 2: Creare il gruppo di sottoreti DB per il clone

Fase 1: Creare le sottoreti per il clone

Se devi creare nuove sottoreti per il clone, esegui un comando simile al seguente. Potrebbe essere necessario eseguire questa operazione quando si crea il clone in un altro ambiente o quando si apportano altre modifiche alla reteVPC, ad esempio utilizzando sottoreti private anziché sottoreti pubbliche.

AWS genera automaticamente l'ID della sottorete. Sostituisci il nome del clone con. VPC *my_vpc* Scegliete l'intervallo di indirizzi per consentire almeno 16 indirizzi IP nell'intervallo. `--cidr-block` È possibile includere qualsiasi altra proprietà che si desidera specificare. Esegui il comando `aws ec2 create-subnet help` per visualizzare tutte le scelte.

```
aws ec2 create-subnet --vpc-id my_vpc \
  --availability-zone AZ_name --cidr-block IP_range
```

L'esempio seguente mostra alcuni attributi importanti di una sottorete appena creata.

```
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1b',
        'AvailableIpAddressCount': 59,
        'CidrBlock': '10.0.0.64/26',
        'State': 'available',
        'SubnetId': 'subnet-44b4a44f4e44db444',
        'VpcId': 'vpc-555fc5df555e555dc',
        ...
        }
```

```
}
```

Fase 2: Creare il gruppo di sottoreti DB per il clone

Se state creando il clone in un set di sottoreti diverso VPC o diverso all'interno della stessaVPC, create un nuovo gruppo di sottoreti DB e lo specificate durante la creazione del clone.

Assicurati di conoscere tutti i seguenti dettagli. Puoi trovarli tutti nell'output degli esempi precedenti.

1. VPCdel cluster originale. Per istruzioni, consulta Fase 3: Controllare le sottoreti del cluster originale.
2. VPCdel clone, se lo state creando in un altroVPC. Per istruzioni, consulta Passaggio 5: verifica VPCs che puoi utilizzare per il clone.
3. Tre AZs associati allo storage Aurora per il cluster originale. Per istruzioni, consulta Fase 1: Verificare le zone di disponibilità del cluster originale.
4. Due o tre AZs associati al gruppo di sottoreti DB per il cluster originale. Per istruzioni, consulta Fase 2: Controllare il gruppo di sottoreti DB del cluster originale.
5. La sottorete IDs associata a AZs tutte le sottoreti del gruppo VPC che intendete utilizzare per il clone. Utilizzate lo stesso `describe-subnets` comando usato inFase 3: Controllare le sottoreti del cluster originale, sostituendo l'ID della destinazione. VPC VPC

Controlla quanti AZs sono entrambi associati all'archiviazione del cluster originale e associati alle sottoreti nella destinazione. VPC Per creare correttamente il clone, devono esserci due o tre AZs elementi in comune. Se ne avete meno di due AZs in comune, tornate aFase 1: Creare le sottoreti per il clone. Crea una, due o tre nuove sottoreti associate all'archiviazione AZs associata al cluster originale.

Scegli le sottoreti nella destinazione VPC che sono associate allo AZs stesso spazio di archiviazione Aurora nel cluster originale. Idealmente, scegline tre. AZs In questo modo avrai la massima flessibilità per distribuire le istanze DB del clone su più istanze AZs per un'elevata disponibilità delle risorse di elaborazione.

Esegui un comando simile al seguente per creare il nuovo gruppo di sottoreti DB. Sostituisci le IDs tue sottoreti nell'elenco. Se specificate la sottorete IDs utilizzando variabili di ambiente, fate attenzione a citare l'elenco dei `--subnet-ids` parametri in modo da conservare le virgolette doppie intorno a. IDs

```
aws rds create-db-subnet-group --db-subnet-group-name my_subnet_group \
```

```
  --subnet-ids '["my_subnet_1","my_subnet_2","my_subnet3"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets for clone'
```

L'esempio seguente mostra l'output parziale del comando. `create-db-subnet-group`

```
{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'my_subnet_group',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets for clone',
        'VpcId': 'vpc-555fc5df555e555dc',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'my_subnet_1',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1c'
                },
                'SubnetStatus': 'Active'
            },
            {
                'SubnetIdentifier': 'my_subnet_2',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1d'
                },
                'SubnetStatus': 'Active'
            }
            ...
        ],
        'SupportedNetworkTypes': [
            'IPV4'
        ]
    }
}
```

A questo punto, non avete ancora creato il clone. Hai creato tutte le risorse pertinenti VPC e le sottoreti in modo da poter specificare i parametri appropriati per i `create-db-instance` comandi `restore-db-cluster-to-point-in-time` and durante la creazione del clone.

Creazione di un clone di Aurora con nuove impostazioni di rete

Una volta verificata la corretta configurazione delle sottoreti e dei VPCs gruppi di sottoreti per l'utilizzo del nuovo clusterAZs, è possibile eseguire l'operazione di clonazione vera e propria. CLIGli esempi

seguenti evidenziano le opzioni quali `--db-subnet-group-name`--`availability-zone`, e `--vpc-security-group-ids` che specificate nei comandi per configurare il clone e le relative istanze DB.

- Passaggio 1: Specificare il gruppo di sottoreti DB per il clone
- Passaggio 2: Specificare le impostazioni di rete per le istanze nel clone
- Fase 3: Stabilire la connettività da un sistema client a un clone

Passaggio 1: Specificare il gruppo di sottoreti DB per il clone

Quando crei il clone, puoi configurare tutte le impostazioni rightVPC, subnet e AZ specificando un gruppo di sottoreti DB. Utilizzate i comandi degli esempi precedenti per verificare tutte le relazioni e le mappature che entrano nel gruppo di sottoreti DB.

Ad esempio, i comandi seguenti dimostrano la clonazione di un cluster originale in un clone. Nel primo esempio, il cluster di origine è associato a due sottoreti e il clone è associato a tre sottoreti. Il secondo esempio mostra il caso opposto, la clonazione da un cluster con tre sottoreti a un cluster con due sottoreti.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-3-subnets \
  --db-cluster-identifier cluster-cloned-to-2-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name two-subnets
```

Se intendi utilizzare istanze Aurora Serverless v2 nel clone, includi un'`--serverless-v2-scaling-configuration`opzione quando crei il clone, come mostrato. In questo modo è possibile utilizzare la `db.serverless` classe durante la creazione di istanze DB nel clone. È inoltre possibile modificare il clone in un secondo momento per aggiungere questo attributo di configurazione di scalabilità. I numeri di capacità in questo esempio consentono a ciascuna istanza Serverless v2 nel cluster di scalare tra 2 e 32 unità di capacità Aurora (). ACUs Per informazioni sulla funzionalità Aurora Serverless v2 e su come scegliere l'intervallo di capacità, vedere. Utilizzo Aurora Serverless v2

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-2-subnets \
  --db-cluster-identifier cluster-cloned-to-3-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name three-subnets \
```

```
  --serverless-v2-scaling-configuration 'MinCapacity=2,MaxCapacity=32'
```

Indipendentemente dal numero di sottoreti utilizzate dalle istanze DB, lo storage Aurora per il cluster di origine e il clone è associato a tre. AZs L'esempio seguente elenca i AZs comandi associati sia al cluster originale che al clone, per entrambi i comandi degli `restore-db-cluster-to-point-in-time` esempi precedenti.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-with-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-with-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d
```

Poiché almeno due dei cluster AZs si sovrappongono tra ogni coppia di cluster originali e cloni, entrambi i cluster possono accedere allo stesso storage Aurora sottostante.

Passaggio 2: Specificare le impostazioni di rete per le istanze nel clone

Quando si creano istanze DB nel clone, per impostazione predefinita queste ereditano il gruppo di sottoreti DB dal cluster stesso. In questo modo, Aurora assegna automaticamente ogni istanza a una

particolare sottorete e la crea nell'AZ associata alla sottorete. Questa scelta è comoda, soprattutto per i sistemi di sviluppo e test, perché non è necessario tenere traccia della sottorete IDs o AZs mentre si aggiungono nuove istanze al clone.

In alternativa, è possibile specificare l'AZ quando si crea un'istanza Aurora DB per il clone. L'AZ specificato deve appartenere all'insieme di AZs quelli associati al clone. Se il gruppo di sottoreti DB che usi per il clone contiene solo due sottoreti, puoi scegliere solo tra quelle AZs associate a quelle due sottoreti. Questa scelta offre flessibilità e resilienza per sistemi ad alta disponibilità, perché potete assicurarvi che l'istanza writer e l'istanza standby reader siano diverse. AZs Oppure, se aggiungi altri lettori al cluster, puoi assicurarti che siano distribuiti su tre. AZs In questo modo, anche nel raro caso di errore di AZ, avrete comunque un'istanza Writer e un'altra istanza Reader in altre dueAZs.

L'esempio seguente aggiunge un'istanza DB di cui è stato effettuato il provisioning a un cluster Aurora SQL Postgre clonato che utilizza un gruppo di sottoreti DB personalizzato.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_postgresql_clone \
  --db-instance-identifier my_postgres_instance \
  --db-subnet-group-name my_new_subnet \
  --engine aurora-postgresql \
  --db-instance-class db.t4g.medium
```

L'esempio seguente mostra l'output parziale di tale comando.

```
{
  'DBInstanceIdentifier': 'my_postgres_instance',
  'DBClusterIdentifier': 'my_aurora_postgresql_clone',
  'DBInstanceClass': 'db.t4g.medium',
  'DBInstanceStatus': 'creating'
  ...
}
```

L'esempio seguente aggiunge un'istanza DB Aurora Serverless v2 a un SQL clone Aurora My che utilizza un gruppo di sottoreti DB personalizzato. Per poter utilizzare le istanze Serverless v2, assicuratevi di specificare l'`--serverless-v2-scaling-configuration`opzione per il comando, come mostrato negli esempi precedenti. `restore-db-cluster-to-point-in-time`

```
aws rds create-db-instance --db-cluster-identifier my_aurora_mysql_clone \
  --db-instance-identifier my_mysql_instance \
  --db-subnet-group-name my_other_new_subnet \
```

```
  --engine aurora-mysql \
  --db-instance-class db.serverless
```

L'esempio seguente mostra l'output parziale di tale comando.

```
{
  'DBInstanceIdentifier': 'my_mysql_instance',
  'DBClusterIdentifier': 'my_aurora_mysql_clone',
  'DBInstanceClass': 'db.serverless',
  'DBInstanceStatus': 'creating'
  ...
}
```

Fase 3: Stabilire la connettività da un sistema client a un clone

Se ti stai già connettendo a un cluster Aurora da un sistema client, potresti voler consentire lo stesso tipo di connettività a un nuovo clone. Ad esempio, potresti connetterti al cluster originale da un'istanza o istanza Amazon Cloud9. EC2 Per consentire connessioni dagli stessi sistemi client o da quelli nuovi creati nella destinazioneVPC, configura sottoreti DB e gruppi di VPC sicurezza equivalenti a quelli di. VPC Specificate quindi il gruppo di sottoreti e i gruppi di sicurezza quando create il clone.

I seguenti esempi configurano un clone Aurora Serverless v2. Tale configurazione si basa sulla combinazione di `--engine-mode provisioned` e `--serverless-v2-scaling-configuration` durante la creazione del cluster DB e quindi sulla `--db-instance-class db.serverless` creazione di ogni istanza DB nel cluster. La modalità `provisioned` motore è l'impostazione predefinita, quindi puoi omettere questa opzione se preferisci.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier serverless-sql-postgres\
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-subnet-group-name 'default-vpc-1234' \
  --vpc-security-group-ids 'sg-4567' \
  --serverless-v2-scaling-configuration 'MinCapacity=0.5,MaxCapacity=16' \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Quindi, quando create le istanze DB nel clone, specificate la stessa opzione. `--db-subnet-group-name` Facoltativamente, è possibile includere l'`--availability-zone`opzione e specificare una delle opzioni AZs associate alle sottoreti in quel gruppo di sottoreti. Tale AZ deve inoltre essere una delle aree AZs associate al cluster originale.

```
aws rds create-db-instance \
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-instance-identifier serverless-sql-postgres-clone-instance \
  --db-instance-class db.serverless \
  --db-subnet-group-name 'default-vpc-987zyx654' \
  --availability-zone 'us-east-1c' \
  --engine aurora-postgresql
```

Spostamento di un cluster da sottoreti pubbliche a sottoreti private

È possibile utilizzare la clonazione per migrare un cluster tra sottoreti pubbliche e private. È possibile eseguire questa operazione aggiungendo ulteriori livelli di sicurezza all'applicazione prima di distribuirla in produzione. Per questo esempio, dovresti avere già configurato le sottoreti private e il NAT gateway prima di iniziare il processo di clonazione con Aurora.

Per i passaggi che coinvolgono Aurora, puoi seguire gli stessi passaggi generali degli esempi precedenti di e. Raccolta di informazioni sull'ambiente di rete Creazione di un clone di Aurora con nuove impostazioni di rete La differenza principale è che, anche se disponete di sottoreti pubbliche che mappano tutte le AZs sottoreti del cluster originale, ora è necessario verificare di disporre di un numero sufficiente di sottoreti private per un cluster Aurora e che tali sottoreti siano associate a tutte le sottoreti utilizzate per l'archiviazione Aurora nel cluster originale. AZs Analogamente ad altri casi d'uso della clonazione, è possibile creare il gruppo di sottoreti DB per il clone con tre o due sottoreti private associate a quanto richiestoAZs. Tuttavia, se si utilizzano due sottoreti private nel gruppo di sottoreti DB, è necessario disporre di una terza sottorete privata associata alla terza AZ utilizzata per l'archiviazione Aurora nel cluster originale.

È possibile consultare questa lista di controllo per verificare che siano soddisfatti tutti i requisiti per eseguire questo tipo di operazione di clonazione.

- Registra AZs i tre associati al cluster originale. Per istruzioni, consulta Fase 1: Verificare le zone di disponibilità del cluster originale.
- Registra le tre o due AZs associate alle sottoreti pubbliche nel gruppo di sottoreti DB per il cluster originale. Per istruzioni, consulta Fase 3: Controllare le sottoreti del cluster originale.
- Crea sottoreti private mappate a tutte e tre le sottoreti associate al cluster originale. AZs Esegui anche qualsiasi altra configurazione di rete, ad esempio la creazione di un NAT gateway, per poter comunicare con le sottoreti private. Per istruzioni, consulta Creare una sottorete nella Guida per l'utente di Amazon Virtual Private Cloud.
- Crea un nuovo gruppo di sottoreti DB contenente tre o due sottoreti private associate al primo AZs punto. Per istruzioni, consulta Fase 2: Creare il gruppo di sottoreti DB per il clone.

Quando tutti i prerequisiti sono soddisfatti, è possibile sospendere l'attività del database sul cluster originale durante la creazione del clone e cambiare l'applicazione per utilizzarlo. Dopo aver creato il clone e aver verificato di potervi connettere, aver eseguito il codice dell'applicazione e così via, potete interrompere l'uso del cluster originale.

End-to-end esempio di creazione di un clone incrociato VPC

La creazione di un clone in un VPC ambiente diverso dall'originale utilizza gli stessi passaggi generali degli esempi precedenti. Poiché l'VPCID è una proprietà delle sottoreti, in realtà non viene specificato come parametro quando si esegue uno qualsiasi dei comandi. VPC RDS CLI La differenza principale è che è più probabile che sia necessario creare nuove sottoreti, nuove sottoreti mappate su specificheAZs, un gruppo di VPC sicurezza e un nuovo gruppo di sottoreti DB. Ciò è particolarmente vero se si tratta del primo cluster Aurora creato al suo interno. VPC

È possibile consultare questa lista di controllo per verificare che siano soddisfatti tutti i requisiti per eseguire questo tipo di operazione di clonazione.

- Registra AZs i tre associati al cluster originale. Per istruzioni, consulta Fase 1: Verificare le zone di disponibilità del cluster originale.
- Registra le tre o due AZs associate alle sottoreti nel gruppo di sottoreti DB per il cluster originale. Per istruzioni, consulta Fase 2: Controllare il gruppo di sottoreti DB del cluster originale.
- Crea sottoreti mappate a tutte e tre quelle associate al AZs cluster originale. Per istruzioni, consulta Fase 1: Creare le sottoreti per il clone.
- Esegui qualsiasi altra configurazione di rete, ad esempio la configurazione di un gruppo di VPC sicurezza, per i sistemi client, i server delle applicazioni e così via per poter comunicare con le istanze DB nel clone. Per istruzioni, consulta Controllo dell'accesso con i gruppi di sicurezza.
- Crea un nuovo gruppo di sottoreti DB contenente tre o due delle sottoreti associate a Fin dal AZs primo punto. Per istruzioni, consulta Fase 2: Creare il gruppo di sottoreti DB per il clone.

Quando tutti i prerequisiti sono soddisfatti, è possibile sospendere l'attività del database sul cluster originale durante la creazione del clone e cambiare l'applicazione per utilizzarlo. Dopo aver creato il clone e aver verificato di potervi connettere, eseguire il codice dell'applicazione e così via, potete valutare se mantenere attivi sia l'originale che i cloni oppure interrompere l'uso del cluster originale.

I seguenti esempi Linux mostrano la sequenza di AWS CLI operazioni per clonare un cluster Aurora DB da VPC uno all'altro. Alcuni campi non pertinenti agli esempi non vengono visualizzati nell'output del comando.

Innanzitutto, controlliamo l'IDsorigine e la destinazioneVPCs. Il nome descrittivo che assegnate a un oggetto VPC quando lo create è rappresentato come un tag nei metadatiVPC.

```
$ aws ec2 describe-vpcs --query '*[].[VpcId,Tags]'
[
    [
        'vpc-0f0c0fc0000b0ffb0',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-source'
            }
        ]
    ],
    [
        'vpc-9e99d9f99a999bd99',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-dest'
            }
        ]
    ]
]
```

Il cluster originale esiste già nell'origine. VPC Per configurare il clone utilizzando lo stesso set AZs di storage Aurora, controlliamo quello usato AZs dal cluster originale.

```
$ aws rds describe-db-clusters --db-cluster-identifier original-cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

Ci assicuriamo che ci siano sottoreti che corrispondano a quelle AZs utilizzate dal cluster originale:`us-east-1c`, e. `us-east-1d` `us-east-1f`

```
$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
  --availability-zone us-east-1c --cidr-block 10.0.0.128/28
{
    'Subnet': {
```

```
        'AvailabilityZone': 'us-east-1c',
        'SubnetId': 'subnet-3333a33be3ef3e333',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1d --cidr-block 10.0.0.160/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1d',
        'SubnetId': 'subnet-4eeb444cd44b4d444',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1f --cidr-block 10.0.0.224/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1f',
        'SubnetId': 'subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}
```

Questo esempio conferma che esistono sottoreti mappate al necessario AZs nella destinazione. VPC

```
aws ec2 describe-subnets --query 'sort_by(*[] | [?VpcId == `vpc-9e99d9f99a999bd99`] |
[].{SubnetId:SubnetId,VpcId:VpcId,AvailabilityZone:AvailabilityZone},
 &AvailabilityZone)' --output table

-------------------------------------------------------------------------
|                            DescribeSubnets                            |
+------------------+---------------------------+-------------------------+
| AvailabilityZone |         SubnetId          |          VpcId          |
+------------------+---------------------------+-------------------------+
|  us-east-1a      |  subnet-000ff0e00000c0aea  |  vpc-9e99d9f99a999bd99  |
|  us-east-1b      |  subnet-1111d111111ca11b1  |  vpc-9e99d9f99a999bd99  |
|  us-east-1c      | subnet-3333a33be3ef3e333   |  vpc-9e99d9f99a999bd99  |
|  us-east-1d      | subnet-4eeb444cd44b4d444   |  vpc-9e99d9f99a999bd99  |
|  us-east-1f      | subnet-66eea6666fb66d66c   |  vpc-9e99d9f99a999bd99  |
+------------------+---------------------------+-------------------------+
```

Prima di creare un cluster Aurora DB inVPC, è necessario disporre di un gruppo di sottoreti DB con sottoreti mappate allo storage utilizzato per AZs Aurora. Quando si crea un cluster normale, è possibile utilizzare qualsiasi set di tre. AZs Quando si clona un cluster esistente, il gruppo di sottoreti deve corrispondere ad almeno due dei tre AZs utilizzati per l'archiviazione Aurora.

```
$ aws rds create-db-subnet-group \
  --db-subnet-group-name subnet-group-in-other-vpc \
  --subnet-ids
 '["subnet-3333a33be3ef3e333","subnet-4eeb444cd44b4d444","subnet-66eea6666fb66d66c"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c'

{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'subnet-group-in-other-vpc',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'subnet-4eeb444cd44b4d444',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1d' }
            },
            {
                'SubnetIdentifier': 'subnet-3333a33be3ef3e333',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1c' }
            },
            {
                'SubnetIdentifier': 'subnet-66eea6666fb66d66c',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1f' }
            }
        ]
    }
}
```

Ora le sottoreti e il gruppo di sottoreti DB sono a posto. L'esempio seguente mostra `restore-db-cluster-to-point-in-time` che clona il cluster. L'`--db-subnet-group-name`opzione associa il clone al set corretto di sottoreti mappate al set corretto del cluster originale. AZs

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier original-cluster \
```

```
  --db-cluster-identifier clone-in-other-vpc \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name subnet-group-in-other-vpc

{
  'DBClusterIdentifier': 'clone-in-other-vpc',
  'DBSubnetGroup': 'subnet-group-in-other-vpc',
  'Engine': 'aurora-postgresql',
  'EngineVersion': '15.4',
  'Status': 'creating',
  'Endpoint': 'clone-in-other-vpc.cluster-c0abcdef.us-east-1.rds.amazonaws.com'
}
```

L'esempio seguente conferma che lo storage Aurora nel clone utilizza lo stesso set di del AZs cluster originale.

```
$ aws rds describe-db-clusters --db-cluster-identifier clone-in-other-vpc \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

A questo punto, è possibile creare istanze DB per il clone. Assicuratevi che il gruppo VPC di sicurezza associato a ciascuna istanza consenta le connessioni dagli intervalli di indirizzi IP utilizzati per le EC2 istanze, i server delle applicazioni e così via presenti nella destinazione. VPC

Clonazione tra account con AWS RAM e Amazon Aurora

Utilizzando AWS Resource Access Manager (AWS RAM) con Amazon Aurora, puoi condividere i cluster e i cloni di Aurora DB che appartengono al tuo AWS account con un altro account o organizzazione. AWS La clonazione tra più account è più veloce in queste situazioni piuttosto che con la creazione e il ripristino di uno snapshot di un database. Puoi creare un clone di uno dei tuoi cluster database Aurora e condividerlo. Oppure puoi condividere il tuo cluster Aurora DB con un altro AWS account e consentire al titolare dell'account di creare il clone. L’approccio scelto varia a seconda del caso d'uso.

Ad esempio, potrebbe essere necessario condividere regolarmente un clone del database finanziario con il team di verifica interno dell'organizzazione. In questo caso, il team di verifica utilizza il suo account AWS per le applicazioni. Puoi concedere all' AWS account del team di controllo l'autorizzazione ad accedere al tuo cluster Aurora DB e clonarlo secondo necessità.

D'altra parte, se un fornitore esterno verifica i tuoi dati finanziari, potresti preferire creare il clone da solo. È quindi possibile concedere al fornitore esterno solo l'accesso al clone.

Puoi anche utilizzare la clonazione tra account per supportare molti degli stessi casi d'uso per la clonazione all'interno dello stesso AWS account, ad esempio sviluppo e test. Ad esempio, l'organizzazione potrebbe utilizzare AWS account diversi per la produzione, lo sviluppo, i test e così via. Per ulteriori informazioni, consulta Panoramica sulla clonazione Aurora.

Pertanto, potresti voler condividere un clone con un altro AWS account o consentire a un altro AWS account di creare cloni dei tuoi cluster Aurora DB. In entrambi i casi, iniziate a usare per creare un oggetto AWS RAM di condivisione. Per informazioni complete sulla condivisione di AWS risorse tra AWS account, consulta la Guida AWS RAM per l'utente.

La creazione di un clone tra più account richiede azioni da parte dell' AWS account proprietario del cluster originale e dell' AWS account che crea il clone. Innanzitutto, il proprietario modifica il cluster per permettere a uno o più account di poterlo clonare. Se uno degli account appartiene a un' AWS organizzazione diversa, AWS genera un invito alla condivisione. L'altro account deve accettare l'invito prima di procedere. Dunque, ogni account autorizzato può clonare il cluster. Durante questo processo, il cluster è definito dal suo unico Amazon Resource Name (ARN).

Come per la clonazione all'interno dello stesso AWS account, lo spazio di archiviazione aggiuntivo viene utilizzato solo se vengono apportate modifiche ai dati dall'origine o dal clone. Gli addebiti per l’archiviazione vengono quindi applicati da quel momento. Se si elimina il cluster di origine, i costi di storage sono distribuiti equamente tra i cluster clonati rimanenti.

Argomenti

- Restrizioni della clonazione tra più account
- Consentire ad altri AWS account di clonare il cluster
- Clonazione di un cluster di proprietà di un altro account AWS

Restrizioni della clonazione tra più account

La clonazione tra più account di Aurora presenta le restrizioni seguenti:

- Non è possibile clonare un Aurora Serverless v1 raggruppati tra più AWS account.
- Non puoi visualizzare o accettare inviti a risorse condivise con. AWS Management Console Usa l' AWS CLI API Amazon RDS o la AWS RAM console per visualizzare e accettare inviti a risorse condivise.

- Puoi creare solo un nuovo clone da una risorsa condivisa con le tue. Account AWS Questo vale sia che la risorsa condivisa sia un cluster Aurora DB originale o un clone creato in precedenza.
- Puoi creare solo un nuovo clone da un clone che è stato condiviso con il tuo account. AWS
- Non puoi condividere risorse (cloni o cluster Aurora DB) che sono state condivise con il tuo account. AWS
- Non è possibile creare più di 15 cloni tra account da un singolo cluster di database Aurora.
- Ciascuno dei 15 cloni di più account deve appartenere a un account diverso. AWS In altre parole, puoi creare un solo clone di un cluster su più account all'interno di qualsiasi account. AWS
- Dopo la clonazione, il cluster originale e il relativo clone sono considerati identici ai fini dell'applicazione dei limiti sui cloni tra account. Non è possibile creare cloni tra più account sia del cluster originale che del cluster clonato all'interno dello stesso account. AWS Il numero totale di cloni tra account per il cluster originale e uno dei suoi cloni non può superare 15.
- Non puoi condividere un cluster Aurora DB con altri AWS account a meno che il cluster non si trovi in uno `ACTIVE` stato.
- Non è possibile rinominare un cluster Aurora DB che è stato condiviso con AWS altri account.
- Non puoi creare un clone tra più account di un cluster crittografato con una chiave RDS predefinita.
- Non è possibile creare cloni non crittografati in un AWS account da cluster Aurora DB crittografati che sono stati condivisi da un altro account. AWS Il proprietario del cluster deve inoltre concedere l'autorizzazione per accedere alla AWS KMS key del cluster di origine. Puoi tuttavia utilizzare una chiave differente quando crei il clone.

Consentire ad altri AWS account di clonare il cluster

Per consentire ad altri AWS account di clonare un cluster di tua proprietà, utilizza AWS RAM per impostare l'autorizzazione di condivisione. In questo modo viene inoltre inviato un invito a ciascuno degli altri account che fanno parte di un' AWS organizzazione diversa.

Per le procedure per condividere le risorse di tua proprietà nella AWS RAM console, consulta Condivisione delle risorse di tua proprietà nella Guida per l'AWS RAM utente.

Argomenti

- Concessione dell'autorizzazione ad altri AWS account per clonare il cluster
- Verifica se un cluster di tua proprietà è condiviso con altri account AWS

Concessione dell'autorizzazione ad altri AWS account per clonare il cluster

Se il cluster che condividi è crittografato, condividi anche la AWS KMS key per il cluster. Puoi consentire agli utenti o ai ruoli AWS Identity and Access Management (IAM) di un AWS account di utilizzare una chiave KMS in un account diverso.

Per fare ciò, devi prima aggiungere l'account esterno (utente root) alla politica chiave della chiave KMS tramite. AWS KMS Non aggiungi singoli utenti o ruoli alla policy delle chiavi, ma solo l'account proprietario esterno. Puoi condividere solamente una chiave KMS che hai creato tu stesso, non la chiave del servizio RDS di default. Per ulteriori informazioni sul controllo accessi per le chiavi KMS, consulta Autenticazione e controllo degli accessi per AWS KMS.

Console

Come concedere l'autorizzazione per clonare il tuo cluster

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli il cluster database che desideri condividere per visualizzare la pagina Dettagli e scegli la scheda Connectivity & security (Connessione e sicurezza).
4. Nella sezione Condividi il cluster DB con altri AWS account, inserisci l'ID numerico dell'account a cui desideri consentire la AWS clonazione di questo cluster. Per un account IDs appartenente alla stessa organizzazione, puoi iniziare a digitare nella casella e quindi scegliere dal menu.

   > **Important**
   >
   > In alcuni casi, potresti volere la clonazione di un cluster da parte di un account che non appartiene alla stessa organizzazione AWS del tuo account. In questi casi, per ragioni di sicurezza la console non segnala chi sia il proprietario dell'ID account o se l'account esiste.
   >
   > Fai attenzione a inserire numeri di conto che non appartengono alla stessa AWS organizzazione del tuo AWS account. Verifica immediatamente che la condivisione sia avvenuta con l'account desiderato.

5. Nella pagina di conferma, verifica che l'ID dell'account specificato sia corretto. Inserisci `share` nel riquadro di conferma per confermare.

Nella pagina Dettagli, viene visualizzata una voce che mostra l'ID dell' AWS account specificato in Account con cui è condiviso questo cluster DB. La colonna Stato mostra inizialmente lo stato Pending (In attesa).

6. Contatta il proprietario dell'altro AWS account o accedi a quell'account se possiedi entrambi. Fornisci istruzioni al proprietario dell'altro account per accettare l'invito di condivisione e clonare il cluster database, nel modo descritto di seguito.

AWS CLI

Come concedere l'autorizzazione per clonare il tuo cluster

1. Raccogli le informazioni per i parametri richiesti. È necessario l'ARN per il cluster e l'ID numerico per l'altro account. AWS
2. Esegui il comando AWS RAM CLI. `create-resource-share`

   In Linux, macOS, oppure Unix:

   ```
   aws ram create-resource-share --name descriptive_name \
     --region region \
     --resource-arns cluster_arn \
     --principals other_account_ids
   ```

   In Windows:

   ```
   aws ram create-resource-share --name descriptive_name ^
     --region region ^
     --resource-arns cluster_arn ^
     --principals other_account_ids
   ```

   Per includere più account IDs per il `--principals` parametro, separateli l'uno IDs dall'altro con spazi. Per specificare se l'account consentito IDs può trovarsi all'esterno AWS dell'organizzazione, includi il `--no-allow-external-principals` parametro `--allow-external-principals` o per`create-resource-share`.

AWS RAM API

Come concedere l'autorizzazione per clonare il tuo cluster

1. Raccogli le informazioni per i parametri richiesti. È necessario l'ARN per il cluster e l'ID numerico per l'altro account. AWS
2. Chiama l'operazione AWS RAM API CreateResourceShare e specifica i seguenti valori:
   - Specificate l'ID dell'account per uno o più AWS account come `principals` parametro.
   - Specifica l'ARN di uno o più cluster database Aurora come parametro `resourceArns`.
   - Specificate se l'account consentito IDs può essere esterno all' AWS organizzazione includendo un valore booleano per il `allowExternalPrincipals` parametro.

Nuova creazione di un cluster che usa la chiave RDS predefinita

Se il cluster crittografato che desideri condividere utilizza la chiave RDS predefinita, assicurati di creare nuovamente il cluster. A tale scopo, crea uno snapshot manuale del cluster DB, utilizza una AWS KMS key e quindi ripristina il cluster in un nuovo cluster. Quindi condividi il nuovo cluster. Per eseguire questo processo, procedi come indicato di seguito.

Ricreare un cluster crittografato che usa una chiave RDS predefinita

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Snapshots (Snapshot) dal riquadro di navigazione.
3. Scegli il tuo snapshot.
4. Sotto Operazioni, scegli Copy Snapshot (Copia snapshot) e poi Enable encryption (Abilita crittografia).
5. Per AWS KMS key, scegli la nuova chiave di crittografia che desideri usare.
6. Ripristina lo snapshot copiato. A tale scopo, segui la procedura in Ripristino da uno snapshot cluster database. La nuova istanza database usa la nuova chiave di crittografia.
7. (Opzionale) Elimina il vecchio cluster database se non ne hai più bisogno. A tale scopo, segui la procedura in Eliminazione di una snapshot del cluster di database. Prima di farlo, conferma che il tuo nuovo cluster abbia tutti i dati necessari e che la tua applicazione può accedere correttamente.

Verifica se un cluster di tua proprietà è condiviso con altri account AWS

Puoi controllare se altri utenti hanno autorizzazione a condividere un cluster. Farlo ti aiuta a capire se il cluster si sta avvicinando al limite massimo di numeri di cloni tra più account.

Per le procedure per condividere le risorse utilizzando la AWS RAM console, consulta Condivisione delle risorse di tua proprietà nella Guida per l'AWS RAM utente.

AWS CLI

Per scoprire se un cluster di tua proprietà è condiviso con altri AWS account

- Chiama il comando AWS RAM CLI `list-principals`, utilizzando l'ID dell'account come proprietario della risorsa e l'ARN del cluster come ARN della risorsa. Puoi visualizzare tutte le condivisioni con il comando seguente. I risultati indicano a quali AWS account è consentito clonare il cluster.

```
aws ram list-principals \
    --resource-arns your_cluster_arn \
    --principals your_aws_id
```

AWS RAM API

Per scoprire se un cluster di tua proprietà è condiviso con altri AWS account

- Chiama l'operazione AWS RAM API ListPrincipals. Usa il tuo ID dell'account come proprietario di risorse e l'ARN del tuo cluster come l'ARN di risorse.

Clonazione di un cluster di proprietà di un altro account AWS

Per clonare un cluster di proprietà di un altro AWS account, usa AWS RAM per ottenere l'autorizzazione a creare il clone. Dopo aver ottenuto l'autorizzazione richiesta, puoi usare la procedura standard per clonare un cluster Aurora.

Puoi anche verificare se un cluster di tua proprietà è un clone di un cluster di proprietà di un account diverso. AWS

Per le procedure di utilizzo delle risorse di proprietà di altri utenti nella AWS RAM console, vedi Accesso alle risorse condivise con te nella Guida per l'AWS RAM utente.

Argomenti

- Visualizzazione degli inviti a clonare i cluster di proprietà di altri account AWS
- Accettazione di inviti a condividere cluster di proprietà di altri account AWS
- Clonazione di un cluster Aurora di proprietà di un altro account AWS
- Verifica se un cluster database è un clone tra più account

Visualizzazione degli inviti a clonare i cluster di proprietà di altri account AWS

Per gestire gli inviti a clonare i cluster di proprietà di AWS account di altre AWS organizzazioni, usa la AWS CLI, la console o l' AWS RAM API. AWS RAM Al momento, non puoi eseguire questa procedura utilizzando la console Amazon RDS.

Per le procedure di utilizzo degli inviti nella AWS RAM console, consulta Accesso alle risorse condivise con te nella Guida per l'utente.AWS RAM

AWS CLI

Per visualizzare gli inviti a clonare i cluster di proprietà di altri account AWS

1. Esegui il comando AWS RAM CLI. `get-resource-share-invitations`

   ```
   aws ram get-resource-share-invitations --region region_name
   ```

   I risultati del comando precedente mostrano tutti gli inviti per clonare cluster, inclusi gli inviti già accettati o rifiutati.
2. (Opzionale) Filtra l'elenco per vedere solamente gli inviti che necessitano una tua azione. A tale scopo, aggiungi il parametro `--query 'resourceShareInvitations[?status==`PENDING`]'`.

AWS RAM API

Per visualizzare gli inviti a clonare i cluster di proprietà di altri account AWS

1. Chiama l'operazione API AWS RAM . `GetResourceShareInvitations` Questa operazione restituisce tutti questi inviti, inclusi gli inviti già accettati o rifiutati.
2. (Opzionale) Trova solamente gli inviti che necessitano una tua azione verificando il campo di restituzione `resourceShareAssociations` per un valore `status` di `PENDING`.

Accettazione di inviti a condividere cluster di proprietà di altri account AWS

Puoi accettare inviti a condividere cluster di proprietà di altri AWS account appartenenti a organizzazioni diverse. AWS Per lavorare con questi inviti, usa il AWS CLI, the AWS RAM e RDS APIs o la console. AWS RAM Al momento, non puoi eseguire questa procedura utilizzando la console RDS.

Per le procedure di utilizzo degli inviti nella AWS RAM console, vedi Accesso alle risorse condivise con te nella Guida per l'AWS RAM utente.

AWS CLI

Per accettare un invito a condividere un cluster da un altro account AWS

1. Trova l'ARN dell'invito eseguendo il comando AWS RAM CLI `get-resource-share-invitations`, come mostrato in precedenza.
2. Accettate l'invito chiamando il comando AWS RAM CLI `accept-resource-share-invitation`, come illustrato di seguito.

   In Linux, macOS, oppure Unix:

   ```
   aws ram accept-resource-share-invitation \
     --resource-share-invitation-arn invitation_arn \
     --region region
   ```

   In Windows:

   ```
   aws ram accept-resource-share-invitation ^
     --resource-share-invitation-arn invitation_arn ^
     --region region
   ```

AWS RAM e API RDS

Come accettare inviti per condividere il cluster di qualcuno

1. Trova l'ARN dell'invito chiamando l'operazione AWS RAM API `GetResourceShareInvitations`, come mostrato in precedenza.
2. Passa quell'ARN come `resourceShareInvitationArn` parametro all'operazione API RDS. AcceptResourceShareInvitation

Clonazione di un cluster Aurora di proprietà di un altro account AWS

Dopo aver accettato l'invito dall' AWS account proprietario del cluster DB, come illustrato in precedenza, è possibile clonare il cluster.

Console

Per clonare un cluster Aurora di proprietà di un altro account AWS

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).

   In cima all'elenco database, dovresti visualizzare uno o più elementi con un valore Ruolo di `Shared from account #account_id`. Per ragioni di sicurezza, puoi visualizzare solamente informazioni limitate riguardo i cluster originali. Puoi visualizzare le proprietà, come motore di database e versione, che devono essere uguali nel tuo cluster clonato.
3. Scegli il cluster che desideri clonare.
4. In Actions (Operazioni), selezionare Create clone (Crea clone).
5. Segui la procedura in Console per completare la configurazione del cluster clonato.
6. Abilita la crittografia per il cluster clonato come necessario. Se il cluster che stai clonando è crittografato, devi abilitare la crittografia per il cluster clonato. L'account AWS che ha condiviso con te il cluster deve condividere anche la chiave KMS utilizzata per crittografare il cluster. È possibile utilizzare la stessa chiave KMS per crittografare il clone o la propria chiave KMS. Non puoi creare un clone tra più account per un cluster crittografato con una chiave KMS di default.

   L'account che possiede la chiave di crittografia deve concedere l'autorizzazione per utilizzare la chiave all'account di destinazione utilizzando la policy delle chiavi. Questo processo è simile a quello per condividere gli snapshot crittografati, utilizzando una policy delle chiavi per concedere l'autorizzazione all'account di destinazione per utilizzare la chiave.

AWS CLI

Per clonare un cluster Aurora di proprietà di un altro account AWS

1. Accetta l'invito dell' AWS account proprietario del cluster DB, come mostrato in precedenza.

2. Clona il cluster specificando l'ARN completo del cluster di origine nel parametro `source-db-cluster-identifier` del comando RDS CLI `restore-db-cluster-to-point-in-time`, come mostrato di seguito.

   Se l'ARN passato come `source-db-cluster-identifier` non è stato condiviso, si presenterà lo stesso errore del caso in cui il cluster indicato non esista.

   In Linux, macOS, oppure Unix:

   ```
   aws rds restore-db-cluster-to-point-in-time \
     --source-db-cluster-identifier=arn:aws:rds:arn_details \
     --db-cluster-identifier=new_cluster_id \
     --restore-type=copy-on-write \
     --use-latest-restorable-time
   ```

   In Windows:

   ```
   aws rds restore-db-cluster-to-point-in-time ^
     --source-db-cluster-identifier=arn:aws:rds:arn_details ^
     --db-cluster-identifier=new_cluster_id ^
     --restore-type=copy-on-write ^
     --use-latest-restorable-time
   ```

3. Se il cluster da clonare è crittografato, devi crittografare il tuo cluster clonato includendo un parametro `kms-key-id`. Questo valore `kms-key-id` può essere lo stesso di quello utilizzato per crittografare il cluster database originale o in alternativa la tua chiave KMS. Il tuo account deve avere l'autorizzazione a utilizzare la chiave di crittografia.

   In Linux, macOS, oppure Unix:

   ```
   aws rds restore-db-cluster-to-point-in-time \
     --source-db-cluster-identifier=arn:aws:rds:arn_details \
     --db-cluster-identifier=new_cluster_id \
     --restore-type=copy-on-write \
     --use-latest-restorable-time \
     --kms-key-id=arn:aws:kms:arn_details
   ```

   In Windows:

   ```
   aws rds restore-db-cluster-to-point-in-time ^
   ```

```
  --source-db-cluster-identifier=arn:aws:rds:arn_details ^
  --db-cluster-identifier=new_cluster_id ^
  --restore-type=copy-on-write ^
  --use-latest-restorable-time ^
  --kms-key-id=arn:aws:kms:arn_details
```

L'account che possiede la chiave di crittografia deve concedere l'autorizzazione per utilizzare la chiave all'account di destinazione utilizzando la policy delle chiavi. Questo processo è simile a quello per condividere gli snapshot crittografati, utilizzando una policy delle chiavi per concedere l'autorizzazione all'account di destinazione per utilizzare la chiave. Di seguito, un esempio di una policy delle chiavi.

```
{
  "Id": "key-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::account_id:user/KeyUser",
        "arn:aws:iam::account_id:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::account_id:user/KeyUser",
        "arn:aws:iam::account_id:root"
      ]},
      "Action": [
        "kms:CreateGrant",
```

```
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
    }
  ]
}
```

> **Note**
>
> Il comando restore-db-cluster-to- point-in-time AWS CLI ripristina solo il cluster DB, non le istanze DB per quel cluster DB. Per creare istanze DB per il cluster DB ripristinato, richiama il comando. create-db-instance Specifica l'identificatore del cluster database ripristinato in `--db-cluster-identifier`.
> Puoi creare le istanze database solo dopo che il comando `restore-db-cluster-to-point-in-time` è terminato e il cluster database è disponibile.

API RDS

Per clonare un cluster Aurora di proprietà di un altro account AWS

1. Accetta l'invito dell' AWS account proprietario del cluster DB, come mostrato in precedenza.
2. Clona il cluster specificando l'ARN completo del cluster di origine nel parametro `SourceDBClusterIdentifier` dell'operazione API RDS `RestoreDBClusterToPointInTime`.

   Se l'ARN passato come `SourceDBClusterIdentifier` non è stato condiviso, allora si presenterà lo stesso errore del caso in cui il cluster indicato non esista.
3. Se il cluster da clonare è crittografato, includi un parametro `KmsKeyId` per crittografare il tuo cluster clonato. Questo valore `kms-key-id` può essere lo stesso di quello utilizzato per crittografare il cluster database originale o in alternativa la tua chiave KMS. Il tuo account deve avere l'autorizzazione a utilizzare la chiave di crittografia.

   Durante la clonazione di un volume, l'account di destinazione deve essere autorizzato all'utilizzo della chiave di crittografia usata per crittografare il cluster di origine. Aurora crittografa il nuovo cluster clonato con la chiave di crittografia specificata in `KmsKeyId`.

L'account che possiede la chiave di crittografia deve concedere l'autorizzazione per utilizzare la chiave all'account di destinazione utilizzando la policy delle chiavi. Questo processo è simile a quello per condividere gli snapshot crittografati, utilizzando una policy delle chiavi per concedere l'autorizzazione all'account di destinazione per utilizzare la chiave. Di seguito, un esempio di una policy delle chiavi.

```
{
  "Id": "key-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::account_id:user/KeyUser",
        "arn:aws:iam::account_id:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::account_id:user/KeyUser",
        "arn:aws:iam::account_id:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
    }
```

```
  ]
}
```

> **Note**
>
> L'operazione DBClusterToPointInTimeRestore RDS API ripristina solo il cluster DB, non le istanze DB per quel cluster di database. Per creare istanze DB per il cluster DB ripristinato, richiama l'operazione Create RDS API. DBInstance Specifica l'identificatore del cluster database ripristinato in `DBClusterIdentifier`. È possibile creare le istanze database solo dopo che l'operazione `RestoreDBClusterToPointInTime` è completata e il cluster di database è disponibile.

Verifica se un cluster database è un clone tra più account

L'oggetto `DBClusters` identifica se ciascun cluster è un clone tra più account. Puoi visualizzare il cluster che puoi clonare utilizzando l'opzione`include-shared` quando esegui il comando RDS CLI `describe-db-clusters`. Tuttavia, non puoi visualizzare la maggior parte dei dettagli di configurazione per tali cluster.

AWS CLI

Come verificare se un cluster database è un clone tra più account

- Denomina il comando RDS CLI `describe-db-clusters`.

  I seguenti esempi mostrano come sono visualizzati i cluster database effettivi o potenziali di cloni tra più account nell'output `describe-db-clusters`. Per i cluster esistenti di proprietà del tuo AWS account, il `CrossAccountClone` campo indica se il cluster è un clone di un cluster DB di proprietà di un altro account. AWS

  In alcuni casi, una voce potrebbe avere un numero di AWS account diverso dal tuo nel `DBClusterArn` campo. In questo caso, quella voce rappresenta un cluster di proprietà di un AWS account diverso e che puoi clonare. Tali voci presentano pochi campi oltre `DBClusterArn`. Quando crei un cluster clonato, specifica che i valori per `StorageEncrypted`, `Engine` e `EngineVersion` siano gli stessi del cluster originale.

  ```
  $aws rds describe-db-clusters --include-shared --region us-east-1
  {
  ```

```
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0
  ]
}
```

API RDS

Come verificare se un cluster database è un clone tra più account

- Richiama l'operazione Descrivi dell'API RDS. DBClusters

  Per i cluster esistenti di proprietà del tuo AWS account, il `CrossAccountClone` campo indica se il cluster è un clone di un cluster DB di proprietà di un altro account. AWS Le voci con un numero di AWS account diverso nel `DBClusterArn` campo rappresentano cluster che è possibile clonare e che sono di proprietà di altri account. AWS Tali voci presentano pochi campi oltre `DBClusterArn`. Quando crei un cluster clonato, specifica che i valori per `StorageEncrypted`, `Engine` e `EngineVersion` siano gli stessi del cluster originale.

  Il seguente esempio mostra un valore di ritorno che dimostra sia i cluster clonati effettivi che potenziali.

```
{
```

```
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0"
      }
  ]
}
```

Integrazione di Aurora con altri servizi AWS

Integra Amazon Aurora con altri AWS servizi in modo da poter estendere il cluster Aurora DB per utilizzare funzionalità aggiuntive nel cloud. AWS

Argomenti

- Integrazione di AWS servizi con Amazon Aurora My SQL
- Integrazione di AWS servizi con Amazon Aurora Postgre SQL

Integrazione di AWS servizi con Amazon Aurora My SQL

Amazon Aurora My SQL si integra con altri AWS servizi in modo da poter estendere il cluster Aurora My SQL DB per utilizzare funzionalità aggiuntive nel cloud. AWS Il cluster Aurora My SQL DB può utilizzare AWS i servizi per eseguire le seguenti operazioni:

- Richiama una funzione in modo sincrono o asincrono utilizzando le AWS Lambda funzioni native o. `lambda_sync` `lambda_async` In alternativa, chiamata asincrona di una funzione AWS Lambda utilizzando la procedura `mysql.lambda_async`.
- Carica dati da testo o XML file archiviati in un bucket Amazon S3 nel tuo cluster DB utilizzando il `LOAD DATA FROM S3` comando or. `LOAD XML FROM S3`
- Salvataggio dei dati nei file di testo archiviati in un bucket Amazon S3 dal cluster DB utilizzando il comando `SELECT INTO OUTFILE S3`.
- Aggiunta o rimozione automatica di repliche Aurora con Application Auto Scaling. Per ulteriori informazioni, consulta Auto Scaling di Amazon Aurora con repliche Aurora.

Per ulteriori informazioni sull'integrazione di Aurora SQL My con AWS altri servizi, consulta. Integrazione di Amazon Aurora SQL My con altri servizi AWS

Integrazione di AWS servizi con Amazon Aurora Postgre SQL

Amazon Aurora Postgre SQL si integra con altri AWS servizi in modo da poter estendere il cluster SQL DB Aurora Postgre per utilizzare funzionalità aggiuntive nel cloud. AWS Il cluster Aurora Postgre SQL DB può utilizzare i AWS servizi per eseguire le seguenti operazioni:

- Raccolta, visualizzazione e valutazione delle prestazioni in modo rapido per i carichi di lavoro del database relazionale con Performance Insights.

- Aggiunta o rimozione automatica di repliche Aurora con Aurora Auto Scaling. Per ulteriori informazioni, consulta Auto Scaling di Amazon Aurora con repliche Aurora.

Per ulteriori informazioni sull'integrazione di Aurora SQL Postgre con AWS altri servizi, consulta. Integrazione di Amazon Aurora SQL Postgre con altri servizi AWS

Manutenzione di un cluster database Amazon Aurora

Periodicamente, Amazon RDS esegue la manutenzione delle risorse Amazon RDS. I seguenti argomenti descrivono queste azioni di manutenzione e come applicarle.

Panoramica degli aggiornamenti di manutenzione dei cluster di database delle

La manutenzione spesso comporta aggiornamenti alle seguenti risorse del cluster database:

- Hardware sottostante
- Sistema operativo (OS) sottostante
- Versione del motore del database

Gli aggiornamenti al sistema operativo si verificano generalmente per problemi di sicurezza. Ti consigliamo di eseguirli il prima possibile. Per ulteriori informazioni sugli aggiornamenti del sistema operativo, vedere .

Argomenti

- Risorse offline durante gli aggiornamenti di manutenzione
- Modifiche differite dell'istanza DB e del cluster DB
- Eventuale coerenza per l'API DescribePendingMaintenanceActions

Risorse offline durante gli aggiornamenti di manutenzione

Per alcune operazioni di manutenzione, Amazon RDS deve portare offline il cluster database per un breve intervallo di tempo. Tra le operazioni di manutenzione che richiedono l'impostazione offline di una risorsa si annovera l'applicazione delle patch necessarie al sistema operativo o al database. L'applicazione delle patch necessarie viene pianificata automaticamente solo per le patch correlate alla sicurezza e all'affidabilità dell'istanza. Tali patch si verificano raramente, in genere una volta ogni pochi mesi. Raramente richiedono più di una frazione del periodo di manutenzione.

Modifiche differite dell'istanza DB e del cluster DB

Le modifiche differite del cluster e dell'istanza DB che hai scelto di non applicare immediatamente vengono applicate durante la finestra di manutenzione. Ad esempio, è possibile scegliere di

modificare le classi di istanza database o i gruppi di parametri cluster o database durante la finestra di manutenzione. Le modifiche specificate utilizzando l'impostazione di riavvio in sospeso non vengono visualizzate nell'elenco Manutenzione in sospeso . Per ulteriori informazioni sulla modifica di un cluster database, consulta Modifica di un cluster database Amazon Aurora.

Per visualizzare le modifiche in sospeso per la prossima finestra di manutenzione, usa il describe-db-clusters AWS CLI comando e controlla il campo. `PendingModifiedValues`

Eventuale coerenza per l'API DescribePendingMaintenanceActions

L'`DescribePendingMaintenanceActions`API Amazon RDS segue un modello di coerenza finale. Ciò significa che il risultato del `DescribePendingMaintenanceActions` comando potrebbe non essere immediatamente visibile a tutti i comandi RDS successivi. Tienilo a mente quando lo usi `DescribePendingMaintenanceActions` subito dopo aver usato un comando API precedente.

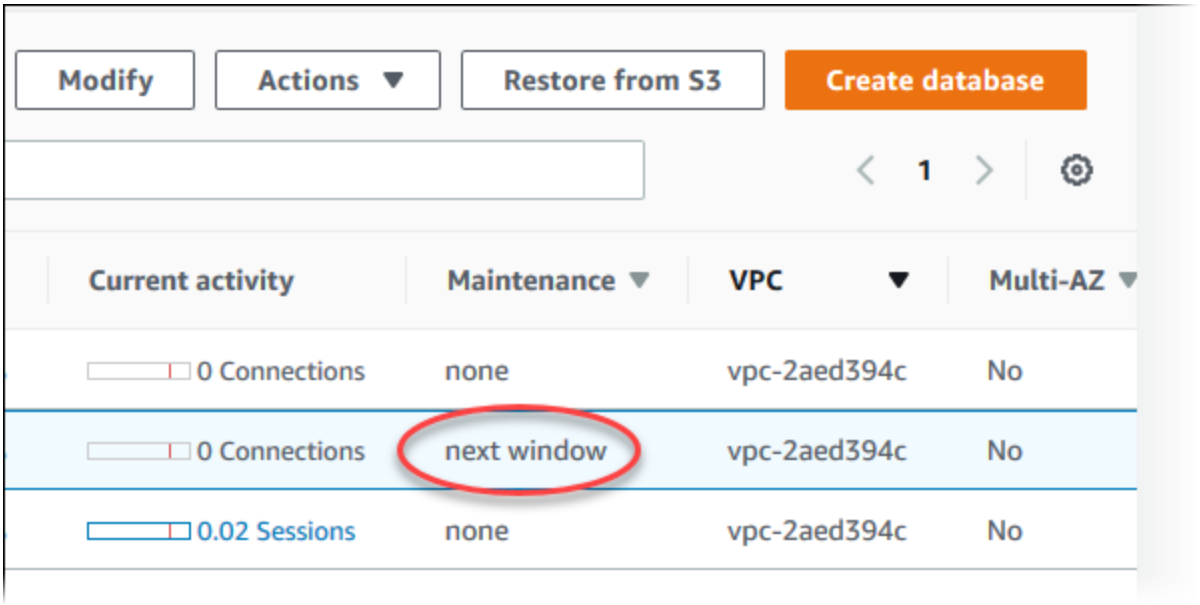
L'eventuale coerenza può influire sul modo in cui hai gestito gli aggiornamenti di manutenzione. Ad esempio, se si esegue il `ApplyPendingMaintenanceActions` comando per aggiornare la versione del motore di database per un cluster DB, alla fine sarà visibile a`DescribePendingMaintenanceActions`. In questo scenario, `DescribePendingMaintenanceActions` potrebbe mostrare che l'azione di manutenzione non è stata applicata anche se lo era.

Per gestire l'eventuale coerenza, puoi fare quanto segue:

- Conferma lo stato del cluster DB prima di eseguire un comando per modificarlo. Esegui il `DescribePendingMaintenanceActions` comando appropriato utilizzando un algoritmo di backoff esponenziale per assicurarti di concedere tempo sufficiente per la propagazione del comando precedente nel sistema. A tale scopo, esegui il `DescribePendingMaintenanceActions` comando ripetutamente, iniziando con un paio di secondi di attesa e aumentando gradualmente fino a cinque minuti di attesa.
- Aggiunge il tempo di attesa tra i comandi successivi, anche se un `DescribePendingMaintenanceActions` comando restituisce una risposta accurata. Applica un algoritmo di backoff esponenziale a partire da un paio di secondi di attesa e aumenta gradualmente fino a circa cinque minuti di attesa.

Visualizzazione degli aggiornamenti di manutenzione in sospeso

Verifica se è disponibile un aggiornamento di manutenzione per il tuo cluster di DB utilizzando la console RDS, l'API RDS o l' AWS CLI API RDS. Se è disponibile un aggiornamento, è indicato nella colonna Maintenance per il cluster di DB sulla console Amazon RDS, come mostrato in questa figura.



Se non è disponibile alcun aggiornamento di manutenzione per il cluster di un' database, il valore della colonna corrispondente è none (nessuno).

Se è disponibile un aggiornamento di manutenzione per il cluster di un' database, la colonna può avere i seguenti valori:

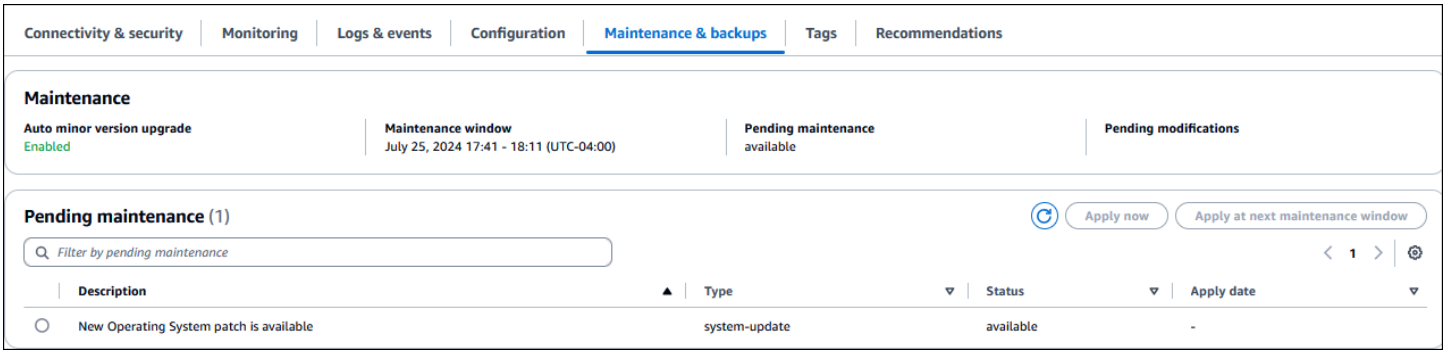
- richiesto – L'operazione di manutenzione sarà applicata alla risorsa e non può essere a tempo indeterminato.
- available (disponibile) – L'operazione di manutenzione è disponibile ma non sarà automaticamente applicata alla risorsa. Puoi applicarla manualmente.
- next window (finestra successiva) – L'operazione di manutenzione sarà applicata alla risorsa durante la finestra di manutenzione successiva.
- In corso: l'azione di manutenzione viene applicata alla risorsa.

Se è disponibile un aggiornamento, puoi scegliere una di queste operazioni:

- Se il valore di manutenzione è nella finestra successiva, rimanda le azioni di manutenzione scegliendo Rinvia aggiornamento da Azioni. Non è possibile posticipare un'azione di manutenzione già iniziata.
- Applicare immediatamente le operazioni di manutenzione.
- Applica le azioni di manutenzione durante la finestra di manutenzione successiva.
- Non eseguire alcuna operazione.

Per eseguire un'azione utilizzando il AWS Management Console

1. Scegli l'istanza o il cluster DB per mostrarne i dettagli.
2. Scegliere Maintenance & backups (Manutenzione e backup). Vengono visualizzate le azioni di manutenzione in sospeso.
3. Scegli l'azione da intraprendere, quindi scegli quando applicarla.



La finestra di manutenzione determina l'avvio delle operazioni in sospeso, ma non limita il tempo di esecuzione totale di tali operazioni. Non è garantito che le operazioni di manutenzione terminino prima della fine della finestra di manutenzione e potrebbero continuare oltre l'ora di fine specificata. Per ulteriori informazioni, consulta Finestra di manutenzione di Amazon RDS.

È inoltre possibile verificare se è disponibile un aggiornamento di manutenzione per il cluster di DB eseguendo il describe-pending-maintenance-actions AWS CLI comando.

Per informazioni sull'applicazione degli aggiornamenti di manutenzione, consultaApplicazione degli aggiornamenti a un cluster di DB.

Azioni di manutenzione per Amazon Aurora

Le seguenti azioni di manutenzione si applicano ai cluster Aurora DB:

- `os-upgrade`— Aggiorna i sistemi operativi di tutte le istanze DB nel cluster DB, utilizzando aggiornamenti continui. Per ulteriori informazioni, consulta the section called “Aggiornamenti del sistema operativo”.
- `system-update`— Applicare una patch al motore DB per Aurora PostgreSQL.

Le seguenti azioni di manutenzione si applicano alle istanze Aurora DB:

- `ca-certificate-rotation`— Aggiorna il certificato Amazon RDS Certificate Authority per l'istanza DB.
- `hardware-maintenance`— Esegue la manutenzione sull'hardware sottostante per l'istanza DB.
- `system-update`— Aggiornare il sistema operativo per l'istanza DB.

Scelta della frequenza degli aggiornamenti di manutenzione di Aurora MySQL

Puoi scegliere con quale frequenza eseguire gli aggiornamenti di Aurora MySQL per ogni cluster di database. La scelta migliore dipende dall'utilizzo di Aurora MySQL e dalle priorità delle applicazioni eseguite in Aurora. Per informazioni sulle versioni con supporto a lungo termine (LTS) di Aurora MySQL che richiedono aggiornamenti meno frequenti, consultare Aurora I miei rilasci di supporto SQL a lungo termine () LTS.

Puoi scegliere di aggiornare un cluster Aurora MySQL più raramente se alcune o tutte le condizioni seguenti sono applicabili al tuo scenario:

- Il ciclo di test per l'applicazione richiede molto tempo per ogni aggiornamento al motore di database di Aurora MySQL.
- Hai molti cluster di database o molte applicazioni, tutti in esecuzione sulla stessa versione di Aurora MySQL. Preferisci aggiornare tutti i cluster di database e le applicazioni associate contemporaneamente.
- Utilizzi Aurora MySQL e RDS per MySQL e preferisci mantenere i cluster Aurora MySQL e le istanze database RDS per MySQL compatibili con lo stesso livello di MySQL.
- L'applicazione Aurora MySQL è in produzione o è comunque una risorsa business-critical. Non puoi avere tempi di inattività dovuti agli aggiornamenti, ad eccezione di rari casi per la distribuzione di patch critiche.

- L'applicazione Aurora MySQL non è limitata da problemi di prestazioni o da carenze nella funzionalità risolti nelle versioni di Aurora MySQL successive.

Se i fattori precedenti si applicano alla tua situazione, puoi limitare il numero di aggiornamenti forzati per un cluster di database Aurora MySQL. Per farlo, scegli una versione di Aurora MySQL specifica, nota come versione con supporto a lungo termine ("Long-Term Support", LTS) quando crei o aggiorni il cluster di database. In questo modo, riduci il numero di cicli di aggiornamenti, cicli di test e interruzioni correlate agli aggiornamenti per lo specifico cluster di database.

Puoi scegliere di aggiornare un cluster Aurora MySQL più spesso, se alcune o tutte le condizioni seguenti sono applicabili al tuo scenario:

- Il ciclo di test per l'applicazione è semplice e breve.
- L'applicazione è ancora in fase di sviluppo.
- L'ambiente del database utilizza diverse versioni di Aurora MySQL oppure versioni di Aurora MySQL e di RDS for MySQL. Ogni cluster Aurora MySQL ha il proprio ciclo di aggiornamento.
- Sei in attesa di determinati miglioramenti alle prestazioni o alle funzionalità per poter aumentare il tuo utilizzo di Aurora MySQL.

Se i fattori precedenti si applicano alla tua situazione, puoi abilitare Aurora per eseguire più spesso gli aggiornamenti importanti. Per farlo, esegui l'aggiornamento di un cluster database Aurora MySQL a una versione di Aurora MySQL più recente rispetto alla versione LTS. In questo modo, i miglioramenti alle prestazioni, le correzioni di bug e le funzionalità più recenti vengono resi disponibili più rapidamente.

Finestra di manutenzione di Amazon RDS

La finestra di manutenzione è un intervallo di tempo settimanale durante il quale vengono applicate eventuali modifiche al sistema. Ogni cluster di DB ha una finestra di manutenzione settimanale. La finestra di manutenzione è un'opportunità per controllare quando vengono apportate modifiche e patch al software. Per ulteriori informazioni sulla regolazione della finestra di manutenzione, vedere.

Durante l'esecuzione delle attività di manutenzione, RDS utilizza alcune risorse del cluster database. Ciò potrebbe influire, in modo minimo, sulle prestazioni. Per un'istanza database, in rari casi, potrebbe essere necessario un failover Multi-AZ per il completamento di un aggiornamento di manutenzione.

Se un evento di manutenzione è pianificato per una settimana specifica, viene avviato durante la finestra di manutenzione di 30 minuti indicata. La maggior parte degli eventi di manutenzione viene completata durante la finestra di manutenzione di 30 minuti, tuttavia l'esecuzione degli eventi di manutenzione di dimensioni maggiori può richiedere più di 30 minuti per il completamento. La finestra di manutenzione viene messa in pausa quando il cluster di DB viene arrestato.

La finestra di manutenzione di 30 minuti è selezionata a caso da un blocco di tempo di 8 ore per regione. Se non specifichi una finestra di manutenzione quando crei il cluster database, Amazon RDS assegna una finestra di manutenzione di 30 minuti in un giorno della settimana selezionato in modo casuale.

La tabella seguente mostra i blocchi temporali per ciascuno Regione AWS dei quali vengono assegnate le finestre di manutenzione predefinite.

Nome della regione	Regione	Periodo di tempo
US East (N. Virginia)	us-east-1	03:00 - 11:00 UTC
Stati Uniti orientali (Ohio)	us-east-2	03:00 - 11:00 UTC
US West (N. Californi a)	us-west-1	06:00 - 14:00 UTC
US West (Oregon)	us-west-2	06:00 - 14:00 UTC
Africa (Cape Town)	af-south-1	03:00 - 11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	06:00 - 14:00 UTC
Asia Pacific (Hyderabad)	ap-south-2	06:30 - 14:30 UTC
Asia Pacifico (Giacarta)	ap-southeast-3	08:00–16:00 UTC
Asia Pacifico (Malesia)	ap-southeast-5	09:00 — 17:00 UTC

Nome della regione	Regione	Periodo di tempo
Asia Pacifico (Melbourne)	ap-southeast-4	11:00 - 19:00 UTC
Asia Pacific (Mumbai)	ap-south-1	06:00 - 14:00 UTC
Asia Pacific (Osaka)	ap-northeast-3	22:00 - 23:59 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00 - 21:00 UTC
Asia Pacific (Singapore)	ap-southeast-1	14:00 - 22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	12:00 - 20:00 UTC
Asia Pacific (Tokyo)	ap-northeast-1	13:00 - 21:00 UTC
Canada (Central)	ca-central-1	03:00 - 11:00 UTC
Canada occidentale (Calgary)	ca-west-1	18:00 - 02:00 UTC
China (Beijing)	cn-north-1	06:00 - 14:00 UTC
China (Ningxia)	cn-northwest-1	06:00 - 14:00 UTC
Europe (Frankfurt)	eu-central-1	21:00 - 05:00 UTC
Europe (Ireland)	eu-west-1	22:00 - 06:00 UTC
Europe (London)	eu-west-2	22:00 - 06:00 UTC
Europa (Milano)	eu-south-1	02:00 - 10:00 UTC
Europe (Paris)	eu-west-3	23:59 - 07:29 UTC
Europa (Spagna)	eu-south-2	02:00 - 10:00 UTC
Europe (Stockholm)	eu-north-1	23:00 - 07:00 UTC
Europa (Zurigo)	eu-central-2	02:00 - 10:00 UTC

Nome della regione	Regione	Periodo di tempo
Israele (Tel Aviv)	il-central-1	03:00 - 11:00 UTC
Medio Oriente (Bahrein)	me-south-1	06:00 - 14:00 UTC
Medio Oriente (Emirati Arabi Uniti)	me-central-1	05:00–13:00 UTC
Sud America (São Paulo)	sa-east-1	00:00 - 08:00 UTC
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	17:00 - 01:00 UTC
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	06:00 - 14:00 UTC

Argomenti

- Impostazione della finestra di manutenzione preferita del cluster database

Impostazione della finestra di manutenzione preferita del cluster database

La finestra di manutenzione del cluster database Aurora deve essere impostata nel periodo di minore utilizzo e quindi potrebbe essere necessario apportare modifiche di tanto in tanto. Durante questo intervallo di tempo il cluster database non è disponibile solo se gli aggiornamenti applicati richiedono un'interruzione. L'interruzione dura il minimo necessario per apportare gli aggiornamenti richiesti.

> **Note**
>
> Per gli aggiornamenti al motore di database, Amazon Aurora gestisce la finestra di manutenzione preferita per un cluster database e non per le singole istanze.

Console

Per impostare la finestra di manutenzione preferita del cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster database per cui si desidera modificare la finestra di manutenzione.
4. Scegliere Modify (Modifica).
5. Nella sezione Maintenance (Manutenzione) aggiornare a finestra di manutenzione.
6. Scegli Continue (Continua).

   Nella pagina di conferma esaminare le modifiche.
7. Per applicare immediatamente le modifiche alla finestra di manutenzione, scegli Immediately (Immediatamente) nella sezione Schedule of modifications (Pianificazione delle modifiche).
8. Scegliere Modify cluster (Modifica cluster) per salvare le modifiche.

   In alternativa, scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per regolare la finestra di manutenzione preferita del cluster DB, usa il AWS CLI `modify-db-cluster`comando con i seguenti parametri:

- `--db-cluster-identifier`
- `--preferred-maintenance-window`

Example

Nell'esempio di codice seguente la finestra di manutenzione viene impostata su martedì dalle 4:00 – alle 4:30.

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
--db-cluster-identifier my-cluster \
```

```
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Per Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier my-cluster ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API RDS

Per impostare la finestra di manutenzione del cluster database preferita, utilizzare l'operazione API Amazon RDS `ModifyDBCluster` con i parametri seguenti:

- `DBClusterIdentifier`
- `PreferredMaintenanceWindow`

Applicazione degli aggiornamenti a un cluster di DB

Con Amazon RDS, puoi scegliere quando eseguire le operazioni di manutenzione. Puoi decidere quando Amazon RDS applicare gli aggiornamenti utilizzando l'API AWS Management Console AWS CLI, o RDS.

Console

Per gestire un aggiornamento per un cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster dell' database da aggiornare.
4. Per Actions (Operazioni), scegliere una delle seguenti opzioni:

   - Aggiorna ora
   - Patch nella finestra successiva

   > **Note**
   >
   > Se scegli Patch nella finestra successiva e in seguito desideri ritardare l'aggiornamento, puoi scegliere Refer upgrade. Non puoi rinviare un'azione di manutenzione se è già stata avviata.
   > Per annullare un'azione di manutenzione, modificare l'istanza DB e disabilitare l'aggiornamento automatico della versione minore.

AWS CLI

Per applicare un aggiornamento in sospeso a un cluster di DB, usa il apply-pending-maintenance-action AWS CLI comando.

Example

Per LinuxmacOS, oUnix:

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
```

```
    --apply-action system-update \
    --opt-in-type immediate
```

Per Windows:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> Per rinviare un'azione di manutenzione, specificare `undo-opt-in` per `--opt-in-type`. Non è possibile specificare `undo-opt-in` per `--opt-in-type` se l'azione di manutenzione è già stata avviata.
> Per annullare un'azione di manutenzione, esegui il modify-db-instance AWS CLI comando e specifica`--no-auto-minor-version-upgrade`.

Per restituire un elenco di risorse con almeno un aggiornamento in sospeso, usa il describe-pending-maintenance-actions AWS CLI comando.

Example

Per LinuxmacOS, oUnix:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Per Windows:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

È inoltre possibile restituire un elenco di risorse per un cluster di DB specificando il `--filters` parametro del `describe-pending-maintenance-actions` AWS CLI comando. Il formato del comando `--filters` è `Name=`*`filter-name`*`,Value=`*`resource-id`*`,...`.

Di seguito sono indicati i valori accettati per il parametro `Name` di un filtro:

- `db-instance-id`— Accetta un elenco di identificatori di istanze DB o Amazon Resource Names (ARNs). L'elenco restituito include solo le azioni di manutenzione in sospeso per le istanze DB identificate da questi identificatori o. ARNs
- `db-cluster-id`— Accetta un elenco di identificatori di cluster DB o ARNs per Amazon Aurora. L'elenco restituito include solo le azioni di manutenzione in sospeso per i cluster DB identificati da questi identificatori o. ARNs

L'esempio seguente restituisce le operazioni di manutenzione in sospeso per i cluster database `sample-cluster1` e `sample-cluster2`.

Example

Per LinuxmacOS, oUnix:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Per Windows:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

API RDS

Per applicare un aggiornamento a un cluster database, chiamare l'operazione `ApplyPendingMaintenanceAction` dell'API Amazon RDS.

Per ottenere un elenco delle risorse con almeno un aggiornamento in sospeso, chiamare l'operazione API Amazon RDS `DescribePendingMaintenanceActions`.

Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora

L'impostazione Aggiornamento automatico versione secondaria specifica se gli aggiornamenti Aurora vengono applicati automaticamente al cluster database. Questi aggiornamenti includono nuove versioni secondarie contenenti funzionalità aggiuntive e patch con correzioni di bug.

Gli aggiornamenti automatici delle versioni secondarie aggiornano periodicamente il database alle versioni recenti del motore di database. Tuttavia, l'aggiornamento potrebbe non includere

sempre la versione più recente del motore di database. Se è necessario mantenere i database su versioni specifiche in momenti particolari, si consiglia di eseguire manualmente l'aggiornamento alle versioni del database necessarie in base alla pianificazione richiesta. In caso di problemi di sicurezza critici o quando una versione raggiunge la end-of-support data di scadenza, Amazon Aurora potrebbe applicare un aggiornamento della versione secondaria anche se non hai abilitato l'opzione Aggiornamento automatico della versione secondaria. Per ulteriori informazioni, consulta la documentazione di aggiornamento per il tuo motore di database specifico.

Consulta Aggiornamento della versione secondaria o del livello di patch di un cluster Aurora My DB SQL e Esecuzione di un aggiornamento di versione minore.

> **Note**
>
> Aurora Global Database non supporta gli aggiornamenti automatici delle versioni secondarie.

Questa impostazione è attivata per impostazione predefinita. Per ogni nuovo cluster database, scegliere il valore appropriato per questa impostazione. Il valore si basa sull'importanza, sulla durata prevista e sulla quantità di test di verifica eseguiti dopo ogni aggiornamento.

Per istruzioni su come attivare o disattivare l'impostazione Aggiornamento automatico versione secondaria, consulta quanto segue:

- Abilitazione degli aggiornamenti automatici delle versioni secondarie per un cluster database Aurora
- Abilitazione degli aggiornamenti automatici delle versioni secondarie per singole istanze database in un cluster database Aurora

> **Important**
>
> Si consiglia vivamente di applicare questa impostazione ai cluster database nuovi ed esistenti e non alle singole istanze database nel cluster. Se un'istanza database nel cluster ha questa impostazione disattivata, il cluster database non viene aggiornato automaticamente.

La tabella seguente mostra come funziona l'impostazione Aggiornamento automatico versione secondaria quando viene applicata a livello di cluster e istanza.

Azione	Impostazioni a livello di cluster	Impostazioni a livello di istanza	Il cluster è stato aggiornato automatic amente?
Impostazione su Vero a livello di cluster database.	True	Impostazione su Vero per tutte le istanze nuove ed esistenti	Sì
Impostazione su Falso a livello di cluster database.	False	Impostazione su Falso per tutte le istanze nuove ed esistenti	No
Precedente impostazione su Vero a livello di cluster database. Impostazione su Falso in almeno un'istanza database.	Impostazione modificata su Falso	Impostazione su Falso per una o più istanze	No
Precedente impostazione su Falso a livello di cluster database. Impostazione su Vero in almeno un'istanza database, ma non in tutte le istanze.	False	Impostazione su Vero in una o più istanze, ma non in tutte le istanze	No
Precedente impostazione su Falso a livello di cluster database. Impostazione su Vero in tutte le istanze database.	Impostazione modificata su Vero	Impostazione su Vero per tutte le istanze	Sì

Gli aggiornamenti automatici delle versioni secondarie vengono comunicati in anticipo tramite un evento cluster database Amazon RDS con una categoria di `maintenance` e un ID di `RDS-EVENT-0156`. Per ulteriori informazioni, consulta Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora.

Gli aggiornamenti automatici vengono eseguiti durante le finestre di manutenzione. Se le singole istanze database nel cluster database hanno finestre di manutenzione diverse dalla finestra di manutenzione del cluster, la finestra di manutenzione del cluster ha la precedenza.

Per ulteriori informazioni sugli aggiornamenti del motore per Aurora PostgreSQL, consultare Aggiornamenti del motore di database per Amazon Aurora Postgre SQL.

Per ulteriori informazioni sull'impostazione di Auto Minor Version Upgrade (Aggiornamento automatico minore della versione) per Aurora MySQL, consulta Abilitazione degli aggiornamenti automatici tra versioni minori di Aurora My SQL. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consulta Aggiornamenti del motore di database per Amazon Aurora My SQL.

Abilitazione degli aggiornamenti automatici delle versioni secondarie per un cluster database Aurora

Eseguire la procedura generale descritta in Modifica del cluster di database tramite la console, la CLI e l'API.

Console

Nella pagina Modifica il cluster DB, nella sezione Manutenzione selezionare la casella di controllo Abilita aggiornamento automatico versione secondaria.

AWS CLI

Chiamare il comando modify-db-cluster AWS CLI . Specificare il nome del cluster database per l'opzione `--db-cluster-identifier` e `true` per l'opzione `--auto-minor-version-upgrade`. Facoltativamente, specificare l'opzione `--apply-immediately` per attivare immediatamente questa impostazione per il cluster database.

API RDS

Richiama l'operazione Modify DBCluster API e specifica il nome del cluster DB per il `DBClusterIdentifier` parametro e `true` per il `AutoMinorVersionUpgrade` parametro. Facoltativamente, impostare il parametro `ApplyImmediately` su `true` per attivare immediatamente questa impostazione per il cluster database.

Abilitazione degli aggiornamenti automatici delle versioni secondarie per singole istanze database in un cluster database Aurora

Eseguire la procedura generale descritta in Modifica di un'istanza database in un cluster database.

Console

Nella pagina Modifica istanza database, nella sezione Manutenzione selezionare la casella di controllo Abilita aggiornamento automatico versione secondaria.

AWS CLI

Chiamare il comando modify-db-instance AWS CLI . Specifica il nome dell'istanza database per l'opzione `--db-instance-identifier` e `true` per l'opzione `--auto-minor-version-upgrade`. Facoltativamente, specifica l'opzione `--apply-immediately` per attivare immediatamente questa impostazione per l'istanza database. Esegui un comando `modify-db-instance` separato per ogni istanza database presente nel cluster.

API RDS

Chiama l'operazione Modify DBInstance API e specifica il nome del cluster DB per il `DBInstanceIdentifier` parametro e `true` per il `AutoMinorVersionUpgrade` parametro. Facoltativamente, imposta il parametro `ApplyImmediately` su `true` per attivare immediatamente questa impostazione per l'istanza database. Chiama un'operazione `ModifyDBInstance` separata per ogni istanza database presente nel cluster.

È possibile utilizzare un comando CLI come il seguente per verificare lo stato dell'impostazione `AutoMinorVersionUpgrade` per tutte le istanze database nei cluster Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Questo comando genera un output simile al seguente:

```
[
  {
      "DBInstanceIdentifier": "db-writer-instance",
      "DBClusterIdentifier": "my-db-cluster-57",
```

```
        "AutoMinorVersionUpgrade": true
    },
    {
        "DBInstanceIdentifier": "db-reader-instance1",
        "DBClusterIdentifier": "my-db-cluster-57",
        "AutoMinorVersionUpgrade": false
    },
    {
        "DBInstanceIdentifier": "db-writer-instance2",
        "DBClusterIdentifier": "my-db-cluster-80",
        "AutoMinorVersionUpgrade": true
    },
... output omitted ...
```

In questo esempio, l'impostazione Abilita aggiornamento automatico versione secondaria è disattivata per il cluster di database `my-db-cluster-57`, perché è disattivata per una delle istanze database del cluster.

Aggiornamenti del sistema operativo per i cluster Aurora DB

Le istanze database nei cluster database Aurora MySQL e Aurora PostgreSQL richiedono occasionalmente aggiornamenti del sistema operativo. Amazon RDS aggiorna il sistema operativo a una versione più recente per migliorare le prestazioni del database e la posizione di sicurezza generale dei clienti. In genere, gli aggiornamenti richiedono circa 10 minuti. Gli aggiornamenti del sistema operativo non modificano la versione del motore database o la classe istanza database di un'istanza database.

Esistono due tipi di aggiornamenti del sistema operativo, differenziati in base alla descrizione dell'azione di manutenzione in sospeso:

- Aggiornamento della distribuzione del sistema operativo: utilizzato per eseguire la migrazione alla versione principale supportata più recente di Amazon Linux. La sua descrizione è. `New Operating System upgrade is available`
- Patch del sistema operativo: utilizzata per applicare varie correzioni di sicurezza e talvolta per migliorare le prestazioni del database. La sua descrizione è`New Operating System patch is available`.

Gli aggiornamenti del sistema operativo possono essere facoltativi o obbligatori:

- Un aggiornamento facoltativo può essere applicato in qualsiasi momento. Sebbene questi aggiornamenti siano facoltativi, ti consigliamo di applicarli periodicamente per mantenere aggiornato il parco istanze RDS. RDS non applica automaticamente questi aggiornamenti.

  Per ricevere una notifica quando diventa disponibile una nuova patch facoltativa del sistema operativo, è possibile iscriversi a RDS-EVENT-0230 nella categoria degli eventi di applicazione delle patch di sicurezza. Per informazioni sulla sottoscrizione agli eventi RDS, consulta Iscrizione alla notifica RDS degli eventi di Amazon.

  > **Note**
  >
  > `RDS-EVENT-0230` non si applica agli aggiornamenti di distribuzione del sistema operativo.

- È richiesto un aggiornamento obbligatorio. Viene inviata una notifica prima dell'aggiornamento obbligatorio. La notifica potrebbe contenere una data di scadenza. Pianificare la programmazione dell'aggiornamento prima di questa data di scadenza. Dopo la data di scadenza specificata, Amazon RDS aggiorna automaticamente il sistema operativo per l'istanza database alla versione più recente durante una delle finestre di manutenzione assegnate.

  Gli aggiornamenti della distribuzione del sistema operativo sono obbligatori.

> **Note**
>
> Potrebbe essere necessario rimanere aggiornati su tutti gli aggiornamenti facoltativi e obbligatori per soddisfare vari obblighi di conformità. Si consiglia di applicare regolarmente tutti gli aggiornamenti resi disponibili da RDS durante le finestre di manutenzione.

Per i cluster Aurora DB, è possibile utilizzare l'opzione di manutenzione a livello di cluster per eseguire aggiornamenti del sistema operativo (OS). Trova l'opzione per eseguire aggiornamenti a livello di cluster nella scheda Manutenzione e backup quando selezioni il nome del tuo cluster DB nella console o usi il comando in. `os-upgrade` AWS CLI Questo metodo preserva la disponibilità di lettura con aggiornamenti continui che applicano automaticamente gli aggiornamenti a poche istanze DB di Reader alla volta. Per evitare failover multipli e ridurre i tempi di inattività non necessari, Aurora aggiorna l'istanza Writer DB per ultima.

Gli aggiornamenti del sistema operativo a livello di cluster avvengono durante la finestra di manutenzione specificata per il cluster. Ciò garantisce aggiornamenti coordinati in tutto il cluster.

Per la compatibilità con le versioni precedenti, Aurora mantiene anche l'opzione di manutenzione a livello di istanza. Tuttavia, ti consigliamo di utilizzare invece gli aggiornamenti a livello di cluster. Se devi utilizzare aggiornamenti a livello di istanza, aggiorna prima le istanze Reader DB in un cluster DB, quindi aggiorna l'istanza Writer DB. Se aggiorni le istanze Reader e Writer contemporaneamente, aumenti la possibilità di downtime dovuti al failover. Trova l'opzione per eseguire aggiornamenti a livello di istanza nella scheda Manutenzione e backup quando selezioni il nome dell'istanza DB nella console o utilizzi il comando in. `system-update` AWS CLI

Gli aggiornamenti del sistema operativo a livello di istanza avvengono durante la finestra di manutenzione specificata per ogni rispettiva istanza. Ad esempio, se un cluster e due istanze di lettura hanno orari di manutenzione diversi, un aggiornamento del sistema operativo a livello di cluster si allinea alla finestra di manutenzione del cluster.

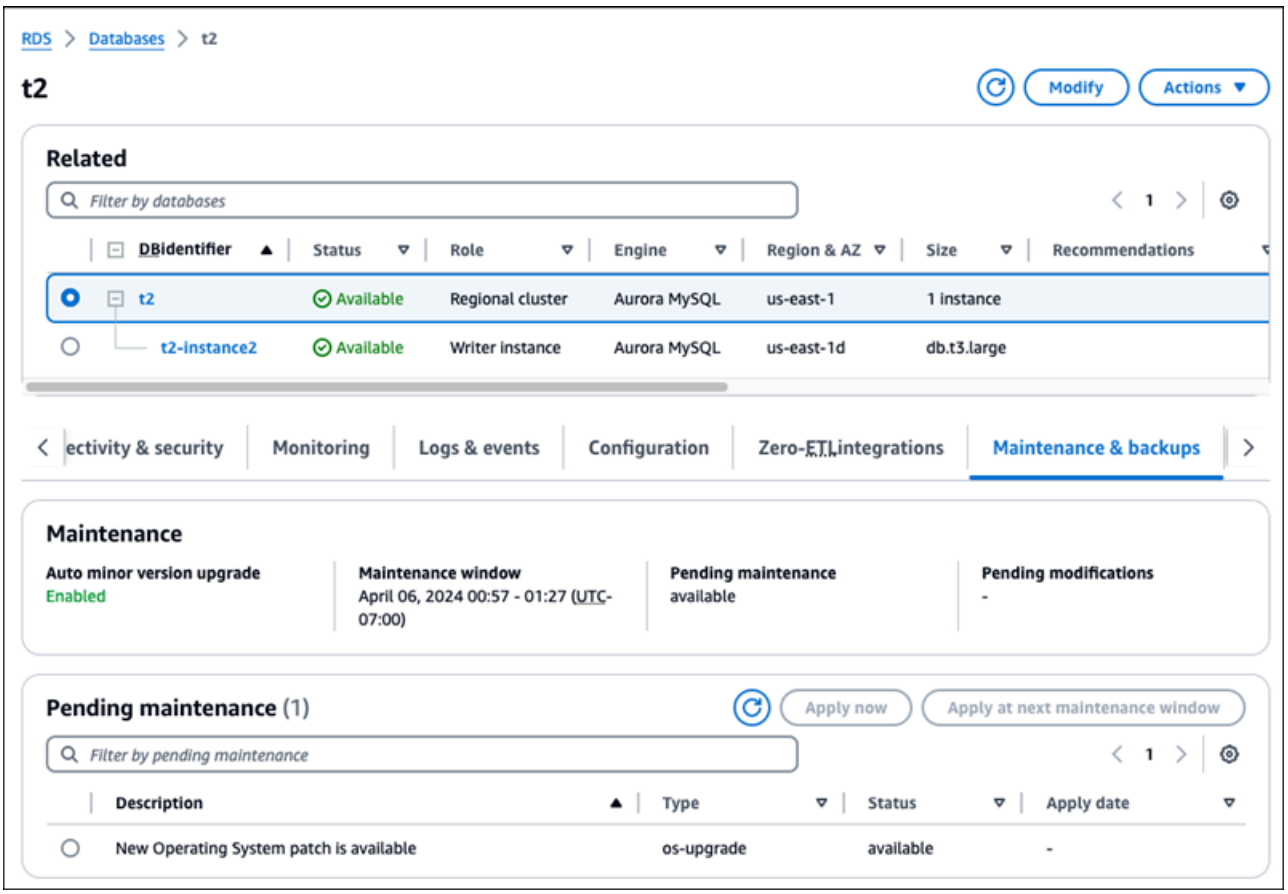
È possibile utilizzare AWS Management Console o the AWS CLI per ottenere informazioni sul tipo di aggiornamento del sistema operativo.

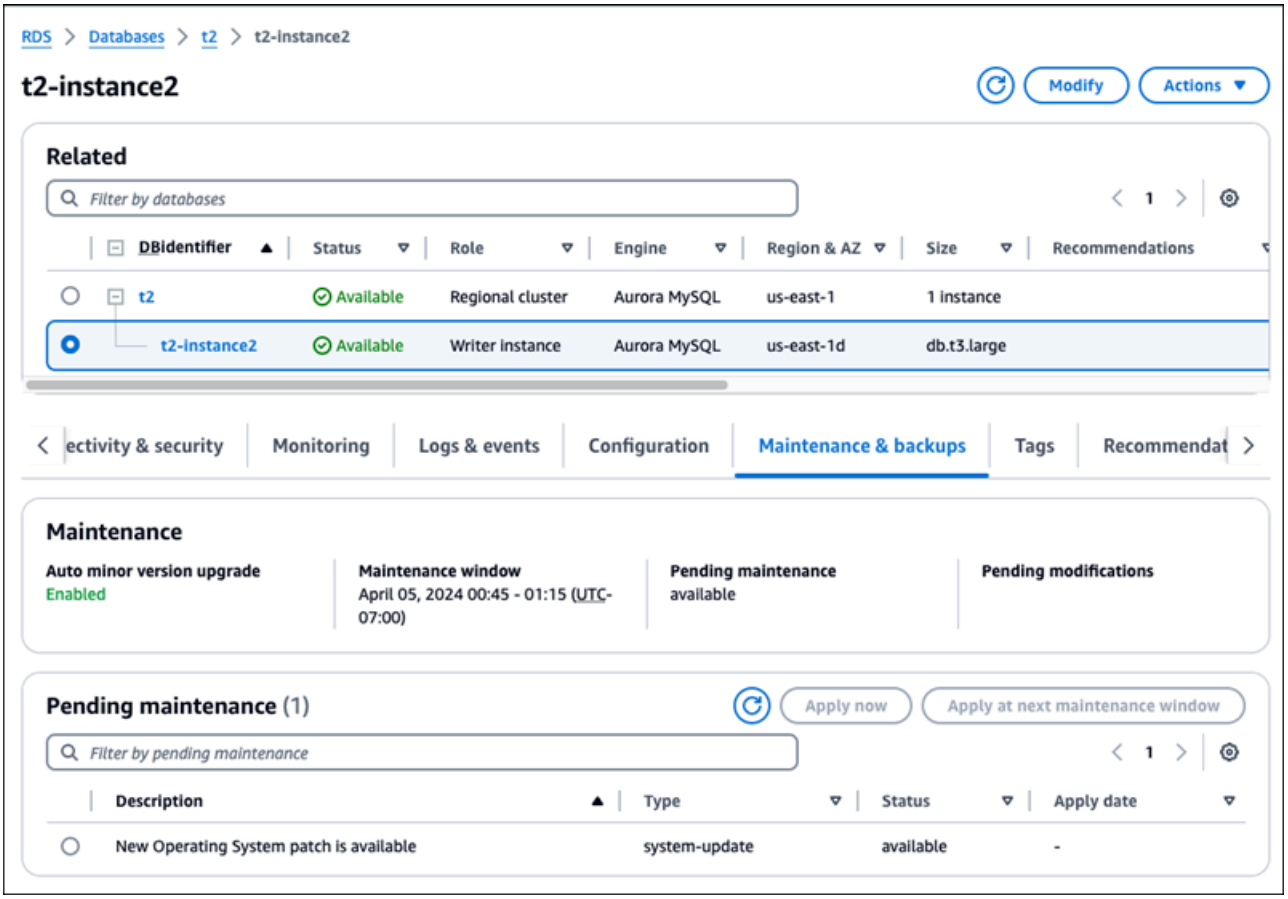
Console

Per ottenere informazioni di aggiornamento, utilizzare il AWS Management Console

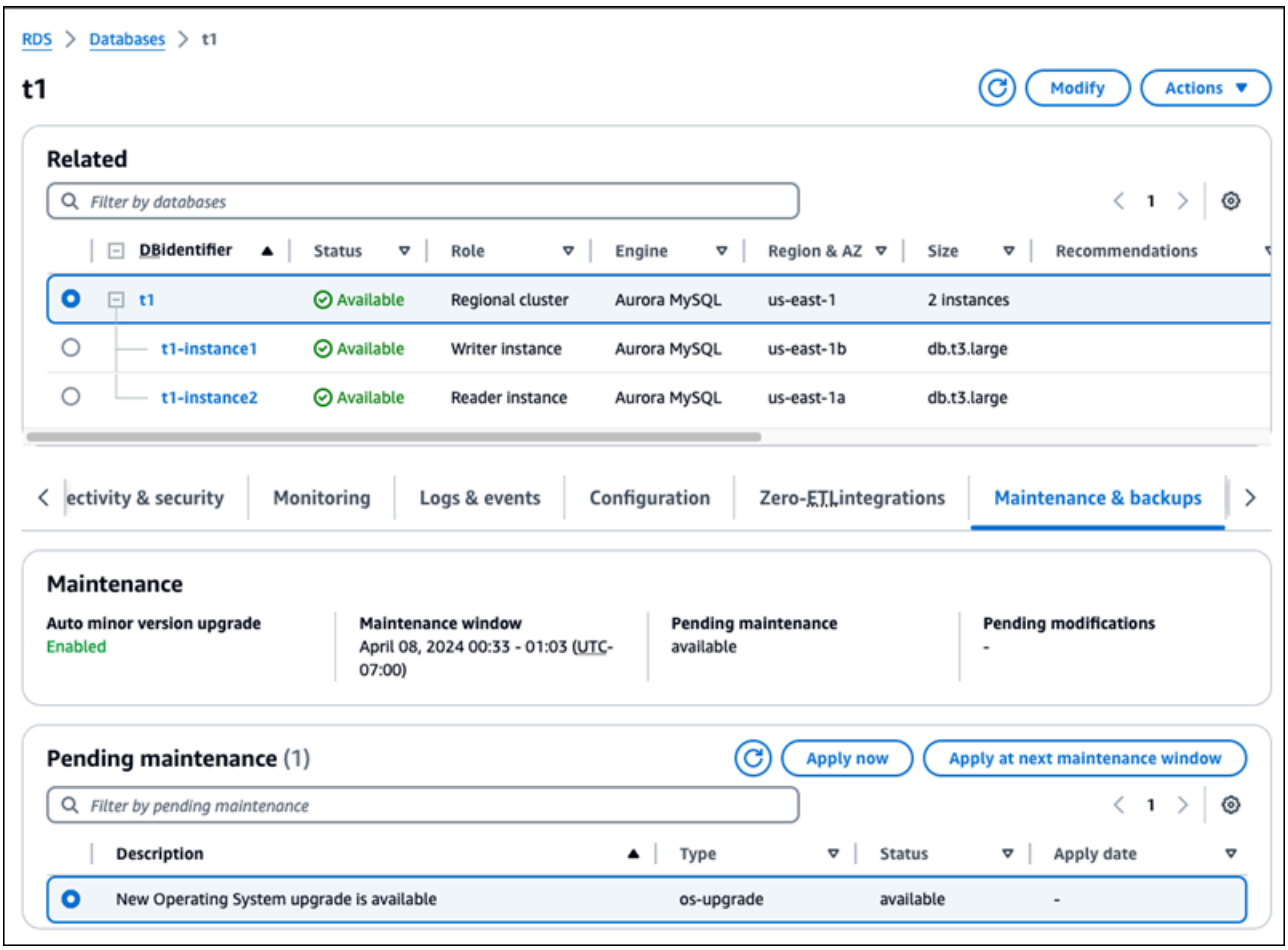
1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, seleziona Databases (Database), quindi scegli l'istanza database.
3. Scegliere Maintenance & backups (Manutenzione e backup).
4. Nella sezione In attesa di manutenzione, trova l'aggiornamento del sistema operativo e controlla il valore Descrizione.

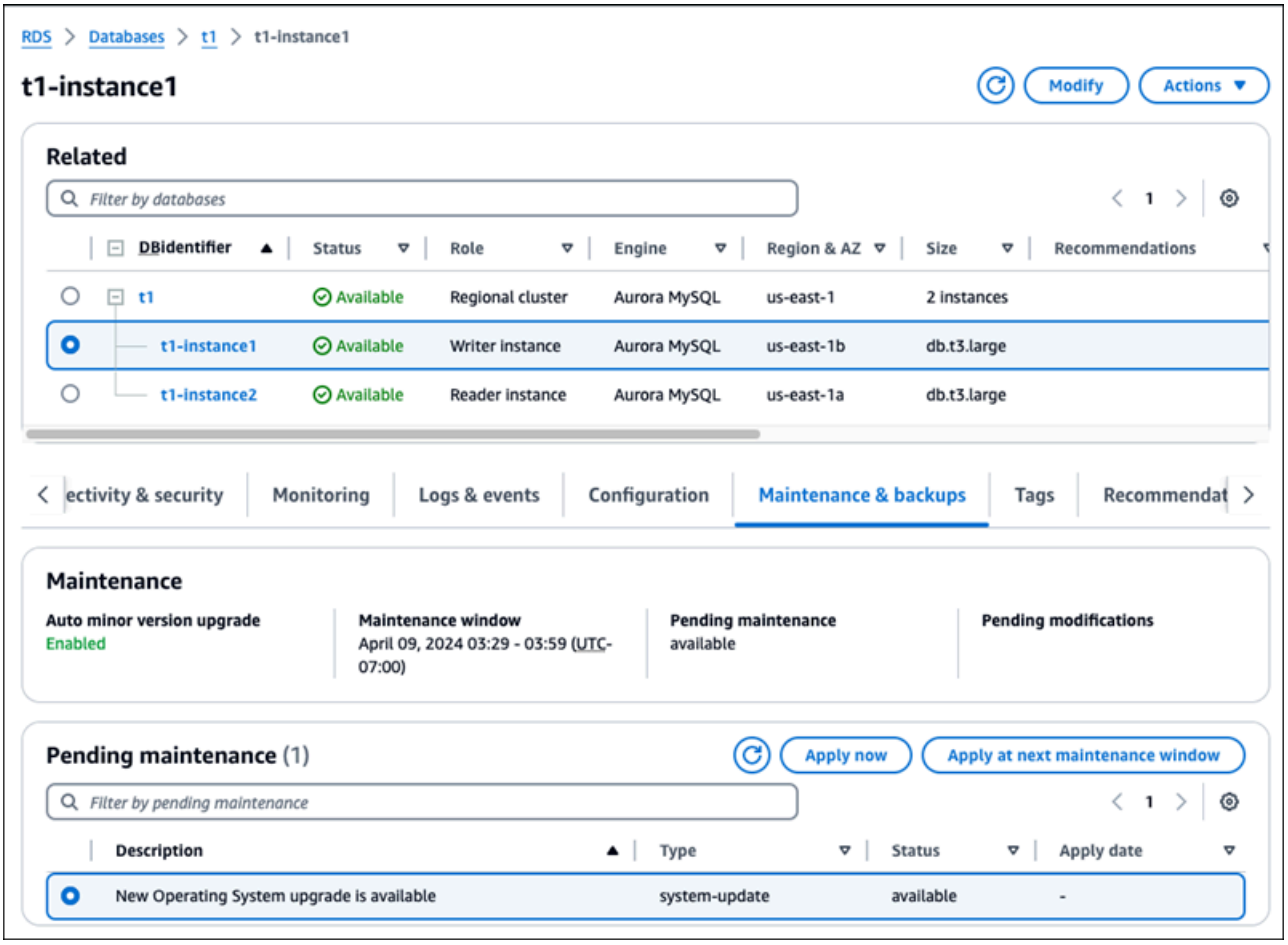
Le immagini seguenti mostrano un cluster DB con un'istanza Writer DB che dispone di una patch per il sistema operativo.

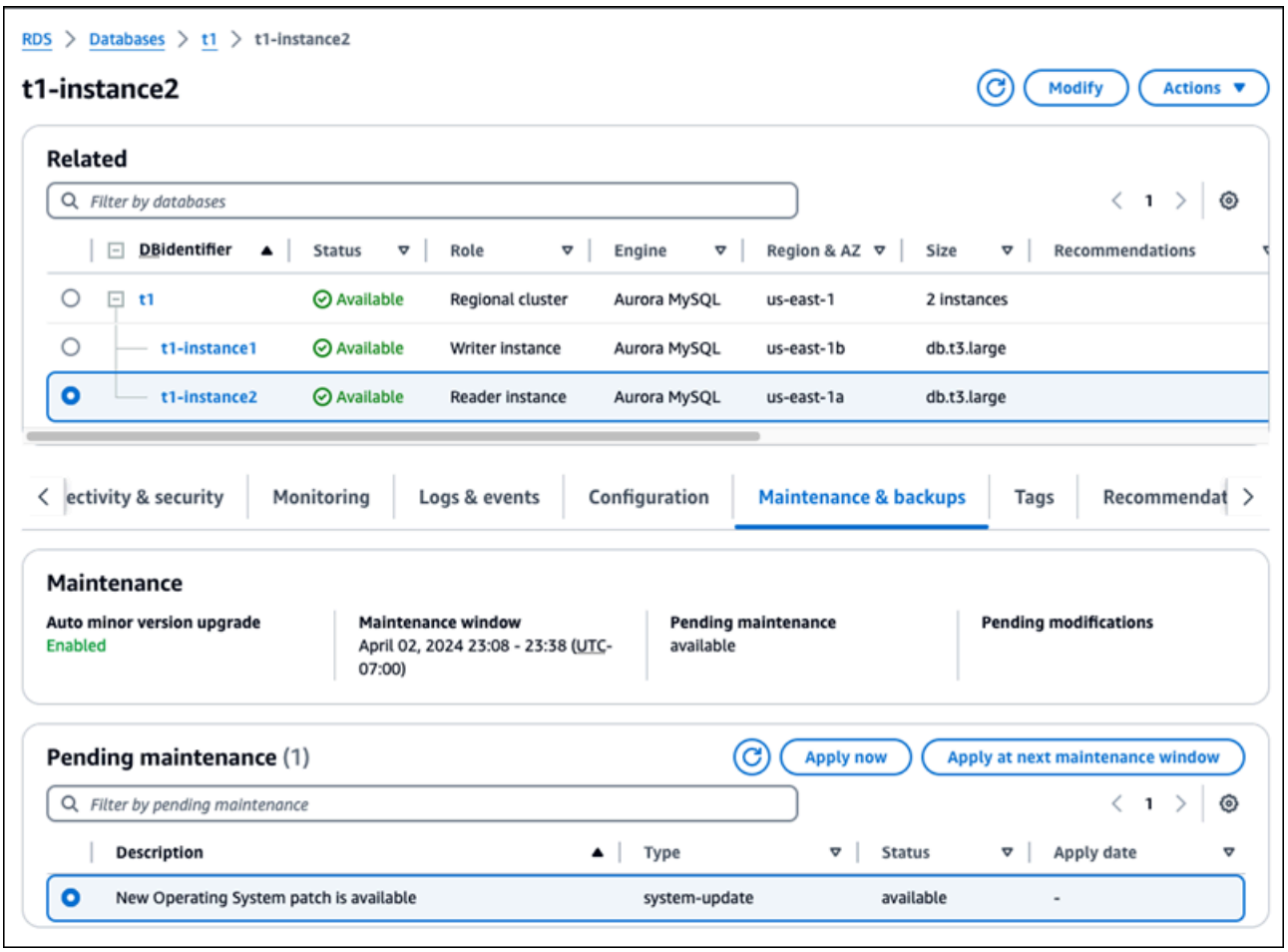

RDS > Databases > t2
t2
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t2
Available
Regional cluster
Aurora MySQL
us-east-1
1 instance
t2-instance2
Available
Writer instance
Aurora MySQL
us-east-1d
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 06, 2024 00:57 - 01:27 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System patch is available
os-upgrade
available
-



Le immagini seguenti mostrano un cluster DB con un'istanza DB writer e un'istanza DB reader. L'istanza writer dispone di un aggiornamento obbligatorio del sistema operativo. L'istanza del lettore dispone di una patch del sistema operativo.


RDS > Databases > t1
t1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 08, 2024 00:33 - 01:03 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
os-upgrade
available
-


RDS > Databases > t1 > t1-instance1
t1-instance1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
Recommendat
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 09, 2024 03:29 - 03:59 (UTC-07:00)
Pending maintenance
available
Pending modifications
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
system-update
available
-



AWS CLI

Per ottenere informazioni di aggiornamento da AWS CLI, utilizzare il describe-pending-maintenance-actionscomando.

```
aws rds describe-pending-maintenance-actions
```

L'output seguente mostra un aggiornamento della distribuzione del sistema operativo per un cluster DB e un'istanza DB.

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:cluster:t3",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "os-upgrade",
          "Description": "New Operating System upgrade is available"
```

```
        }
      ]
    },
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:t3-instance1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "Description": "New Operating System upgrade is available"
        }
      ]
    },
  ]
}
```

L'output seguente mostra una patch del sistema operativo per un'istanza DB.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System patch is available"
    }
  ]
}
```

Disponibilità di aggiornamenti del sistema operativo

Gli aggiornamenti del sistema operativo sono specifici per la versione del motore database e la classe istanza database. Pertanto, le istanze database ricevono o richiedono aggiornamenti in momenti diversi. Se è disponibile un aggiornamento del sistema operativo per l'istanza database basato sulla versione del motore e sulla classe di istanza, l'aggiornamento viene visualizzato nella console. Può essere visualizzato anche eseguendo il describe-pending-maintenance-actions AWS CLI comando o chiamando l'operazione API DescribePendingMaintenanceActionsRDS. Se è disponibile un aggiornamento per l'istanza, puoi aggiornare il sistema operativo seguendo le istruzioni in Applicazione degli aggiornamenti a un cluster di DB.

Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB

Potrebbe essere necessario riavviare il cluster DB o alcune istanze all'interno del cluster, di solito per motivi di manutenzione. Ad esempio, si supponga di modificare i parametri all'interno di un gruppo di parametri o di associare un gruppo di parametri diverso al cluster. In questi casi, è necessario riavviare il cluster affinché le modifiche abbiano effetto. Analogamente, è possibile riavviare una o più istanze DB di lettura all'interno del cluster. È possibile organizzare le operazioni di riavvio per singole istanze per ridurre al minimo i tempi di inattività per l'intero cluster.

Il tempo necessario per riavviare ogni istanza DB nel cluster dipende dall'attività del database al momento del riavvio. Dipende anche dal processo di ripristino dello specifico motore di DB. Se è pratico, ridurre l'attività del database su quella particolare istanza prima di avviare il processo di riavvio. In questo modo è possibile ridurre il tempo necessario per riavviare il database.

È possibile riavviare ogni istanza DB nel cluster solo quando è nello stato disponibile. Un'istanza database può non essere disponibile per vari motivi. Questi includono lo stato di arresto del cluster, una modifica applicata all'istanza e un'azione della finestra di manutenzione, ad esempio un aggiornamento della versione.

Il riavvio di un'istanza database comporta il riavvio del processo del motore di database. Il riavvio di un'istanza database comporta un'interruzione temporanea, durante la quale lo stato dell'istanza database viene impostato su rebooting (riavvio in corso).

> **Note**
>
> Se un'istanza DB non utilizza le ultime modifiche al gruppo di parametri DB associato, AWS Management Console mostra il gruppo di parametri DB con lo stato di riavvio in sospeso. Lo stato dei gruppi di parametro pending-reboot non prevede un riavvio automatico nel corso della prossima finestra di manutenzione. Per applicare le ultime modifiche del parametro su quella istanza database, riavviare manualmente l’istanza database. Per ulteriori informazioni sui gruppi di parametri, consultare .

Argomenti

- Riavvio di un'istanza database in un cluster Aurora
- Riavvio di un cluster Aurora con disponibilità di lettura

- Riavvio di un cluster Aurora senza disponibilità di lettura
- Verifica del tempo di attività per i cluster e le istanze Aurora
- Esempi di operazioni di riavvio Aurora

Riavvio di un'istanza database in un cluster Aurora

Questa procedura è l'operazione più importante che si esegue durante il riavvio con Aurora. Molte delle procedure di manutenzione prevedono il riavvio di una o più istanze database Aurora in un determinato ordine.

Console

Per riavviare un'istanza database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e l'istanza database da riavviare.
3. In Actions (Operazioni), scegliere Reboot (Riavvia).

   Viene visualizzata la pagina Reboot DB Instance (Riavvia istanza database).
4. Scegliere Reboot (Riavvia) per riavviare l'istanza database.

   Oppure scegliere Cancel (Annulla).

AWS CLI

Per riavviare un'istanza DB utilizzando il AWS CLI, chiama il `reboot-db-instance`comando.

Example

Per LinuxmacOS, oUnix:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance
```

Per Windows:

```
aws rds reboot-db-instance ^
```

```
    --db-instance-identifier mydbinstance
```

RDS API

Per riavviare un'istanza DB utilizzando Amazon RDSAPI, chiama l'`RebootDBInstance`operazione.

Riavvio di un cluster Aurora con disponibilità di lettura

Con la funzionalità di disponibilità di lettura, puoi riavviare l'istanza writer del tuo cluster Aurora senza riavviare le istanze di lettura nel cluster DB primario o secondario. In questo modo si contribuisce a mantenere alta la disponibilità del cluster per le operazioni di lettura durante il riavvio dell'istanza di scrittura. È possibile riavviare le istanze di lettura in un secondo momento, secondo una pianificazione adatta a te. Ad esempio, in un cluster di produzione è possibile riavviare le istanze del lettore una alla volta, iniziando solo al termine del riavvio dell'istanza primaria. Per ogni istanza DB che si riavvia, attenersi alla procedura in Riavvio di un'istanza database in un cluster Aurora.

La funzionalità di disponibilità in lettura per i cluster DB primari è disponibile in Aurora SQL My versione 2.10 e successive. La disponibilità di lettura per i cluster DB secondari è disponibile in Aurora SQL My versione 3.06 e successive.

Per Aurora Postgre SQL questa funzionalità è disponibile di default nelle seguenti versioni:

- 15.2 o versioni successive alla 15
- 14.7 o versioni successive alla 14
- 13.10 o versioni successive alla 13
- 12.14 e versioni successive alla 12

Per ulteriori informazioni sulla funzionalità di disponibilità di lettura in Aurora SQL Postgre, vedere. Miglioramento della disponibilità di lettura delle repliche Aurora

Prima di questa funzionalità, il riavvio dell'istanza principale provocava il riavvio di ogni istanza del lettore contemporaneamente. Se il cluster Aurora sta eseguendo una versione precedente, utilizzare invece la procedura di riavvio in Riavvio di un cluster Aurora senza disponibilità di lettura.

> **Note**
>
> La modifica al comportamento di riavvio nei cluster Aurora DB con disponibilità di lettura è diversa per i database globali Aurora nelle versioni di Aurora My precedenti alla SQL 3.06. Se

si riavvia l'istanza di scrittura per il cluster principale in un database globale Aurora, le istanze di lettura nel cluster primario rimangono disponibili. Tuttavia, le istanze database in qualsiasi cluster secondario si riavviano contemporaneamente.
Una versione limitata della funzionalità di disponibilità di lettura migliorata è supportata dai database globali di Aurora per le SQL versioni 12.16, 13.12, 14.9, 15.4 e successive di Aurora Postgre.

Si riavvia spesso il cluster dopo aver apportato modifiche ai gruppi di parametri del cluster. È possibile apportare modifiche ai parametri seguendo le procedure in . Si supponga di riavviare l'istanza database di scrittura in un cluster Aurora per applicare le modifiche ai parametri del cluster. Alcune o tutte le istanze database di lettura potrebbero continuare a utilizzare le vecchie impostazioni dei parametri. Tuttavia, le diverse impostazioni dei parametri non influiscono sull'integrità dei dati del cluster. Tutti i parametri del cluster che influiscono sull'organizzazione dei file di dati vengono utilizzati solo dall'istanza database di scrittura.

Ad esempio, in un SQL cluster Aurora My, è possibile aggiornare i parametri del cluster come `binlog_format` e `innodb_purge_threads` sull'istanza writer prima delle istanze reader. Solo l'istanza di scrittura sta scrivendo registri binari e eliminando i record di annullamento. Per i parametri che modificano il modo in cui le query interpretano SQL le istruzioni o l'output delle query, potrebbe essere necessario riavviare immediatamente le istanze del lettore. Lo si fa per evitare comportamenti imprevisti dell'applicazione durante le query. Ad esempio, supponiamo di modificare il parametro `lower_case_table_names` e di riavviare l'istanza di scrittura. In questo caso, le istanze di lettura potrebbero non essere in grado di accedere a una tabella appena creata finché non vengono tutte riavviate.

Per un elenco di tutti i parametri del SQL cluster Aurora My, vedere. Parametri a livello di cluster

Per un elenco di tutti i parametri del SQL cluster Aurora Postgre, vedere. Parametri a livello di cluster Aurora PostgreSQL

Tip

Aurora My SQL potrebbe comunque riavviare alcune istanze del lettore insieme all'istanza writer se il cluster sta elaborando un carico di lavoro con un throughput elevato.
La riduzione del numero di riavvii si applica anche durante le operazioni di failover. Aurora My riavvia SQL l'istanza Writer DB e la destinazione di failover solo durante un failover. Altre istanze DB di lettura nel cluster rimangono disponibili per continuare a elaborare le query

tramite connessioni all'endpoint di lettura. Pertanto, è possibile migliorare la disponibilità durante un failover se è presente più di un'istanza database di lettura in un cluster.

Riavvio di un cluster Aurora senza disponibilità di lettura

Se non si utilizza la funzionalità della disponibilità di lettura, è possibile riavviare l'intero cluster database Aurora riavviando l'istanza database di scrittura di tale cluster. A tale scopo, segui la procedura in Riavvio di un'istanza database in un cluster Aurora.

Il riavvio dell'istanza DB di scrittura avvia anche un riavvio per ogni istanza DB di lettura nel cluster. In questo modo, tutte le modifiche dei parametri a livello di cluster vengono applicate contemporaneamente a tutte le istanze DB. Tuttavia, il riavvio di tutte le istanze DB causa una breve interruzione del cluster. Le istanze DB di lettura rimangono non disponibili fino a quando l'istanza DB di scrittura termina il riavvio e diventa disponibile.

Questo comportamento di riavvio si applica a tutti i cluster DB creati in Aurora My SQL versione 2.09 e precedenti.

Per Aurora Postgre SQL questo comportamento si applica alle seguenti versioni:

- 14.6 e versioni precedenti alla 14
- 13.9 e versioni precedenti alla 13
- 12.13 e versioni precedenti alla 12
- Tutte le versioni di Postgre 11 SQL

Nella RDS console, l'istanza Writer DB ha il valore Writer nella colonna Role della pagina Databases. In RDSCLI, l'output del `describe-db-clusters` comando include una sezione`DBClusterMembers`. L'elemento `DBClusterMembers` che rappresenta l'istanza DB di scrittura ha un valore di `true` per il campo `IsClusterWriter`.

Important

Con la funzionalità di disponibilità in lettura, il comportamento di riavvio è diverso in Aurora My SQL e Aurora SQL Postgre: le istanze Reader DB in genere rimangono disponibili durante il riavvio dell'istanza writer. Quindi è possibile riavviare le istanze di lettura al momento opportuno. È possibile riavviare le istanze di lettura con una pianificazione scaglionata se

si desidera che alcune istanze di lettura siano sempre disponibili. Per ulteriori informazioni, consulta Riavvio di un cluster Aurora con disponibilità di lettura.

Verifica del tempo di attività per i cluster e le istanze Aurora

È possibile controllare e monitorare il periodo di tempo dall'ultimo riavvio per ogni istanza database nel cluster Aurora. Il CloudWatch parametro Amazon `EngineUptime` riporta il numero di secondi trascorsi dall'ultima volta in cui è stata avviata un'istanza DB. È possibile esaminare questo parametro in un point-in-time per scoprire il tempo di attività per l'istanza database. È inoltre possibile monitorare il parametro nel tempo per rilevare quando l'istanza viene riavviata.

È altresì possibile esaminare il parametro `EngineUptime` a livello del cluster. Le dimensioni `Minimum` e `Maximum` segnalano i valori di attività minori e maggiori per tutte le istanze database nel cluster. Per verificare l'ultima volta in cui una qualsiasi istanza di lettura in un cluster è stata riavviata, o riavviata per un altro motivo, monitorare il parametro a livello di cluster utilizzando la dimensione `Minimum`. Per verificare quale istanza nel cluster è durata più a lungo senza un riavvio, monitorare il parametro a livello di cluster utilizzando la dimensione `Maximum`. Ad esempio, è possibile confermare che tutte le istanze database nel cluster siano state riavviate dopo una modifica della configurazione.

Tip

Per il monitoraggio a lungo termine, si consiglia di monitorare il parametro `EngineUptime` per singole istanze anziché a livello di cluster. Il parametro `EngineUptime` a livello di cluster viene impostato su zero quando viene aggiunta una nuova istanza database al cluster. Tali modifiche del cluster possono avvenire nell'ambito di operazioni di manutenzione e dimensionamento come quelle eseguite da Auto Scaling.

CLIGli esempi seguenti mostrano come esaminare la `EngineUptime` metrica per le istanze writer e reader in un cluster. Gli esempi utilizzano un cluster denominato `tpch100g`. Questo cluster ha un'istanza database di scrittura `instance-1234`. Dispone inoltre di due istanze database di lettura `instance-7448` e `instance-6305`.

Innanzitutto, il comando `reboot-db-instance` riavvia una delle istanze di lettura. Il comando `wait` attende fino al termine del riavvio dell'istanza.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
```

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-6305
```

Il CloudWatch `get-metric-statistics` comando esamina la `EngineUptime` metrica negli ultimi cinque minuti a intervalli di un minuto. Il tempo di attività dell'istanza `instance-6305` viene ripristinato a zero e ricomincia a contare verso l'alto. Questo AWS CLI esempio per Linux utilizza la sostituzione di `$( )` variabili per inserire i timestamp appropriati nei comandi. CLI Utilizza anche il comando Linux `sort` per ordinare l'output nel momento in cui è stato raccolto il parametro. Il valore del timestamp è il terzo campo di ogni riga di output.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 231.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 291.0 2021-03-16T18:20:00+00:00 Seconds
DATAPOINTS 351.0 2021-03-16T18:21:00+00:00 Seconds
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
```

Il tempo di attività minimo per il cluster viene ripristinato a zero perché una delle istanze del cluster è stata riavviata. Il tempo di attività massimo per il cluster non viene ripristinato perché almeno una delle istanze database nel cluster è rimasta disponibile.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Minimum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63099.0 2021-03-16T18:12:00+00:00 Seconds
DATAPOINTS 63159.0 2021-03-16T18:13:00+00:00 Seconds
DATAPOINTS 63219.0 2021-03-16T18:14:00+00:00 Seconds
DATAPOINTS 63279.0 2021-03-16T18:15:00+00:00 Seconds
DATAPOINTS 51.0 2021-03-16T18:16:00+00:00 Seconds
```

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63389.0 2021-03-16T18:16:00+00:00 Seconds
DATAPOINTS 63449.0 2021-03-16T18:17:00+00:00 Seconds
DATAPOINTS 63509.0 2021-03-16T18:18:00+00:00 Seconds
DATAPOINTS 63569.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 63629.0 2021-03-16T18:20:00+00:00 Seconds
```

Quindi un altro comando `reboot-db-instance` riavvia l'istanza di scrittura del cluster. Un altro comando `wait` si interrompe fino al termine del riavvio dell'istanza di scrittura.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
  "DBInstanceIdentifier": "instance-1234",
  "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-1234
```

Ora il parametro `EngineUptime` per l'istanza di scrittura mostra che l'istanza `instance-1234` è stata riavviata di recente. Anche l'istanza di lettura `instance-6305` è stata riavviata automaticamente insieme all'istanza di scrittura. Questo cluster esegue Aurora My SQL 2.09, che non mantiene le istanze del lettore in esecuzione al riavvio dell'istanza writer.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-1234 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63749.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 63809.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 63869.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 41.0 2021-03-16T18:25:00+00:00 Seconds
DATAPOINTS 101.0 2021-03-16T18:26:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
```

```
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 531.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 49.0 2021-03-16T18:26:00+00:00 Seconds
```

Esempi di operazioni di riavvio Aurora

I seguenti SQL esempi Aurora My mostrano diverse combinazioni di operazioni di riavvio per istanze DB reader e writer in un cluster Aurora DB. Dopo ogni riavvio, SQL le query dimostrano l'uptime delle istanze nel cluster.

Argomenti

- Individuazione delle istanze di scrittore e lettore per un cluster Aurora
- Riavvio di una singola istanza di lettura
- Riavvio dell'istanza di scrittura
- Riavvio indipendente di scrittore e lettori
- Applicazione di una modifica dei parametri del cluster a un cluster Aurora My SQL versione 2.10

Individuazione delle istanze di scrittore e lettore per un cluster Aurora

In un SQL cluster Aurora My con più istanze DB, è importante sapere chi è l'autore e quali sono i lettori. Le istanze di scrittura e lettura possono anche cambiare ruolo quando si verifica un'operazione di failover. Pertanto, è meglio eseguire un controllo come il seguente prima di eseguire qualsiasi operazione che richieda un'istanza di scrittura o lettura. In questo caso, i valori `False` per `IsClusterWriter` identificano le istanze di lettura `instance-6305` e `instance-7448`. Il valore `True` identifica l'istanza di scrittura, `instance-1234`.

```
$ aws rds describe-db-clusters --db-cluster-id tpch100g \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:     tpch100g
Instance:    instance-6305    False
Instance:    instance-7448    False
```

```
Instance:    instance-1234    True
```

Prima di iniziare gli esempi di riavvio, l'istanza di scrittura ha un tempo di attività di circa una settimana. La SQL query in questo esempio mostra un modo SQL specifico di My per controllare l'uptime. È possibile utilizzare questa tecnica in un'applicazione di database. Per un'altra tecnica che utilizza AWS CLI e funziona per entrambi i motori Aurora, vedi. Verifica del tempo di attività per i cluster e le istanze Aurora

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-08 17:49:06.000000 | 174h 42m|
+----------------------------+---------+
```

Riavvio di una singola istanza di lettura

Questo esempio riavvia una delle istanze DB di lettura. Probabilmente questa istanza è stata sovraccaricata da una query sovradimensionata o da molte connessioni simultanee. Oppure è rimasta indietro rispetto all'istanza di scrittura a causa di un problema di rete. Dopo aver avviato l'operazione di riavvio, l'esempio utilizza un comando `wait` per sospendere fino a quando l'istanza non diventa disponibile. A quel punto, l'istanza ha un tempo di attività di alcuni minuti.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-6305
$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
```

```
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:35:02.000000 | 00h 03m |
+----------------------------+---------+
```

Il riavvio dell'istanza di lettura non ha influito sul tempo di attività dell'istanza di scrittura. Ha ancora un tempo di attività di circa una settimana.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+----------+
| Last Startup               | Uptime   |
+----------------------------+----------+
| 2021-03-08 17:49:06.000000 | 174h 49m |
+----------------------------+----------+
```

Riavvio dell'istanza di scrittura

Questo esempio riavvia l'istanza di scrittura. Questo cluster esegue Aurora My SQL versione 2.09. Poiché la SQL versione di Aurora My è precedente alla 2.10, il riavvio dell'istanza writer riavvia anche tutte le istanze di lettura nel cluster.

Un comando `wait` si interrompe fino al termine del riavvio. Ora il tempo di attività per quell'istanza viene ripristinato a zero. È possibile che un'operazione di riavvio possa richiedere tempi sostanzialmente diversi per le istanze DB di scrittura e lettura. Le istanze database di scrittura e lettura eseguono diversi tipi di operazioni di pulizia a seconda dei ruoli.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-1234",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-1234
```

```
$ mysql -h instance-1234.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:27.000000 | 00h 00m |
+----------------------------+---------+
```

Dopo il riavvio per l'istanza database di scrittura, entrambe le istanze database di lettura hanno anche il ripristino del tempo di attività. Il riavvio dell'istanza di scrittura ha causato il riavvio delle istanze di lettura. Questo comportamento si applica ai cluster Aurora Postgre e SQL ai cluster SQL Aurora My precedenti alla versione 2.10.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:35.000000 | 00h 00m |
+----------------------------+---------+

$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:33.000000 | 00h 01m |
+----------------------------+---------+
```

Riavvio indipendente di scrittore e lettori

Questi esempi successivi mostrano un cluster che esegue Aurora My SQL versione 2.10. In questa SQL versione di Aurora My e successive, puoi riavviare l'istanza di Writer senza causare il riavvio

di tutte le istanze del lettore. In questo modo, le applicazioni a uso intensivo di query non subiscono interruzioni quando si riavvia l'istanza di scrittura. È possibile riavviare le istanze di lettura in un secondo momento. È possibile eseguire questi riavvii in un momento di scarso traffico di query. È inoltre possibile riavviare le istanze di lettura una alla volta. In questo modo, almeno un'istanza di lettura è sempre disponibile per il traffico di query dell'applicazione.

L'esempio seguente utilizza un cluster denominato`cluster-2393`, che esegue Aurora My SQL version. `5.7.mysql_aurora.2.10.0` Questo cluster ha un'istanza di scrittura denominata `instance-9404` e tre istanze di lettura denominate `instance-6772`, `instance-2470` e `instance-5138`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:        cluster-2393
Instance:       instance-5138        False
Instance:       instance-2470        False
Instance:       instance-6772        False
Instance:       instance-9404        True
```

Il controllo del valore `uptime` di ogni istanza di database tramite il comando `mysql` mostra che ognuna ha approssimativamente lo stesso tempo di attività. Ad esempio, ecco il tempo di attività per `instance-5138`.

```
mysql> SHOW GLOBAL STATUS LIKE 'uptime';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Uptime        | 3866  |
+---------------+-------+
```

Utilizzando CloudWatch, possiamo ottenere le informazioni sull'uptime corrispondenti senza accedere effettivamente alle istanze. In questo modo, un amministratore può monitorare il database ma non può visualizzare o modificare alcun dato della tabella. In tal caso, specifichiamo un periodo di tempo che dura cinque minuti e controlliamo il valore del tempo di attività ogni minuto. I valori di attività crescenti dimostrano che le istanze non sono state riavviate durante quel periodo.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
```

```
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4648.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4708.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4768.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4828.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4888.0 2021-03-17T23:46:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4315.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4375.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4435.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4495.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4555.0 2021-03-17T23:46:00+00:00 Seconds
```

Ora riavviamo una delle istanze di lettura, `instance-5138`. Aspettiamo che l'istanza diventi nuovamente disponibile dopo il riavvio. Ora il monitoraggio del tempo di attività per un periodo di cinque minuti mostra che il tempo di attività è stato ripristinato a zero durante quel periodo. Il valore di attività più recente è stato misurato cinque secondi dopo il completamento del riavvio.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4500.0 2021-03-17T23:46:00+00:00 Seconds
DATAPOINTS 4560.0 2021-03-17T23:47:00+00:00 Seconds
DATAPOINTS 4620.0 2021-03-17T23:48:00+00:00 Seconds
```

```
DATAPOINTS 4680.0 2021-03-17T23:49:00+00:00 Seconds
DATAPOINTS  5.0 2021-03-17T23:50:00+00:00 Seconds
```

Successivamente, eseguiamo un riavvio per l'istanza di scrittur, `instance-9404`. Confrontiamo i valori di attività per l'istanza di scrittura e una delle istanze di lettura. Così facendo, possiamo vedere che il riavvio dello scrittore non ha causato un riavvio per i lettori. Nelle versioni precedenti ad Aurora My SQL 2.10, i valori di uptime per tutti i lettori venivano reimpostati contemporaneamente all'autore.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
{
  "DBInstanceIdentifier": "instance-9404",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-9404

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 371.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 431.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 491.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 551.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS  37.0  2021-03-18T00:01:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5215.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 5275.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 5335.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 5395.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS 5455.0 2021-03-18T00:01:00+00:00 Seconds
```

Per assicurarsi che tutte le istanze di lettura abbiano le stesse modifiche ai parametri di configurazione dell'istanza di scrittura, riavviare tutte le istanze di lettura dopo quelle di scrittura. Questo esempio riavvia tutti i lettori e quindi attende che tutti siano disponibili prima di procedere.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6772
{
  "DBInstanceIdentifier": "instance-6772",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}

$ aws rds wait db-instance-available --db-instance-id instance-6772
$ aws rds wait db-instance-available --db-instance-id instance-2470
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

Ora possiamo vedere che l'istanza database di scrittura ha il tempo di attività più alto. Il valore di attività di questa istanza è aumentato costantemente durante tutto il periodo di monitoraggio. Le istanze database di lettura sono state tutte riavviate dopo il lettore. Possiamo vedere il punto nel periodo di monitoraggio in cui ogni lettore è stato riavviato e il suo tempo di attività è stato ripristinato a zero.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 457.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 517.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 577.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 637.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS  697.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
```

```
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-2470 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5819.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS  35.0  2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 95.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 155.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 215.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 1085.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 1145.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 1205.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS  49.0  2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 109.0 2021-03-18T00:12:00+00:00 Seconds
```

Applicazione di una modifica dei parametri del cluster a un cluster Aurora My SQL versione 2.10

L'esempio seguente dimostra come applicare una modifica dei parametri a tutte le istanze DB nel cluster Aurora My 2.10. SQL Con questa SQL versione di Aurora My, riavvii l'istanza di writer e tutte le istanze di Reader in modo indipendente.

L'esempio utilizza il parametro `lower_case_table_names` di SQL configurazione My per l'illustrazione. Quando questa impostazione dei parametri è diversa tra le istanze database di scrittura e lettura, una query potrebbe non essere in grado di accedere a una tabella dichiarata con un nome maiuscolo o minuscolo misto. Oppure, se due nomi di tabelle differiscono solo in termini di lettere maiuscole e minuscole, una query potrebbe accedere alla tabella errata.

Questo esempio mostra come determinare le istanze di scrittura e lettura nel cluster esaminando l'attributo `IsClusterWriter` di ciascuna istanza. Il cluster è denominato `cluster-2393`. Il cluster ha un'istanza di scrittura denominata `instance-9404`. Le istanze di lettura nel cluster sono denominate `instance-5138` e `instance-2470`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
```

```
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
cluster-2393
instance-5138        False
instance-2470        False
instance-9404        True
```

Per dimostrare gli effetti della modifica del parametro `lower_case_table_names`, abbiamo impostato due gruppi di parametri del cluster DB. Il gruppo di parametri `lower-case-table-names-0` ha questo parametro impostato su 0. Il gruppo di parametri `lower-case-table-names-1` ha questo gruppo di parametri impostato su 1.

```
$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-0' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-0
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-0",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-0"
    }
}

$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-1' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-1",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-1"
    }
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-0 \
  --parameters
 ParameterName=lower_case_table_names,ParameterValue=0,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-0"
}
```

```
$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-1 \
    --parameters
 ParameterName=lower_case_table_names,ParameterValue=1,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-1"
}
```

Il valore predefinito di `lower_case_table_names` è 0. Con questa impostazione dei parametri, la tabella `foo` è distinta dalla tabella F00. Questo esempio verifica che il parametro sia ancora all'impostazione predefinita. Quindi l'esempio crea tre tabelle che differiscono solo per lettere maiuscole e minuscole nei nomi.

```
mysql> create database lctn;
Query OK, 1 row affected (0.07 sec)

mysql> use lctn;
Database changed
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> create table foo (s varchar(128));
mysql> insert into foo values ('Lowercase table name foo');

mysql> create table Foo (s varchar(128));
mysql> insert into Foo values ('Mixed-case table name Foo');

mysql> create table FOO (s varchar(128));
mysql> insert into FOO values ('Uppercase table name FOO');

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
```

```
| s                          |
+----------------------------+
| Mixed-case table name Foo |
+----------------------------+

mysql> select * from FOO;
+----------------------------+
| s                          |
+----------------------------+
| Uppercase table name FOO |
+----------------------------+
```

Successivamente, associamo il gruppo di parametri DB al cluster per impostare il parametro `lower_case_table_names` su 1. Questa modifica ha effetto solo dopo il riavvio di ogni istanza database.

```
$ aws rds modify-db-cluster --db-cluster-identifier cluster-2393 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
  "DBClusterIdentifier": "cluster-2393",
  "DBClusterParameterGroup": "lower-case-table-names-1",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.0"
}
```

Il primo riavvio che facciamo è per l'istanza database di scrittura. Quindi aspettiamo che l'istanza diventi nuovamente disponibile. A quel punto, ci connettiamo all'endpoint di scrittura e verifichiamo che l'istanza di scrittura abbia il valore del parametro modificato. Il comando `SHOW TABLES` conferma che il database contiene le tre tabelle diverse. Tuttavia, le query che fanno riferimento a tabelle denominate `foo`, `Foo` o `F00` accedono tutte alla tabella il cui nome è interamente minuscolo, `foo`.

```
# Rebooting the writer instance
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
$ aws rds wait db-instance-available --db-instance-id instance-9404
```

Ora, le query che utilizzano l'endpoint cluster mostrano gli effetti della modifica dei parametri. Indipendentemente dal fatto che il nome della tabella nella query sia maiuscolo, minuscolo o misto, l'SQListruzione accede alla tabella il cui nome è tutto in minuscolo.

```
mysql> select @@lower_case_table_names;
+--------------------------+
```

```
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+

mysql> use lctn;
mysql> show tables;
+----------------+
| Tables_in_lctn |
+----------------+
| FOO            |
| Foo            |
| foo            |
+----------------+

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

L'esempio successivo mostra le stesse query della precedente. In questo caso, le query utilizzano l'endpoint di lettura e vengono eseguite su una delle istanze database di lettura. Queste istanze non sono ancora state riavviate. Pertanto, hanno ancora l'impostazione originale per il parametro `lower_case_table_names`. Ciò significa che le query possono accedere a ciascuna delle tabelle `foo`, `Foo` e `FOO`.

```
mysql> select @@lower_case_table_names;
```

```
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                         |
+--------------------------+
| Mixed-case table name Foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
+--------------------------+
```

Successivamente, riavviamo una delle istanze di lettura e aspettiamo che diventi nuovamente disponibile.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-2470
```

Mentre è connesso all'endpoint dell'istanza per `instance-2470`, una query mostra che il nuovo parametro è attivo.

```
mysql> select @@lower_case_table_names;
```

```
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+
```

A questo punto, le due istanze di lettura nel cluster sono in esecuzione con impostazioni `lower_case_table_names` diverse. Pertanto, qualsiasi connessione all'endpoint di lettura del cluster utilizza un valore per questa impostazione che è imprevedibile. È importante riavviare immediatamente l'altra istanza di lettura in modo che entrambe abbiano impostazioni coerenti.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

L'esempio seguente conferma che tutte le istanze del lettore hanno la stessa impostazione per il parametro `lower_case_table_names`. I comandi controllano il valore di impostazione `lower_case_table_names` su ogni istanza di lettura. Quindi lo stesso comando che utilizza l'endpoint di lettura dimostra che ogni connessione all'endpoint di lettura utilizza una delle istanze di lettura, ma quale sia non è prevedibile.

```
# Check lower_case_table_names setting on each reader instance.

$ mysql -h instance-5138.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

$ mysql -h instance-2470.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-2470            |                        1 |
+--------------------------+--------------------------+
```

```
# Check lower_case_table_names setting on the reader endpoint of the cluster.

$ mysql -h cluster-2393.cluster-ro-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

# Run query on writer instance

$ mysql -h cluster-2393.cluster-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-9404            |                        1 |
+--------------------------+--------------------------+
```

Con la modifica dei parametri applicata ovunque, possiamo vedere l'effetto dell'impostazione `lower_case_table_names=1`. Se la tabella viene definita `foo`, `Foo` o `F00` la query converte il nome in `foo` e accede alla stessa tabella in ogni caso.

```
mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
```

```
| s                         |
+---------------------------+
| Lowercase table name foo |
+---------------------------+
```

Failing su un cluster Amazon Aurora DB

È possibile eseguire un failover manuale di un cluster Aurora DB, ad esempio, quando si desidera sostituire un'istanza DB Writer con un'istanza Writer DB con Aurora Serverless v2 un'istanza Writer.

Aurora esegue il failover su una nuova istanza DB primaria in due modi:

- Promuovendo un'istanza Reader DB esistente alla nuova istanza primaria
- Creando una nuova istanza primaria

Se il cluster DB dispone di una o più istanze di lettura, un lettore viene promosso a istanza principale durante un evento di errore. Per aumentare la disponibilità del cluster DB, si consiglia di creare almeno una o più istanze di lettura in due o più zone di disponibilità diverse. Per ulteriori informazioni sul meccanismo di failover, vedere. Tolleranza ai guasti di un cluster DB Aurora

È possibile utilizzare AWS Management Console AWS CLI, o RDS API per eseguire un failover manuale.

Console

Per eseguire il failover di un cluster DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi seleziona un'istanza DB nel cluster DB di cui desideri eseguire il failover.
3. Per Actions (Operazioni), scegliere Failover.

   Viene visualizzata la pagina di conferma.
4. Scegli Failover.

   La pagina Database mostra che lo stato del cluster DB è in fase di failing-over. Lo stato torna a Disponibile al termine del failover e vengono visualizzati i ruoli per le istanze DB primarie nuove e precedenti.

AWS CLI

Per eseguire il failover di un cluster DB utilizzando il AWS CLI, chiamate il failover-db-clustercomando. Specifica i seguenti parametri:

- `--db-cluster-identifier`— Il cluster DB di cui si desidera eseguire il failover.
- `--target-db-instance-identifier`— Il nome dell'istanza DB da promuovere a istanza DB principale.

Example

Per LinuxmacOS, oUnix:

```
aws rds failover-db-cluster \
    --db-cluster-identifier mydbcluster \
    --target-db-instance-identifier mydbcluster-instance-2
```

Per Windows:

```
aws rds failover-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --target-db-instance-identifier mydbcluster-instance-2
```

RDS API

Per modificare un cluster DB utilizzando Amazon RDSAPI, chiama l'ailoverDBClusteroperazione F. Specifica i seguenti parametri:

- DBClusterIdentifier
- argetDBInstanceIdentificatore T

Eliminazione di cluster e istanze database di Aurora

È possibile eliminare un cluster database Aurora quando non è più necessario. L'eliminazione del cluster rimuove il volume del cluster contenente tutti i dati. Prima di eliminare il cluster, puoi decidere di salvare una snapshot dei dati. È possibile ripristinare lo snapshot in un secondo momento per creare un nuovo cluster contenente gli stessi dati.

Inoltre, è possibile eliminare le istanze database da un cluster, conservando però il cluster stesso e i dati che contiene. L'eliminazione delle istanze database consente di ridurre i costi se il cluster non è occupato e non ha bisogno della capacità di calcolo di più istanze database.

Argomenti

- Eliminazione di un cluster database Aurora
- Protezione dall'eliminazione per i cluster Aurora
- Eliminazione di un cluster Aurora arrestato
- Eliminazione di Aurora I SQL miei cluster che sono repliche lette
- Lo snapshot finale durante l'eliminazione di un cluster
- Eliminazione di un'istanza database da un cluster database Aurora

Eliminazione di un cluster database Aurora

Aurora non fornisce un metodo con una singola operazione per eliminare un cluster database. Questa scelta progettuale ha lo scopo di impedire la perdita accidentale di dati o di portare l'applicazione offline. Le applicazioni Aurora sono in genere mission-critical e richiedono elevata disponibilità. Pertanto, Aurora facilita il dimensionamento del cluster verso l'alto e verso il basso aggiungendo e rimuovendo istanze database. La rimozione del cluster database stesso richiede di effettuare un'eliminazione separata.

Utilizza la procedura generale seguente per rimuovere tutte le istanze database da un cluster e quindi eliminare il cluster stesso.

1. Elimina tutte le istanze reader nel cluster. Utilizza la procedura descritta in Eliminazione di un'istanza database da un cluster database Aurora.

   Se il cluster dispone di più istanze di lettura, l'eliminazione di una delle istanze riduce la capacità di calcolo del cluster. L'eliminazione delle istanze reader garantisce innanzitutto che il cluster rimanga disponibile per tutta la procedura e non esegua operazioni di failover non necessarie.

2. Elimina l'istanza writer dal cluster. Ancora una volta, utilizza la procedura in Eliminazione di un'istanza database da un cluster database Aurora.

   Quando elimini le istanze database, il cluster e il relativo volume cluster associato vengono mantenuti anche dopo l'eliminazione di tutte le istanze database.

3. Elimina il cluster database.

   - AWS Management Console: scegli il cluster, quindi scegli Elimina nel menu lOperazioni. Puoi scegliere le opzioni seguenti per conservare i dati del cluster in caso di necessità in un secondo momento:
     - Crea uno snapshot finale del volume del cluster. L'impostazione predefinita prevede la creazione di uno snapshot finale.
     - Mantieni backup automatici. L'impostazione predefinita non prevede il mantenimento dei backup automatici.

     > **Note**
     >
     > I backup automatici per i cluster Aurora Serverless v1 DB non vengono conservati.

     Aurora richiede inoltre di confermare che intendi veramente eliminare il cluster.
   - CLIe API: richiama il `delete-db-cluster` CLI comando o l'operazione. `DeleteDBCluster` API Puoi scegliere le opzioni seguenti per conservare i dati del cluster in caso di necessità in un secondo momento:
     - Crea uno snapshot finale del volume del cluster.
     - Mantieni backup automatici.

     > **Note**
     >
     > I backup automatici per i cluster Aurora Serverless v1 DB non vengono conservati.

Argomenti

- Eliminazione di un cluster Aurora vuoto
- Eliminazione di un cluster Aurora con una singola istanza database
- Eliminazione di un cluster Aurora con più istanze database

Eliminazione di un cluster Aurora vuoto

Puoi eliminare un cluster DB vuoto utilizzando AWS Management Console AWS CLI, o Amazon RDSAPI.

> **Tip**
>
> Puoi mantenere un cluster senza istanze DB per preservare i tuoi dati senza incorrere in CPU costi per il cluster. Puoi iniziare a utilizzare di nuovo il cluster molto rapidamente creando una o più nuove istanze database per il cluster. Puoi eseguire operazioni amministrative specifiche di Aurora sul cluster se questo non dispone di istanze database associate. Non puoi accedere ai dati o eseguire operazioni che richiedono la connessione a un'istanza database.

Console

Per eliminare un cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Database, quindi scegli il cluster database multi-AZ che vuoi eliminare.
3. In Actions (Azioni), scegliere Delete (Elimina).
4. Per creare uno snapshot database multi-AZ finale per il cluster di database, scegli Crea snapshot finale. Si tratta dell'impostazione di default.
5. Se si è scelto di creare uno snapshot finale, immettere il Final snapshot name (Nome dello snapshot finale).
6. Per mantenere i backup automatici, scegliere Retain automated backups (Mantieni backup automatici). Questa non è l'impostazione predefinita.
7. Immettere **`delete me`** nella casella.
8. Scegli Elimina.

CLI

Per eliminare un cluster Aurora DB vuoto utilizzando AWS CLI, chiamare il delete-db-clustercomando.

Supponiamo che il cluster vuoto `deleteme-zero-instances` sia stato utilizzato solo per lo sviluppo e il test e non contenga dati importanti. In tal caso, non è necessario conservare una snapshot del volume del cluster quando si elimina il cluster. Nell'esempio seguente viene illustrato un cluster che non contiene istanze database e quindi elimina il cluster vuoto senza creare uno snapshot finale o mantenendo backup automatici.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-zero-instances --output
 text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-zero-instances

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-zero-instances \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-zero-instances",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

RDS API

Per eliminare un cluster Aurora DB vuoto utilizzando Amazon RDSAPI, chiama l'operazione D. eleteDBCluster

Eliminazione di un cluster Aurora con una singola istanza database

Puoi anche eliminare l'ultima istanza database anche se il cluster database è protetto dall'eliminazione. In questo caso, il cluster di database continua a esistere e i dati vengono conservati. Puoi accedere di nuovo ai dati collegando una nuova istanza database al cluster.

Nell'esempio seguente viene illustrato come il comando `delete-db-cluster` non funziona quando al cluster sono ancora associate istanze database. Questo cluster ha una singola istanza database writer. Esaminando le istanze database nel cluster, si controlla l'attributo `IsClusterWriter` di ciascuna di esse. Il cluster potrebbe avere zero o un'istanza database writer. Un valore `true` indica un'istanza database writer. Un valore `false` indica un'istanza database reader. Il cluster potrebbe avere zero, una o più istanze database reader. In questo caso, eliminiamo l'istanza database writer utilizzando il comando `delete-db-instance`. Non appena l'istanza database entra nello stato `deleting`, possiamo eliminare anche il cluster. Anche per questo esempio, supponiamo che il

cluster non contenga dati che valga la pena conservare. Pertanto, non creiamo uno snapshot del volume del cluster né manteniamo backup automatici.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only --skip-final-
snapshot
An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-only \
  --query '*[].[DBClusterIdentifier,Status,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]'
[
    [
        "deleteme-writer-only",
        "available",
        [
            [
                "instance-2130",
                true
            ]
        ]
    ]
]

$ aws rds delete-db-instance --db-instance-identifier instance-2130
{
  "DBInstanceIdentifier": "instance-2130",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-only",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Eliminazione di un cluster Aurora con più istanze database

Se il cluster contiene più istanze database, in genere esiste una singola istanza writer e una o più istanze reader. Le istanze reader aiutano con la disponibilità elevata, rimanendo in standby per assumere il controllo se l'istanza writer riporta un problema. Le istanze reader possono inoltre essere utilizzate per il dimensionamento del cluster in modo da gestire un carico di lavoro a uso intensivo di lettura senza aggiungere sovraccarico all'istanza writer.

Per eliminare un cluster con più istanze database reader, elimina prima le istanze reader, quindi l'istanza writer. L'eliminazione dell'istanza di scrittura lascia il cluster e i relativi dati al loro posto. Il cluster viene eliminato mediante un'azione distinta.

- Per la procedura di eliminazione di un'istanza database Aurora, consulta Eliminazione di un'istanza database da un cluster database Aurora.
- Per la procedura di eliminazione di un'istanza database writer in un cluster Aurora, consulta Eliminazione di un cluster Aurora con una singola istanza database.
- Per la procedura di eliminazione di un cluter Aurora vuoto, consulta Eliminazione di un cluster Aurora vuoto.

Questo CLI esempio mostra come eliminare un cluster contenente sia un'istanza Writer DB che un'istanza DB a lettore singolo. L'output di `describe-db-clusters` mostra che `instance-7384` è l'istanza writer e `instance-1039` è l'istanza reader. Nell'esempio viene eliminata prima l'istanza reader, poiché l'eliminazione dell'istanza writer mentre esiste ancora un'istanza reader provocherebbe un'operazione di failover. Non ha senso promuovere l'istanza reader a writer se prevedi di eliminare anche quell'istanza. Di nuovo, supponiamo che queste istanze `db.t2.small` vengano utilizzate solo per lo sviluppo e il test e quindi l'operazione di eliminazione ignora lo snapshot finale e non mantiene i backup automatici.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader --skip-
final-snapshot

An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-and-reader --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
```

```
Cluster:        deleteme-writer-and-reader
Instance:       instance-1039  False
Instance:       instance-7384   True

$ aws rds delete-db-instance --db-instance-identifier instance-1039
{
  "DBInstanceIdentifier": "instance-1039",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-7384
{
  "DBInstanceIdentifier": "instance-7384",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-and-reader",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Nell'esempio seguente viene illustrato come eliminare un cluster database contenente un'istanza database writer e più istanze database reader. Utilizza un output conciso dal comando `describe-db-clusters` per ottenere un report delle istanze writer e reader. Ancora una volta, eliminiamo tutte le istanze database reader prima di eliminare l'istanza database writer. Non importa con quale ordine eliminiamo le istanze database reader.

Supponiamo che questo cluster con diverse istanze database contenga dati che vale la pena conservare. Pertanto, il comando `delete-db-cluster` di questo esempio include i parametri `--no-skip-final-snapshot` e `--final-db-snapshot-identifier` per specificare i dettagli della snapshot da creare. Include anche il parametro `--no-delete-automated-backups` per mantenere i backup automatici.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-multiple-readers --
output text \
```

```
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-multiple-readers
Instance:       instance-1010   False
Instance:       instance-5410   False
Instance:       instance-9948   False
Instance:       instance-8451   True

$ aws rds delete-db-instance --db-instance-identifier instance-1010
{
  "DBInstanceIdentifier": "instance-1010",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-5410
{
  "DBInstanceIdentifier": "instance-5410",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-9948
{
  "DBInstanceIdentifier": "instance-9948",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-8451
{
  "DBInstanceIdentifier": "instance-8451",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-multiple-readers \
  --no-delete-automated-backups \
  --no-skip-final-snapshot \
  --final-db-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
  "DBClusterIdentifier": "deleteme-multiple-readers",
  "Status": "available",
  "Engine": "aurora-mysql"
```

```
}
```

Nell'esempio seguente viene illustrato come confermare che Aurora abbia creato la snapshot richiesta. Puoi richiedere i dettagli per la snapshot specifica specificandone l'identificativo `deleteme-multiple-readers-final-snapshot`. Puoi inoltre ottenere un report di tutte le snapshot per il cluster che è stato eliminato specificando l'identificativo del cluster `deleteme-multiple-readers`. Entrambi i comandi restituiscono informazioni sulla stessa snapshot.

```
$ aws rds describe-db-cluster-snapshots \
  --db-cluster-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...

$ aws rds describe-db-cluster-snapshots --db-cluster-identifier deleteme-multiple-
readers
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...
```

Protezione dall'eliminazione per i cluster Aurora

Non puoi eliminare cluster per i quali è abilitata la protezione dall'eliminazione. Puoi eliminare istanze database all'interno del cluster, ma non il cluster stesso. In questo modo, il volume del cluster contenente tutti i dati è al sicuro dall'eliminazione accidentale. Aurora applica la protezione dall'eliminazione per un cluster DB indipendentemente dal fatto che si tenti di eliminare il cluster utilizzando la console AWS CLI, il o il. RDS API

La protezione da eliminazione viene abilitata per impostazione predefinita quando crei un cluster di database di produzione con la AWS Management Console. Tuttavia, la protezione dall'eliminazione è disabilitata per impostazione predefinita se si crea un cluster utilizzando o. AWS CLI API L'attivazione o la disattivazione della protezione contro l'eliminazione non causa un'interruzione. Per poter eliminare il cluster, modifica il cluster e disabilita la protezione dall'eliminazione. Per ulteriori informazioni sull'attivazione e sulla disattivazione della protezione da eliminazione, consulta Modifica del cluster di database tramite la console, la CLI e l'API.

> **Tip**
>
> Anche se tutte le istanze database sono state eliminate, potrai comunque accedere ai dati creando una nuova istanza database nel cluster.

Eliminazione di un cluster Aurora arrestato

Non puoi eliminare un cluster se si trova nello stato `stopped`. In questo caso, avvia il cluster prima di eliminarlo. Per ulteriori informazioni, consulta Avvio di un cluster di database Aurora.

Eliminazione di Aurora I SQL miei cluster che sono repliche lette

Per Aurora MySQL, non è possibile eliminare un'istanza DB in un cluster DB se entrambe le seguenti condizioni sono vere:

- Il cluster database è una replica di lettura di un altro cluster di database Aurora.
- L'istanza database è l'unica istanza nel cluster database.

Per eliminare un'istanza database in questo caso, prima di tutto promuovi il cluster di database in modo che non sia più una replica di lettura. Al termine della promozione, puoi eliminare l'istanza database finale nel cluster database. Per ulteriori informazioni, consulta Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS.

Lo snapshot finale durante l'eliminazione di un cluster

In questa sezione, gli esempi mostrano come è possibile scegliere se eseguire un'istantanea finale quando si elimina un cluster Aurora. Se si sceglie di creare uno snapshot finale ma il nome specificato corrisponde a uno snapshot esistente, l'operazione si interrompe con un errore. In

questo caso, esamina i dettagli dello snapshot per confermare se rappresenta il dettaglio corrente o se si tratta di uno snapshot precedente. Se lo snapshot esistente non contiene i dati più recenti che desideri conservare, rinomina lo snapshot e riprova oppure specifica un nome diverso per il parametro snapshot finale.

Eliminazione di un'istanza database da un cluster database Aurora

Puoi eliminare un'istanza database da un cluster database Aurora come parte del processo di eliminazione dell'intero cluster. Se il cluster contiene un certo numero di istanze database, l'eliminazione del cluster richiede l'eliminazione di ogni singola istanza database. Puoi eliminare una o più istanze reader da un cluster anche se il cluster in esecuzione. Questa operazione può essere effettuata per ridurre la capacità di calcolo e i costi associati se il cluster non è occupato.

Per eliminare un'istanza database, devi specificare il nome dell'istanza.

È possibile eliminare un'istanza DB utilizzando il AWS Management Console AWS CLI, il o il RDSAPI.

> **Note**
>
> Quando una replica Aurora viene eliminata, l'endpoint dell'istanza viene rimosso immediatamente e la replica Aurora viene rimossa dall'endpoint di lettura. Se sono presenti istruzioni in esecuzione nella replica Aurora da eliminare, è disponibile un periodo di tolleranza di tre minuti. L'esecuzione delle istruzioni esistenti può essere completata correttamente durante questo periodo di tolleranza. Quando finisce il periodo di tolleranza, la replica Aurora viene chiusa ed eliminata.

Per i cluster database Aurora, l'eliminazione di un'istanza database non comporta necessariamente l'eliminazione dell'intero cluster. Puoi eliminare un'istanza database in un cluster Aurora per ridurre la capacità di calcolo e i costi associati quando il cluster non è occupato. Per informazioni sulle circostanze speciali per i cluster Aurora che dispongono di un'istanza database o zero istanze database, consulta Eliminazione di un cluster Aurora con una singola istanza database e Eliminazione di un cluster Aurora vuoto.

> **Note**
>
> Non puoi eliminare un'istanza database se è abilitata la protezione dall'eliminazione. Per ulteriori informazioni, consulta Protezione dall'eliminazione per i cluster Aurora.

Puoi disabilitare la protezione dall'eliminazione modificando l'istanza database. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Console

Per eliminare un'istanza database in un cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e selezionare l'istanza database da eliminare.
3. In Actions (Azioni), selezionare Delete (Elimina).
4. Immettere **`delete me`** nella casella.
5. Scegli Elimina.

AWS CLI

Per eliminare un'istanza DB utilizzando il AWS CLI, chiama il delete-db-instancecomando e specifica il `--db-instance-identifier` valore.

Example

Per LinuxmacOS, oUnix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance
```

Per Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance
```

RDS API

Per eliminare un'istanza DB utilizzando Amazon RDSAPI, chiama l'eleteDBInstanceoperazione D e specifica il `DBInstanceIdentifier` parametro.

> **Note**
>
> Quando lo stato di un'istanza DB è`deleting`, il relativo valore del certificato CA non viene visualizzato nella RDS console o nell'output dei AWS CLI comandi o RDS API delle operazioni. Per ulteriori informazioni sui certificati CA, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

Etichettatura di Amazon Aurora e risorse Amazon RDS

Un RDS tag Amazon è una coppia nome-valore che definisci e associ a una RDS risorsa Amazon come un'istanza DB o uno snapshot DB. Il nome viene definito chiave. Facoltativamente, puoi fornire un valore per la chiave.

Puoi utilizzare il AWS Management Console AWS CLI, il o Amazon RDS API per aggiungere, elencare ed eliminare tag nelle RDS risorse Amazon. Quando usi CLI oAPI, assicurati di fornire l'Amazon Resource Name (ARN) per la RDS risorsa con cui lavorare. Per ulteriori informazioni sulla creazione di unARN, consultaCostruzione di un ARN per Amazon RDS.

Puoi utilizzare i tag per aggiungere metadati alle tue risorse Aurora e Amazon RDS. Puoi utilizzare i tag per aggiungere notazioni personalizzate su istanze di database, snapshot, cluster Aurora e così via. In questo modo puoi documentare le tue risorse Aurora e Amazon RDS. I tag possono inoltre essere utilizzati con procedure di manutenzione automatizzate.

In particolare, puoi utilizzare questi tag con IAM le politiche. Puoi usarli per gestire l'accesso ad Aurora e alle RDS risorse Amazon e per controllare quali azioni possono essere applicate a tali risorse. Puoi utilizzare questi tag anche per tenere traccia dei costi raggruppando le spese per risorse con tag simili.

Puoi taggare le seguenti Aurora e Amazon RDS:

- Istanze DB
- Cluster database
- Cluster globali Aurora
- Endpoint del cluster DB
- Repliche di lettura
- Snapshot DB
- Snapshot cluster database
- Istanze database riservate
- Abbonamenti a eventi
- Gruppi di opzioni database
- Gruppi di parametri database
- Gruppi di parametri di cluster database

- Gruppi di sottoreti database
- RDSProxy
- RDSEndpoint proxy
- Distribuzioni blu/verde
- Zero integrazioni ETL

> **Note**
>
> Attualmente, non è possibile RDS contrassegnare proxy e endpoint RDS proxy utilizzando. AWS Management Console

Argomenti

- Perché usare i tag RDS delle risorse di Amazon?
- Come funzionano i tag RDS delle risorse di Amazon
- Le migliori pratiche per etichettare le risorse Amazon RDS
- Copia di tag in snapshot di cluster database
- Aggiungere ed eliminare tag in Amazon RDS
- Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare

Perché usare i tag RDS delle risorse di Amazon?

Puoi usare i tag per fare quanto segue:

- Categorizza RDS le tue risorse per applicazione, progetto, reparto, ambiente e così via. Ad esempio, è possibile utilizzare una chiave tag per definire una categoria, in cui il valore del tag è un elemento di questa categoria. Potresti creare il tag`environment=prod`. Oppure puoi definire una chiave di tag `project` e un valore di tag di`Salix`, che indica che una RDS risorsa Amazon è assegnata al progetto Salix.
- Automatizza le attività di gestione delle risorse. Ad esempio, è possibile creare una finestra di manutenzione per le istanze con tag diversa dalla finestra per `environment=prod` le istanze contrassegnate. `environment=test` È inoltre possibile configurare istantanee DB automatiche per le istanze con tag. `environment=prod`

- Controlla l'accesso alle RDS risorse all'interno di una IAM policy. Puoi farlo utilizzando la chiave di condizione `aws:ResourceTag/tag-key` globale. Ad esempio, una policy potrebbe consentire solo agli utenti del `DBAdmin` gruppo di modificare le istanze DB con `environment=prod` cui sono contrassegnate. Per informazioni sulla gestione dell'accesso alle risorse con tag tramite IAM policy, vedere Gestione delle identità e degli accessi per Amazon Aurora Controlling access to AWS resources nella AWS Identity and Access Management User Guide.
- Monitora le risorse in base a un tag. Ad esempio, puoi creare una CloudWatch dashboard Amazon per le istanze DB contrassegnate con`environment=prod`.
- Tieni traccia dei costi raggruppando le spese per risorse con tag simili. Ad esempio, se tagghi RDS le risorse associate al progetto Salix con`project=Salix`, puoi generare report sui costi e allocare le spese per questo progetto. Per ulteriori informazioni, consulta Come funziona la AWS fatturazione con i tag in Amazon RDS.

Come funzionano i tag RDS delle risorse di Amazon

AWS non applica alcun significato semantico ai tag. I tag sono interpretati prettamente come stringhe di caratteri.

Argomenti

- Set di tag su Amazon RDS
- Struttura dei tag in Amazon RDS
- RDSRisorse Amazon idonee per l'etichettatura
- Come funziona la AWS fatturazione con i tag in Amazon RDS

Set di tag su Amazon RDS

Ogni RDS risorsa Amazon ha un contenitore chiamato set di tag. Il contenitore include tutti i tag assegnati alla risorsa. Una risorsa ha esattamente un set di tag.

Un set di tag contiene da 0 a 50 tag. Se aggiungete un tag a una RDS risorsa con la stessa chiave di un tag di risorsa esistente, il nuovo valore sovrascrive quello precedente.

Struttura dei tag in Amazon RDS

La struttura di un RDS tag è la seguente:

Chiave tag

La chiave è il nome richiesto del tag. Il valore della stringa deve contenere da 1 a 128 caratteri Unicode e non può essere preceduto da o. `aws:` `rds:` La stringa può contenere solo l'insieme di lettere, cifre, spazi bianchi Unicode,,,,,`_`, `.` e. `:` `/` `=` `+` `-` `@` L'espressione regolare di Java è. `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"` Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole. Pertanto, le chiavi `project` e `Project` sono distinte.

Una chiave è unica per un set di tag. Ad esempio, non è possibile avere una coppia di chiavi in un set di tag con la stessa chiave ma con valori diversi, come `project=Trinity` e. `project=Xanadu`

Valore del tag

Il valore è un valore di stringa opzionale del tag. Il valore della stringa deve essere composto da 1 a 256 caratteri Unicode. La stringa può contenere solo l'insieme di lettere, cifre, spazi bianchi Unicode,,,,,, `_.`, `:` e. `/` `=` `+` `-` `@` L'espressione regolare di Java è. `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"` I valori di tag fanno distinzione tra maiuscole e minuscole. Pertanto, i valori `prod` e `Prod` sono distinti.

I valori non devono essere necessariamente univoci in un set di tag e possono essere nulli. Ad esempio, puoi avere una coppia chiave-valore in un set di tag `project=Trinity` e `cost-center=Trinity`.

RDSRisorse Amazon idonee per l'etichettatura

Puoi taggare le seguenti RDS risorse Amazon:

- Istanze DB
- Cluster database
- Endpoint del cluster DB
- Repliche di lettura
- Snapshot DB
- Snapshot cluster database
- Istanze database riservate
- Abbonamenti a eventi
- Gruppi di opzioni database
- Gruppi di parametri database

- Gruppi di parametri di cluster database
- Gruppi di sottoreti database
- RDSProxy
- RDSEndpoint proxy

> **Note**
>
> Attualmente, non è possibile RDS etichettare proxy e endpoint RDS proxy utilizzando il. AWS Management Console

- Distribuzioni blu/verde
- Zero ETL integrazioni (anteprima)

Come funziona la AWS fatturazione con i tag in Amazon RDS

Utilizza i tag per organizzare la AWS fattura in modo da rispecchiare la tua struttura dei costi. A tale scopo, registrati per ricevere la Account AWS fattura con i valori chiave dell'etichetta inclusi. Per visualizzare il costo delle risorse combinate, puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione per visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing .

Come funzionano i tag di allocazione dei costi con le istantanee dei cluster DB

È possibile aggiungere un tag a uno snapshot del cluster DB. Tuttavia, l'addebito non rifletterà questo raggruppamento. Affinché i tag di allocazione dei costi si applichino agli snapshot del cluster DB, devono essere soddisfatte le seguenti condizioni:

- I tag devono essere allegati all'istanza DB principale.
- L'istanza DB principale deve esistere nella Account AWS stessa istantanea del cluster DB.
- L'istanza DB principale deve esistere nella Regione AWS stessa istantanea del cluster DB.

Le istantanee del cluster DB sono considerate orfane se non esistono nella stessa regione dell'istanza DB principale o se l'istanza DB principale viene eliminata. Le snapshot DB orfane non supportano i tag di allocazione dei costi. I costi delle istantanee orfane vengono aggregati in un'unica

voce senza tag. Le istantanee dei cluster DB tra più account non sono considerate orfane quando vengono soddisfatte le seguenti condizioni:

- Esistono nella stessa regione dell'istanza DB principale.
- L'istanza DB principale è di proprietà dell'account di origine.

> **Note**
>
> Se l'istanza DB principale è di proprietà di un account diverso, i tag di allocazione dei costi non si applicano agli snapshot tra account nell'account di destinazione.

Le migliori pratiche per etichettare le risorse Amazon RDS

Quando utilizzi i tag, ti consigliamo di attenerti alle seguenti best practice:

- Documenta le convenzioni per l'uso dei tag seguite da tutti i team dell'organizzazione. In particolare, assicuratevi che i nomi siano sia descrittivi che coerenti. Ad esempio, standardizza il formato `environment:prod` anziché etichettare alcune risorse con. `env:production`

  > **Important**
  >
  > Non archiviate informazioni di identificazione personale (PII) o altre informazioni riservate o sensibili nei tag.

- Automatizza l'etichettatura per garantire la coerenza. Ad esempio, puoi utilizzare le seguenti tecniche:
  - Includi tag in un AWS CloudFormation modello. Quando crei risorse con il modello, le risorse vengono etichettate automaticamente.
  - Definisci e applica i tag utilizzando AWS Lambda le funzioni.
  - Crea un SSM documento che includa i passaggi per aggiungere tag alle tue RDS risorse.
- Usa i tag solo quando necessario. È possibile aggiungere fino a 50 tag per una singola RDS risorsa, ma è consigliabile evitare la proliferazione e la complessità non necessarie dei tag.
- Controlla periodicamente i tag per verificarne la pertinenza e la precisione. Rimuovi o modifica i tag obsoleti secondo necessità.
- Prendi in considerazione la possibilità di creare tag con il AWS Tag Editor in. AWS Management Console Puoi utilizzare il Tag Editor per aggiungere tag a più AWS risorse supportate, incluse RDS

le risorse, contemporaneamente. Per ulteriori informazioni, consulta Tag Editor nella Guida per l'utente di AWS Resource Groups.

Copia di tag in snapshot di cluster database

Quando si crea o si ripristina un cluster DB, è possibile specificare che i tag del cluster vengano copiati nelle istantanee del cluster DB. La copia dei tag garantisce che i metadati per gli snapshot DB corrispondano a quelli del cluster database di origine. Garantisce inoltre che tutte le policy di accesso per lo snapshot DB corrispondano a quelle del cluster database di origine. I tag non vengono copiati per impostazione predefinita.

È possibile specificare che i tag vengano copiati nelle snapshot DB per le seguenti azioni:

- Creazione di un cluster di database
- Ripristino di un cluster DB
- Creazione di una replica di lettura
- Copia di una snapshot cluster database

> **Note**
>
> In alcuni casi, è possibile includere un valore per il `--tags` parametro del create-db-snapshot AWS CLI comando. Oppure potreste fornire almeno un tag all'reateDBSnapshotAPIoperazione C. In questi casi, RDS non copia i tag dall'istanza DB di origine alla nuova istantanea del database. Questa funzionalità si applica anche se per l'istanza database di origine è stata attivata l'opzione `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`).
> Con questo approccio puoi creare la copia di un'istanza database da uno snapshot DB ed evitare di aggiungere tag che non si applicano alla nuova istanza database. Lo snapshot DB viene creato utilizzando il AWS CLI `create-db-snapshot` comando (o l'`CreateDBSnapshot`RDSAPIoperazione). Dopo aver creato uno snapshot DB, puoi aggiungere i tag come descritto più avanti in questo argomento.

Aggiungere ed eliminare tag in Amazon RDS

Puoi eseguire le operazioni indicate di seguito:

- Crea tag quando crei una risorsa, ad esempio quando esegui il AWS CLI comando`create-db-instance`.
- Aggiungi tag a una risorsa esistente utilizzando il comando`add-tags-to-resource`.
- Elenca i tag associati a una risorsa specifica utilizzando il comando`list-tags-for-resource`.
- Aggiorna i tag usando il comando`add-tags-to-resource`.
- Rimuovi i tag da una risorsa usando il comando`remove-tags-from-resource`.

Le seguenti procedure mostrano come è possibile eseguire operazioni di tagging tipiche sulle risorse relative alle istanze DB e ai cluster Aurora DB. Tieni presente che i tag sono memorizzati nella cache a fini di autorizzazione. Per questo motivo, quando aggiungi o aggiorni tag sulle RDS risorse Amazon, possono passare diversi minuti prima che le modifiche siano disponibili.

Console

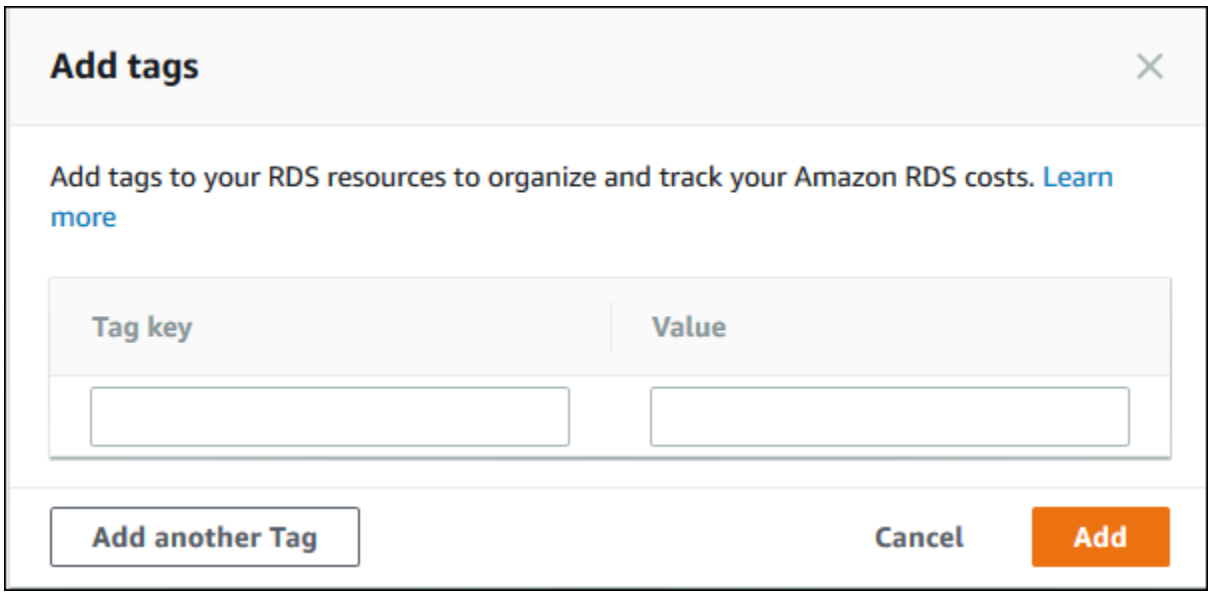
Il processo per etichettare una RDS risorsa Amazon è simile per tutte le risorse. La procedura seguente mostra come etichettare un'istanza Amazon RDS DB.

Aggiunta di un tag a un'istanza database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).

   > **Note**
   >
   > Per filtrare l'elenco di istanze database nel riquadro Databases (Database), inserire una stringa di testo per Filter instances (Filtra istanze). Vengono visualizzate solo le istanze database che contengono la stringa.

3. Scegliere il nome dell'istanza database a cui si desidera aggiungere tag per visualizzarne i dettagli.
4. Nella sezione dei dettagli, scorrere verso il basso fino alla sezione Tags (Tag).
5. Scegliere Aggiungi. Viene visualizzata la finestra Add tags (Aggiungi tag).



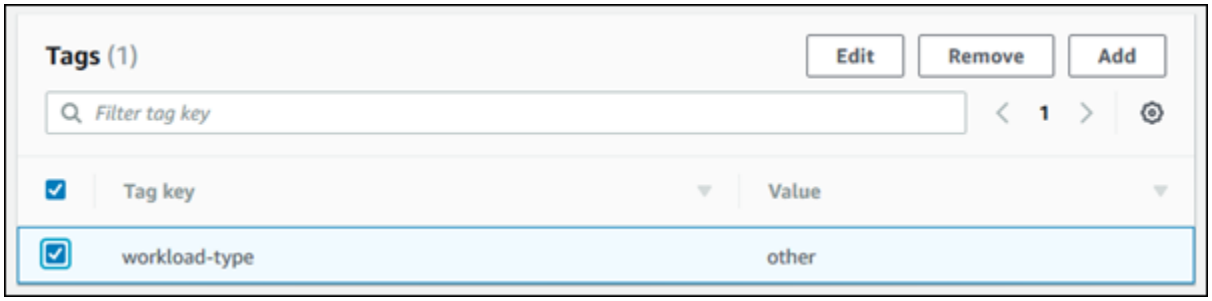
6. Inserire un valore per Tag key (Chiave tag) e Value (Valore).
7. Per aggiungere un altro tag, scegliere Add another Tag (Aggiungi un altro tag) e inserire un valore per Tag key (Chiave tag) e Value (Valore).

   Ripetere questa operazione tutte le volte necessarie.
8. Scegliere Aggiungi.

Eliminazione di un tag da un'istanza database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).

   > **Note**
   >
   > Per filtrare l'elenco di istanze database nel riquadro Databases (Database), inserire una stringa di testo nella casella Filter instances (Filtra istanze). Vengono visualizzate solo le istanze database che contengono la stringa.
3. Scegliere il nome dell'istanza database per visualizzarne i dettagli.
4. Nella sezione dei dettagli, scorrere verso il basso fino alla sezione Tags (Tag).
5. Scegliere il tag da eliminare.



6. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra Delete tags (Elimina tag).

AWS CLI

È possibile aggiungere, elencare o rimuovere i tag per un'istanza database utilizzando AWS CLI.

- Per aggiungere uno o più tag a una RDS risorsa Amazon, usa il AWS CLI comando `add-tags-to-resource`.
- Per elencare i tag su una RDS risorsa Amazon, usa il AWS CLI comando `list-tags-for-resource`.
- Per rimuovere uno o più tag da una RDS risorsa Amazon, usa il AWS CLI comando `remove-tags-from-resource`.

Per ulteriori informazioni su come costruire il necessarioARN, consultaCostruzione di un ARN per Amazon RDS.

RDS API

Puoi aggiungere, elencare o rimuovere tag per un'istanza DB utilizzando Amazon RDSAPI.

- Per aggiungere un tag a una RDS risorsa Amazon, usa l'`AddTagsToResource`operazione.
- Per elencare i tag assegnati a una RDS risorsa Amazon, usa il `ListTagsForResource`.
- Per rimuovere i tag da una RDS risorsa Amazon, usa l'`RemoveTagsFromResource`operazione.

Per ulteriori informazioni su come costruire il necessarioARN, consultaCostruzione di un ARN per Amazon RDS.

Quando si XML utilizza Amazon RDSAPI, i tag utilizzano lo schema seguente:

```
<Tagging>
```

```
      <TagSet>
          <Tag>
              <Key>Project</Key>
              <Value>Trinity</Value>
          </Tag>
          <Tag>
              <Key>User</Key>
              <Value>Jones</Value>
          </Tag>
      </TagSet>
  </Tagging>
```

La tabella seguente fornisce un elenco dei XML tag consentiti e delle relative caratteristiche. I valori di `Key` e `Value` fanno distinzione tra maiuscole e minuscole. Ad esempio, `project=Trinity` e `PROJECT=Trinity` sono tag distinti.

Elemento del tagging	Descrizione
TagSet	Un set di tag è un contenitore per tutti i tag assegnati a una RDS risorsa Amazon. Ogni risorsa può disporre di un solo set di tag. Collaborate con un TagSet solo tramite Amazon RDSAPI.
Tag	Un tag è una coppia chiave-valore definita dall'utente. Un set di tag può contenere da 1 a 50 tag.
Chiave	Una chiave corrisponde al nome obbligatorio del tag. Per le restrizioni, vediStruttura dei tag in Amazon RDS. Il valore della stringa può essere composto da 1 a 128 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti Unicode, '_', '.', '/', '=', '+', '-' (espressioni regolari Java: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Le chiavi devono essere uniche per un set di tag. Ad esempio, non è possibile avere una coppia di chiavi in un set di tag con la stessa chiave ma con valori diversi, ad esempio. project/Trinity and project/Xanadu
Valore	Un valore è il valore opzione del tag. Per le restrizioni, vedere. Struttura dei tag in Amazon RDS

Elemento del tagging	Descrizione
	Il valore della stringa può essere composto da 1 a 256 caratteri Unicode e non può avere il prefisso `aws:` o `rds:`. La stringa può contenere solo il set di lettere, cifre, spazi vuoti Unicode, '_', '.', '/', '=', '+', '-' (espressioni regolari Java: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). I valori non devono essere necessariamente univoci in un set di tag e possono essere Null. Ad esempio, puoi avere una coppia chiave-valore in un set di tag di. project/Trinity and cost-center/Trinity

Tutorial: utilizza i tag per specificare i cluster DB Aurora da arrestare

Si assuma di creare un numero di cluster database Aurora in un ambiente di sviluppo o di test. È necessario mantenere tutti questi cluster per diversi giorni. Alcuni cluster eseguono test durante la notte. Altri cluster possono essere fermati durante la notte e ricominciare il giorno successivo. L'esempio seguente mostra come assegnare un tag a quei cluster che possono essere arrestati durante la notte. L'esempio mostra come uno script può rilevare i cluster che hanno quel tag e quindi arrestarli. In questo esempio, la parte del valore della coppia chiave-valore non ha importanza. La presenza del tag `stoppable` indica che il cluster dispone di questa proprietà definita dall'utente.

Per specificare quali cluster DB Aurora arrestare

1. Determina il ARN tipo di cluster che desideri designare come bloccabile.

   I comandi e APIs per l'etichettatura funzionano con. ARNs In questo modo, possono funzionare senza problemi tra AWS regioni, AWS account e diversi tipi di risorse che potrebbero avere nomi brevi identici. È possibile specificare l'ID del cluster ARN anziché l'ID del cluster nei CLI comandi che operano sui cluster. Sostituisci il nome del tuo cluster con *`dev-test-cluster`*. Nei comandi successivi che utilizzano ARN parametri, sostituisci ARN il tuo cluster. ARNInclude l'ID AWS dell'account e il nome della AWS regione in cui si trova il cluster.

   ```
   $ aws rds describe-db-clusters --db-cluster-identifier dev-test-cluster \
     --query "*[].{DBClusterArn:DBClusterArn}" --output text
   arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
   ```

2. Aggiungi il tag `stoppable` a questo cluster.

Scegli il nome per il tag. Con questo approccio puoi evitare di definire una convenzione di denominazione che codifichi tutte le informazioni rilevanti nei nomi. Con la convenzione puoi codificare le informazioni nel nome dell'istanza database o nei nomi di altre risorse. Poiché questo esempio considera il tag come un attributo presente o assente, viene omessa la parte `Value=` del parametro `--tags`.

```
$ aws rds add-tags-to-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster \
  --tags Key=stoppable
```

3. Verifica che il tag sia presente nel cluster.

   Questi comandi recuperano le informazioni sui tag per il cluster in JSON formato e in testo semplice separato da tabulazioni.

```
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
{
    "TagList": [
        {
            "Key": "stoppable",
            "Value": ""

        }
    ]
}
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster --output
 text
TAGLIST stoppable
```

4. Per arrestare tutti i cluster designati come `stoppable`, prepara un elenco di tutti i cluster. Scorri l'elenco e verifica se ogni cluster è contrassegnato con l'attributo pertinente.

   Questo esempio di Linux utilizza lo shell scripting per salvare l'elenco dei cluster in un file temporaneo e quindi eseguire CLI comandi ARNs per ogni cluster.

```
$ aws rds describe-db-clusters --query "*[].[DBClusterArn]" --output text >/tmp/
cluster_arns.lst
$ for arn in $(cat /tmp/cluster_arns.lst)
do
```

```
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 'TAGLIST\tstoppable')"
  if [[ ! -z "$match" ]]
  then
      echo "Cluster $arn is tagged as stoppable. Stopping it now."
# Note that you can specify the full ARN value as the parameter instead of the
 short ID 'dev-test-cluster'.
      aws rds stop-db-cluster --db-cluster-identifier $arn
  fi
done

Cluster arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster is tagged as
 stoppable. Stopping it now.
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "us-east-1e",
            "us-east-1c",
            "us-east-1d"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "dev-test-cluster",
        ...
```

Puoi eseguire uno script come questo alla fine di ogni giorno per assicurarti che i cluster non essenziali vengano arrestati. Puoi inoltre pianificare un processo utilizzando un'utilità come `cron` per eseguire tale controllo ogni notte. Ad esempio, puoi eseguire questa operazione nel caso in cui alcuni cluster venissero lasciati in esecuzione per errore. Puoi quindi ottimizzare il comando che prepara l'elenco di cluster da controllare.

Il comando seguente produce un elenco di cluster che si trovano nello stato `available`. Lo script può ignorare i cluster che sono già stati arrestati, perché avranno valori di stato diversi, ad esempio `stopped` o `stopping`.

```
$ aws rds describe-db-clusters \
  --query '*[].{DBClusterArn:DBClusterArn,Status:Status}|[?Status == `available`]|[].
{DBClusterArn:DBClusterArn}' \
  --output text
arn:aws:rds:us-east-1:123456789:cluster:cluster-2447
arn:aws:rds:us-east-1:123456789:cluster:cluster-3395
```

```
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
arn:aws:rds:us-east-1:123456789:cluster:pg2-cluster
```

Tip

Puoi utilizzare l'assegnazione di tag e la ricerca di cluster con i tag per ridurre i costi in altri modi. Supponi, ad esempio, uno scenario in cui i cluster database Aurora sono utilizzati per lo sviluppo e il test. In questo caso, potresti designare alcuni cluster da eliminare alla fine di ogni giornata o eliminare solo le istanze database del lettore. In alternativa, potresti designarli per modificare le istanze database in classi di istanza database di piccole dimensioni durante i periodi previsti di basso utilizzo.

Nomi di risorse Amazon (ARNs) in Amazon RDS

Le risorse create in Amazon Web Services sono identificate in modo univoco con un Amazon Resource Name (ARN). Per alcune RDS operazioni Amazon, devi identificare in modo univoco una RDS risorsa Amazon specificando la sua. ARN Ad esempio, quando crei una replica di lettura di un'istanza RDS DB, devi fornire la replica di lettura ARN per l'istanza DB di origine.

Per informazioni sulla creazione ARN e sull'ottenimento di un'esistenteARN, consulta i seguenti argomenti.

Argomenti

- Costruzione di un ARN per Amazon RDS
- Creare un account esistente ARN per Amazon RDS

Costruzione di un ARN per Amazon RDS

Le risorse create in Amazon Web Services sono identificate in modo univoco con un Amazon Resource Name (ARN). È possibile creare un ARN per una risorsa Amazon RDS utilizzando la sintassi seguente.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Nome della regione	Regione	Endpoint	Protocollo	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
		rds-fips.us-east-1.api.aws	HTTPS	
		rds-fips.us-east-1.amazonaws.com	HTTPS	

Nome della regione	Regione	Endpoint	Protocollo
		rds.us-east-1.api.aws	HTTPS
Stati Uniti occidentali (California settentrionale)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
Africa (Cape Town)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
Asia Pacifico (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS
Asia Pacific (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
		rds.ap-south-2.api.aws	HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Asia Pacifico (Giacarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com rds.ap-southeast-3.api.aws	HTTPS HTTPS
Asia Pacifico (Malesia)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Asia Pacifico (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS
Asia Pacifico (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS
Asia Pacifico (Osaka-Locale)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS
Asia Pacifico (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS
Asia Pacifico (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com rds.ap-southeast-1.api.aws	HTTPS HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Asia Pacifico (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
		rds.ap-southeast-2.api.aws	HTTPS
Asia Pacifico (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
Asia Pacifico (Tailandia)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
Asia Pacifico (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
		rds.ap-northeast-1.api.aws	HTTPS
Canada (Centrale)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
		rds.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.api.aws	HTTPS
		rds-fips.ca-central-1.amazonaws.com	HTTPS
Canada occidentale (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS
		rds-fips.ca-west-1.amazonaws.com	HTTPS
Europa (Francoforte)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
		rds.eu-central-1.api.aws	HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Europa (Irlanda)	eu-west-1	rds.eu-west-1.amazonaws.com rds.eu-west-1.api.aws	HTTPS HTTPS
Europa (Londra)	eu-west-2	rds.eu-west-2.amazonaws.com rds.eu-west-2.api.aws	HTTPS HTTPS
Europa (Milano)	eu-south-1	rds.eu-south-1.amazonaws.com rds.eu-south-1.api.aws	HTTPS HTTPS
Europa (Parigi)	eu-west-3	rds.eu-west-3.amazonaws.com rds.eu-west-3.api.aws	HTTPS HTTPS
Europa (Spagna)	eu-south-2	rds.eu-south-2.amazonaws.com rds.eu-south-2.api.aws	HTTPS HTTPS
Europa (Stoccolma)	eu-north-1	rds.eu-north-1.amazonaws.com rds.eu-north-1.api.aws	HTTPS HTTPS
Europa (Zurigo)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS
Israele (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS
Messico (centrale)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS

Nome della regione	Regione	Endpoint	Protocollo
Medio Oriente (Bahrein)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS
Medio Oriente (Emirati Arabi Uniti)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS
Sud America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

La tabella riportata di seguito mostra il formato da utilizzare quando si crea un ARN per un determinato tipo di risorsa di Amazon RDS.

Tipo di risorsa	Formato ARN
Istanza database	arn:aws:rds: ::db: *<region>* *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
Cluster DB	arn:aws:rds: *<region>* :cluster: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
Sottoscrizione a eventi	arn:aws:rds: *<region>* :it: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB parameter group (Gruppo di parametri database)	arn:aws:rds: *<region>* ::pg: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
Gruppo di parametri del cluster DB	arn:aws:rds: *<region>* :cluster-pg: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
Istanza database riservata	arn:aws:rds: *<region>* ::ri: *<account>* *<name>*

Tipo di risorsa	Formato ARN
	Per esempio: `arn:aws:rds: us-east-2 :123456789012 :ri:my-reserved-postgresql`
Gruppo di sicurezza DB	arn:aws:rds: *<region>* :secgrp: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :secgrp:my-public`
Snapshot di database automatizzato	arn:aws:rds: *<region>* *<account>* ::snapshot:rds: *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :snapshot:rds: my-mysql-db-2019-07-22-07-23`
Snapshot di cluster database automatizzato	arn:aws:rds: *<region>* *<account>* :cluster-snapshot:rds: *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot:rds: my-aurora-cluster-2019-07-22-16-16`
Snapshot di database manuale	arn:aws:rds: *<region>* :snapshot: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`

Tipo di risorsa	Formato ARN
Snapshot del cluster di database manuale	arn:aws:rds: *<region>* *<account>* :cluster-snapshot: *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`
Gruppo di sottoreti DB	arn:aws:rds: *<region>* :subgrp: *<account>* *<name>* Per esempio: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

Creare un account esistente ARN per Amazon RDS

Puoi ottenere l'ARNimmagine di una RDS risorsa usando AWS Management Console, AWS Command Line Interface (AWS CLI), o RDSAPI.

Console

Per ottenerne una ARN da AWS Management Console, accedi alla risorsa ARN per cui desideri una risorsa e visualizza i dettagli di quella risorsa.

Ad esempio, è possibile ottenere i ARN dati relativi a un cluster DB dalla scheda Configurazione dei dettagli del cluster DB.

AWS CLI

Per ottenere un ARN da AWS CLI per una particolare RDS risorsa, si usa il `describe` comando relativo a quella risorsa. La tabella seguente mostra ogni AWS CLI comando e la ARN proprietà utilizzata con il comando per ottenere unARN.

AWS CLI comando	ARNproprietà
describe-event-subscriptions	EventSubscriptionArn

AWS CLI comando	ARNproprietà
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-gruppi	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	R eservedDBInstance Arn
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

Ad esempio, il AWS CLI comando seguente ottiene la ARN per un'istanza DB.

Example

Per LinuxmacOS, oUnix:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

Per Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
```

```
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

L'output del comando è simile al seguente:

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

RDS API

Per ottenere un valore ARN per una particolare RDS risorsa, è possibile richiamare le seguenti RDS API operazioni e utilizzare le ARN proprietà illustrate di seguito.

RDSAPIoperazione	ARNproprietà
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
escribeDBParameterGruppi D	DBParameterGroupArn
D escribeDBCluster Parameter Groups	DBClusterParameterGroupArn
D escribeDBInstances	DBInstanceArn
D escribeDBSecurity Gruppi	DBSecurityGroupArn
D escribeDBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	R eservedDBInstance Arn
Gruppi D escribeDBSubnet	DBSubnetGroupArn
D escribeDBClusters	DBClusterArn

RDSAPIoperazione	ARNproprietà
D escribeDBCluster Istantanee	DBClusterSnapshotArn

Aggiornamenti di Amazon Aurora

Gli aggiornamenti di Amazon Aurora vengono resi disponibili periodicamente. Gli aggiornamenti vengono applicati ai cluster di database di Amazon Aurora durante le finestre di manutenzione del sistema. La tempistica di applicazione degli aggiornamenti dipende dall'impostazione della regione e della finestra di manutenzione per il cluster DB e anche dal tipo di aggiornamento. Gli aggiornamenti richiedono il riavvio del database, pertanto ci sarà un tempo di inattività di 20 - 30 secondi, al termine del quale si potrà ricominciare a usare i cluster di database o i cluster. Puoi visualizzare o modificare le impostazioni della finestra di manutenzione dalla AWS Management Console.

> **Note**
>
> Il tempo necessario per riavviare l'istanza DB dipende dal processo di ripristino dell'arresto anomalo, l'attività del database al momento del riavvio e il comportamento del motore DB specifico. Per ottimizzare i tempi di riavvio, è consigliabile ridurre l'attività del database il più possibile durante il processo di riavvio. Riducendo l'attività del database si riduce l'attività di rollback per le transazioni in transito.

Per informazioni sugli aggiornamenti del sistema operativo per Amazon Aurora, consulta Aggiornamenti del sistema operativo per i cluster Aurora DB.

Alcuni aggiornamenti sono specifici di un motore database supportato da Aurora. Per ulteriori informazioni sugli aggiornamenti del motore database, consulta la tabella seguente.

Motore del database	Aggiornamenti
Amazon Aurora My SQL	Consulta la sezione Aggiornamenti del motore di database per Amazon Aurora My SQL
Amazon Aurora Poster SQL	Per informazioni, consulta Aggiornamenti del motore di database per Amazon Aurora Postgre SQL

Identificare la versione Amazon Aurora

Amazon Aurora include alcune funzionalità generiche per Aurora e disponibili per tutti i cluster database Aurora. Aurora include altre funzionalità specifiche per un particolare motore del database

supportato da Aurora. Queste funzionalità sono disponibili solo per i cluster Aurora DB che utilizzano quel motore di database, come Aurora Postgre. SQL

Un'istanza database Aurora fornisce due numeri di versione: il numero di versione di Aurora e quello del motore di database di Aurora. Per i numeri di versione di Aurora viene utilizzato il formato seguente.

```
<major version>.<minor version>.<patch version>
```

Per ottenere il numero di versione di Aurora da un'istanza database Aurora che utilizza un motore di database specifico, potrai servirti di una delle query seguenti.

Motore di database	Query
Amazon Aurora My SQL	`SELECT AURORA_VERSION();` `SHOW @@aurora_version;`
Amazon Aurora Poster SQL	`SELECT AURORA_VERSION();`

Supporto esteso di Amazon RDS con

Con l'estensione del supporto di Amazon RDS, puoi continuare a eseguire il tuo database su una versione principale del motore dopo la data di fine del supporto standard per Aurora a un costo aggiuntivo. Dopo la data di fine del supporto standard di Aurora, se non hai disabilitato RDS Extended Support durante la creazione o il ripristino delle istanze DB Amazon RDS Amazon Aurora le registrerà automaticamente in RDS Extended Support. La registrazione automatica a RDS Extended Support non modifica il motore del database e non influisce sull'uptime o sulle prestazioni dell'istanza DB.

RDS Extended Support fornisce i seguenti aggiornamenti e supporto tecnico:

- Aggiornamenti di sicurezza critici e di livello elevato CVEs per l'istanza DB o il cluster di database, incluso il motore di database
- Correzioni di bug e patch per problemi critici
- La possibilità di aprire casi di supporto e ricevere assistenza per la risoluzione dei problemi nell'ambito del contratto di servizio standard di Amazon RDS

Questa offerta a pagamento ti dà più tempo per eseguire l'aggiornamento a una versione del motore principale supportata. Ad esempio, la data di fine del supporto standard di Aurora per Aurora MySQL versione 2 è il 31 ottobre 2024. Tuttavia, non sei pronto per l'aggiornamento manuale alla versione 3 di Aurora MySQL prima di tale data. In questo caso, Amazon Aurora registra automaticamente il cluster in RDS Extended Support il 31 ottobre 2024 e puoi continuare a eseguire Aurora MySQL versione 2. A partire dal 1° dicembre 2024, Amazon Aurora ti addebita automaticamente l'importo del supporto RDS Extended Support.

RDS Extended Support è disponibile fino a 3 anni dopo la data di fine del supporto standard di per una versione principale del motore (3 anni e 4 mesi per Aurora MySQL versione 2). Trascorso questo periodo, se non hai aggiornato la versione del motore principale a una versione supportata, Aurora aggiornerà automaticamente la versione principale del motore. Ti consigliamo di eseguire l'aggiornamento a una versione del motore principale supportata il prima possibile.

Per ulteriori informazioni sulle date di fine del supporto standard di Aurora e sulle date di fine del supporto esteso per RDS, consulta Calendario delle release per le versioni principali di Aurora MySQL e Calendario delle release per le versioni principali di Aurora PostgreSQL.

Argomenti

- Panoramica di Amazon RDS Extended Support
- Costi di Amazon RDS Extended Support
- Versioni con Amazon RDS Extended Support
- Aurora e le responsabilità dei clienti con Amazon RDS Extended Support
- Creazione Aurora DB o un cluster globale con Amazon RDS Extended Support
- Visualizzazione della registrazione delle globali in Amazon RDS Extended Support
- Visualizzazione delle date di supporto per le versioni del motore in Amazon RDS Extended Support
- Ripristino di Aurora DB o di un cluster globale con Amazon Extended Support RDS

Panoramica di Amazon RDS Extended Support

Dopo la data di fine del supporto standard di Aurora, se non hai disabilitato RDS Extended Support durante la creazione o il ripristino delle istanze DB Amazon RDS Amazon Aurora le registrerà automaticamente in RDS Extended Support. Aurora aggiorna automaticamente l'istanza DB all'ultima versione secondaria rilasciata prima della data di fine del supporto standard di Aurora, se non utilizzi già quella versione. Amazon Aurora effettuerà l'upgrade della versione secondaria solo dopo la fine della data di supporto standard di Aurora per la versione principale del motore.

È possibile creare nuovi database con le principali versioni del motore che hanno raggiunto la data di fine del supporto standard di Aurora. Aurora registra automaticamente questi nuovi database in RDS Extended Support e ti addebita il costo di questa offerta.

Se esegui l'upgrade a un motore che è ancora coperto dal supporto standard Aurora prima della data di fine del supporto standard di Aurora, Amazon RDS Amazon Aurora non registrerà il tuo motore a Extended Support.

Se il ripristino non riesce, Amazon Aurora registrerà automaticamente il motore a RDS Extended Support con una versione compatibile con lo snapshot.

Puoi terminare l'iscrizione a RDS Extended Support in qualsiasi momento.

Per ulteriori informazioni sulle date di fine del supporto standard di Aurora e sulle date di fine del supporto esteso per RDS, consulta Calendario delle release per le versioni principali di Aurora MySQL e Calendario delle release per le versioni principali di Aurora PostgreSQL.

Costi di Amazon RDS Extended Support

Ti verranno addebitati i costi per tutti i motori iscritti all'Extended RDS Support a partire dal giorno successivo alla data di fine del supporto standard di Aurora. Per la data di fine del supporto standard di Aurora, vedi. Versioni principali di Amazon Aurora

Il costo aggiuntivo per RDS Extended Support si interrompe automaticamente quando si esegue una delle seguenti azioni:

- Passa a una versione del motore coperta dal supporto standard.
- Elimina il database su cui è in esecuzione una versione principale dopo la data di fine del supporto standard di Aurora.

Gli addebiti verranno riavviati se la versione del motore di destinazione entrerà in RDS Extended Support in futuro.

Ad esempio, 11 entrerà in Extended Support il 1° marzo 2024, ma gli addebiti inizieranno solo il 1° aprile 2024. Aggiorna il tuo SQL database Postgre 11 a Postgre . SQL Ti verranno addebitati solo 30 giorni di Extended Support 11. SQL Continuerai a eseguire Postgre SQL 12 su questa istanza DB oltre la data di RDS fine del supporto standard del 28 febbraio 2025. Il tuo database sarà nuovamente soggetto ai costi di RDS Extended Support a partire dal 1° marzo 2025.

Per ulteriori informazioni, consulta i prezzi di di Amazon Aurora.

Evitare i costi per Amazon RDS Extended Support

È possibile evitare l'addebito dell'RDSExtended Support impedendo ad Aurora di creare o ripristinare un' Aurora DB o un cluster globale oltre la data di fine del supporto standard di Aurora . A tale scopo, utilizzare o il. AWS CLI RDS API

Nel AWS CLI, specificare `open-source-rds-extended-support-disabled` l'`--engine-lifecycle-support`opzione. Nel RDSAPI, specificare `open-source-rds-extended-support-disabled` il `LifeCycleSupport` parametro. Per ulteriori informazioni, consulta Creazione Aurora DB o di un cluster globale o Ripristino di Aurora DB o di un cluster globale.

Versioni con Amazon RDS Extended Support

RDS Extended Support è disponibile per Aurora MySQL e per Aurora PostgreSQL. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora.

RDS Extended Support è disponibile solo su alcune versioni secondarie. Le versioni secondarie sono indicate come compatibili con RDS Extended Support solo dopo che le versioni principali hanno raggiunto la data di fine del ciclo di vita della Community. Per ulteriori informazioni, vedere Calendario di rilascio per Aurora MySQL nelle Note di rilascio di Aurora MySQL e Calendario di rilascio per Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

RDS Extended Support è disponibile solo suAurora Serverless v2. Non è disponibile su. Aurora Serverless v1

Puoi anche visualizzare le informazioni sulle date di supporto per le versioni del motore utilizzando AWS CLI o l'API RDS. Per ulteriori informazioni, consulta Visualizzazione delle date di supporto per le versioni del motore in Amazon RDS Extended Support.

Aurora e le responsabilità dei clienti con Amazon RDS Extended Support

Il seguente contenuto descrive le responsabilità di Aurora e le tue responsabilità con Extended RDS Support.

Argomenti

- Responsabilità di Amazon Aurora
- Le tue responsabilità

Responsabilità di Amazon Aurora

Dopo la data di fine del supporto standard di Aurora, Amazon Aurora fornirà patch, correzioni di bug e aggiornamenti per i motori registrati in Extended Support. RDS Ciò avverrà per un massimo di 3 anni o fino a quando non smetterai di usare i motori, a seconda dell'evento che si verifica per primo.

Le patch riguarderanno i livelli di gravità Critical e High, secondo CVEs quanto definito dai livelli di gravità del National Vulnerability Database (NVD)CVSS. Per ulteriori informazioni, consulta Vulnerability Metrics (Metriche relative alla vulnerabilità).

Le tue responsabilità

Sei responsabile dell'applicazione delle patch, delle correzioni di bug e degli aggiornamenti forniti per le globali registrati in Extended Support. RDS Amazon Aurora si riserva il diritto di modificare,

sostituire o ritirare tali patch, correzioni di bug e upgrade in qualsiasi momento. Se è necessaria una patch per risolvere problemi critici di sicurezza o stabilità, Aurora si riserva il diritto di aggiornare DB o i cluster globali con la patch o di richiedere l'installazione della patch.

Sei inoltre responsabile dell'aggiornamento del motore a una versione più recente del motore prima della RDS fine della data di Extended Support. La data di RDS fine del supporto esteso è in genere 3 anni dopo la data di fine del ciclo di vita della community con la data di . . Per la data di RDS fine dell'Extended Support per la versione principale del motore di database, vedereVersioni principali di Amazon Aurora.

Se non esegui l'upgrade del motore, dopo la data di RDS fine della data di Extended Support, Aurora tenterà di aggiornare il motore a una versione più recente del motore supportata dal supporto standard Aurora . Se l'aggiornamento non riesce, Aurora si riserva il diritto di eliminare l' DB o il cluster globale che esegue il motore oltre la data di fine del supporto standard di Aurora. Tuttavia, prima di farlo, Aurora conserverà i dati di quel motore.

Creazione Aurora DB o un cluster globale con Amazon RDS Extended Support

Quando crei Aurora DB o un cluster globale, seleziona Enable RDS Extended Support nella console oppure usa l'opzione Extended Support in o AWS CLI il parametro in. RDS API Quando registri un' Aurora o un cluster globale in Amazon Extended RDS Support, questo viene registrato permanentemente in Extended Support per tutta la durata dell' globale. RDS

Se si utilizza la console, è necessario selezionare Enable RDS Extended Support. L'impostazione non è selezionata per impostazione predefinita.

Se usi AWS CLI o RDS API e non specifichi l'impostazione RDS Extended Support, Amazon abilita per impostazione RDS predefinita Extended RDS Support. Quando si esegue l'automazione utilizzando AWS CloudFormationo altri servizi, questo comportamento predefinito mantiene la disponibilità del database oltre la data di fine del supporto standard di Aurora.

È possibile impedire la registrazione a Extended RDS Support utilizzando AWS CLIo RDSAPIper creare un cluster globale.

Argomenti

- RDSComportamento dell'Extended Support
- Considerazioni per l'RDSExtended Support

- Crea Aurora DB o un cluster RDS globale con Extended Support

RDSComportamento dell'Extended Support

La tabella seguente riassume cosa succede quando una versione principale del motore raggiunge la fine del supporto standard di Aurora.

RDSStato di Extended Support*	Comportamento
Abilitato	Amazon ti RDS addebita i costi dell'RDSExtended Support.
Disabilitato	Amazon RDS aggiorna l' DB o il cluster globale a una versione del motore supportata. Questo aggiornamento viene eseguito alla fine del supporto standard di Aurora o poco dopo.

* Nella RDS console, lo stato RDS Extended Support appare come Sì o No. In AWS CLI o RDSAPI, lo stato RDS Extended Support appare come `open-source-rds-extended-support` o`open-source-rds-extended-support-disabled`.

Considerazioni per l'RDSExtended Support

Prima di creare Aurora DB o un cluster globale, considera i seguenti elementi:

- Una volta trascorsa la data di fine del supporto standard di Aurora, puoi impedire la creazione di o di un nuovo cluster globale ed evitare i costi di Extended Support. RDS A tale scopo, usa il o il AWS CLI . RDS API Nel AWS CLI, specificare `open-source-rds-extended-support-disabled` l'`--engine-lifecycle-support`opzione. Nel RDSAPI, specificare `open-source-rds-extended-support-disabled` il `LifeCycleSupport` parametro. Se si specifica `open-source-rds-extended-support-disabled` e la data di fine del supporto standard di Aurora è trascorsa, la creazione di un' cluster Aurora DB o un cluster globale avrà sempre esito negativo.
- RDSL'Extended Support è impostato a livello di cluster. I membri di un cluster avranno sempre la stessa impostazione per RDS Extended Support `--engine-lifecycle-support` nella RDS console, in AWS CLI, e `EngineLifecycleSupport` in RDSAPI.

Per ulteriori informazioni, consulta Versioni di Amazon Aurora.

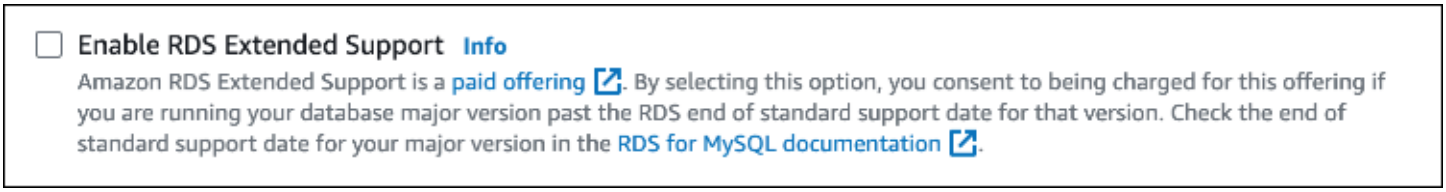
Crea Aurora DB o un cluster RDS globale con Extended Support

È possibile creare Aurora DB o un cluster globale con una versione RDS Extended Support utilizzando AWS Management Console il, il, o AWS CLI il. RDS API

Console

Quando crei un cluster Aurora DB o un cluster globale, un'istanza DB o un cluster , nella sezione Opzioni del motore, seleziona Enable Extended RDS Support. Questa impostazione non è selezionata per impostazione predefinita.

L'immagine seguente mostra l'impostazione Enable RDS Extended Support:



AWS CLI

Quando esegui il AWS CLI comando create-db-clusteror create-global-clustercreate-db-instance, seleziona RDS Extended Support specificando `open-source-rds-extended-support` l'`--engine-lifecycle-support`opzione. Per impostazione predefinita, questa opzione è impostata su. `open-source-rds-extended-support`

Per impedire la creazione di Aurora DB o di un cluster globale dopo la data di fine del supporto standard di Aurora, `open-source-rds-extended-support-disabled` specificare l'opzione. `--engine-lifecycle-support` In questo modo, eviterai i costi associati all'RDSExtended Support.

RDS API

Quando utilizzi l'RDSAPIoperazione Amazon C reateDBCluster o CreateGlobalCluster, seleziona RDS Extended Support impostando il `EngineLifecycleSupport` parametro su`open-source-rds-extended-support`. Questo parametro è impostato su `open-source-rds-extended-support` per impostazione predefinita.

Per impedire la creazione di Aurora DB o di un cluster globale dopo la data di fine del supporto standard di Aurora, `open-source-rds-extended-support-disabled` specificare il parametro. `EngineLifecycleSupport` In questo modo, eviterai i costi associati all'RDSExtended Support.

Per ulteriori informazioni, consulta i seguenti argomenti:

- Per creare un cluster Aurora DB, segui le istruzioni per il tuo motore DB in. Creazione di un cluster database Amazon Aurora
- Per creare un cluster globale, segui le istruzioni relative al tuo motore DB inCreazione di un database globale Amazon Aurora.

Visualizzazione della registrazione delle globali in Amazon RDS Extended Support

È possibile visualizzare la registrazione delle dei cluster globali in RDS Extended Support utilizzando l'API, the o RDS. AWS Management Console AWS CLI

> **Note**
>
> La colonna RDS Extended Support nella console, l'`-engine-lifecycle-support`opzione - in e il AWS CLI`EngineLifecycleSupport` parametro nell'API RDS indicano solo l'iscrizione a RDS Extended Support. I costi per RDS Extended Support iniziano solo quando la versione del motore DB ha raggiunto la fine del supporto standard Aurora. Per ulteriori informazioni, consulta Versioni principali di Amazon Aurora.
> Ad esempio, si dispone di un database Aurora PostgreSQL 11 registrato in RDS Extended Support. Il 1° aprile 2024, Amazon RDS ha iniziato ad addebitarti il supporto RDS Extended Support per questo database. Il 31 luglio 2024, hai aggiornato questo database ad Aurora PostgreSQL 12. Lo stato RDS Extended Support per questo database rimane abilitato. Tuttavia, i costi di RDS Extended Support per questo database sono stati interrotti perché Aurora PostgreSQL 12 non aveva ancora raggiunto la fine del supporto standard di Aurora. Amazon RDS non addebiterà alcun costo per l'RDS Extended Support per questo database fino al 1° marzo 2025, data in cui terminerà il supporto standard di Aurora per Aurora PostgreSQL 12.

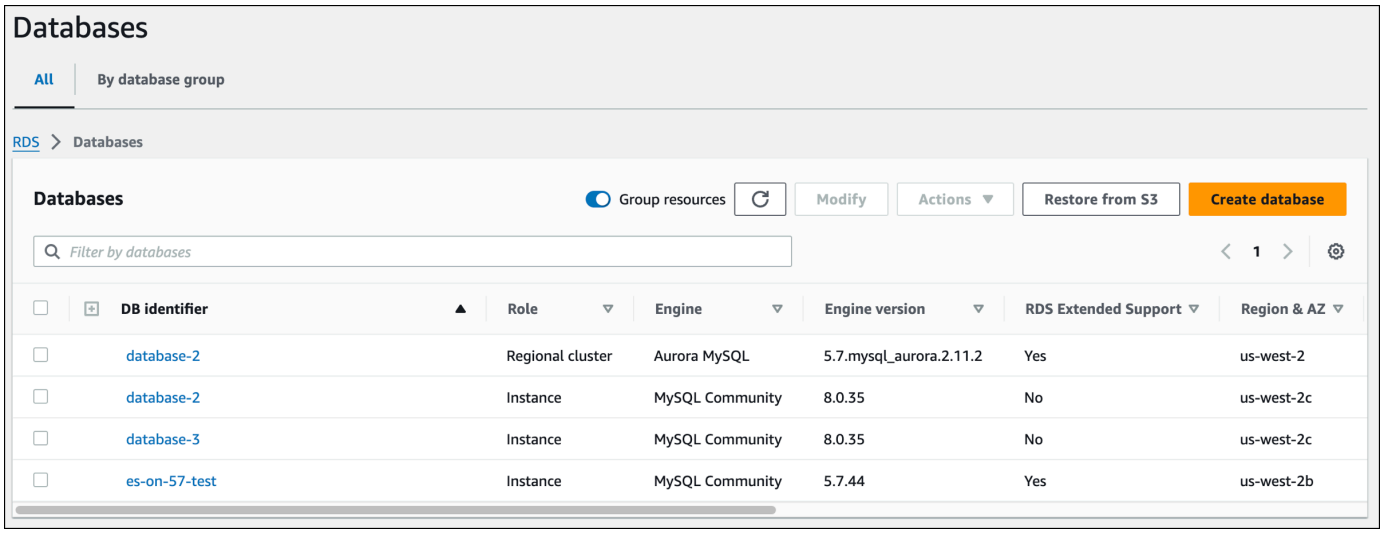
Console

Per visualizzare la registrazione delle globali in RDS Extended Support

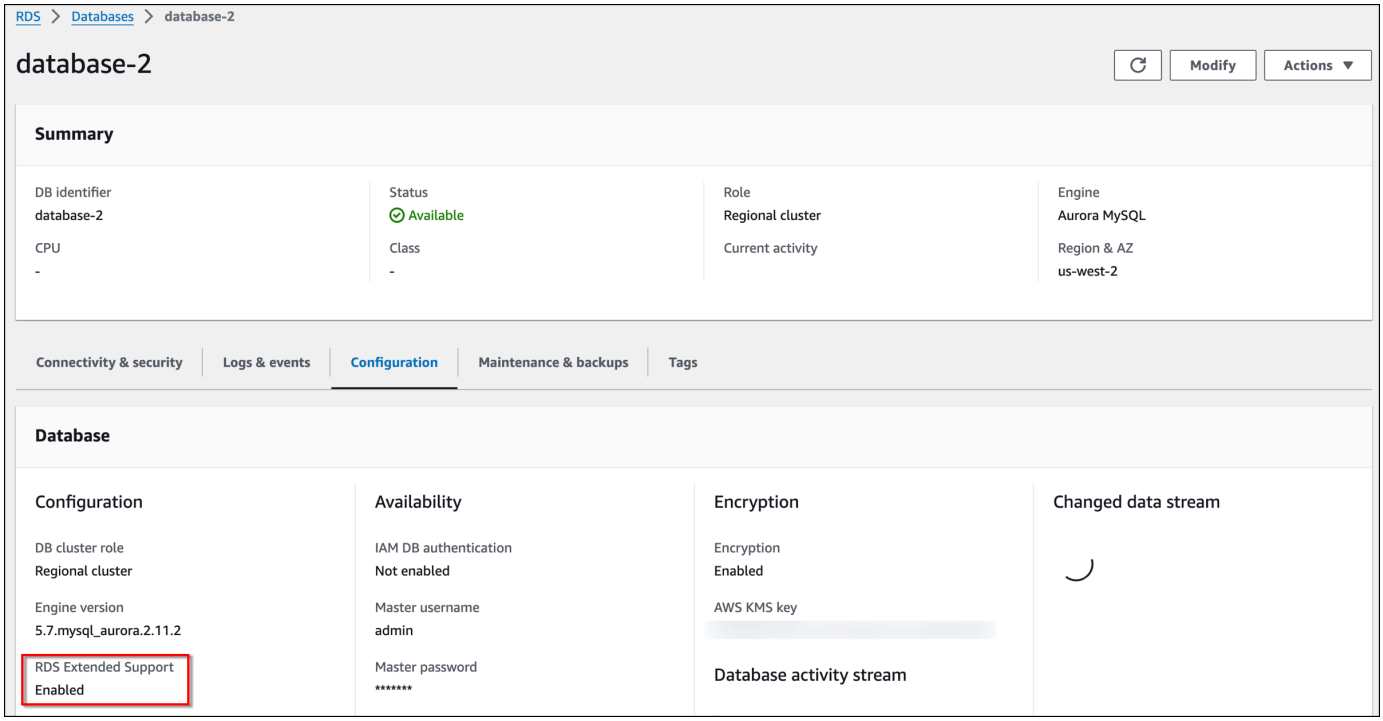
1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel riquadro di navigazione, scegli Databases (Database). Il valore in RDS Extended Support indica se Aurora DB o un cluster globale è registrato in RDS Extended Support. Se non viene visualizzato alcun valore, RDS Extended Support non è disponibile per il database.

> **Tip**
>
> Se la colonna RDS Extended Support non viene visualizzata, scegli l'icona Preferenze, quindi attiva RDS Extended Support.



3. È inoltre possibile visualizzare la registrazione nella scheda Configurazione per ogni database. Scegli un database sotto l'identificatore DB. Nella scheda Configurazione, guarda in Extended Support per vedere se il database è registrato o meno.



AWS CLI

Per visualizzare la registrazione dei database in RDS Extended Support utilizzando il comando AWS CLI, esegui il comando describe-db-clustersor describe-global-clustersdescribe-db-instancesor describe-db-clustershttps://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-clusters.html clusters).

Se RDS Extended Support è disponibile per un database, la risposta include il parametro`EngineLifecycleSupport`. Il valore `open-source-rds-extended-support` indica che Aurora DB o un cluster globale è registrato in RDS Extended Support. Il valore `open-source-rds-extended-support-disabled` indica che la registrazione dell' del cluster globale in RDS Extended Support è stata disabilitata.

Esempio

Il comando seguente restituisce informazioni per tutti i cluster Aurora DB:

```
aws rds describe-db-clusters
```

La risposta seguente mostra che un motore Aurora PostgreSQL in esecuzione sul cluster Aurora DB è registrato in RDS Extended Support: `database-1`

```
{
    "DBClusterIdentifier": "database-1",
    ...
    "Engine": "aurora-postgresql",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

API RDS

Per visualizzare la registrazione dei database in RDS Extended Support utilizzando l'API Amazon RDS, usa DescribedBInstances o DescribeDBClusters o operation. DescribeGlobalClusters

Se RDS Extended Support è disponibile per un database, la risposta include il parametro`EngineLifecycleSupport`. Il valore `open-source-rds-extended-support` indica che Aurora DB o un cluster globale è registrato in RDS Extended Support. Il valore `open-source-rds-extended-support-disabled` indica che la registrazione dell' del cluster globale in RDS Extended Support è stata disabilitata.

Visualizzazione delle date di supporto per le versioni del motore in Amazon RDS Extended Support

Puoi visualizzare informazioni sulle date di supporto per le versioni del motore per le tue Aurora DB o cluster globali in Amazon RDS Extended Support utilizzando o l'API RDS. AWS CLI Queste informazioni possono aiutarti a pianificare gli upgrade.

AWS CLI i comandi e le operazioni dell'API RDS restituiscono le date di inizio e fine per il supporto standard di e il supporto esteso di RDS. Queste date sono disponibili anche nelle tabelle delle versioni principali del motore. Per ulteriori informazioni, consultaVersioni principali di Amazon Aurora.

AWS CLI

Per visualizzare le date di inizio e di fine del supporto standard di Aurora e del supporto esteso di RDS per le versioni principali del motore AWS CLI utilizzando, describe-db-major-engineesegui il comando -versions.

Questo comando restituisce i seguenti parametri pertinenti:

- `SupportedEngineLifecycles`— Questo parametro è una matrice di oggetti che include `LifecycleSupportName`,`LifecycleSupportStartDate`, e`LifecycleSupportEndDate`.

- `LifecycleSupportName`— Questo parametro indica il tipo di supporto utilizzato dalla versione del motore: Aurora standard support `open-source-rds-standard-support` () o RDS Extended Support `open-source-rds-extended-support` ().
- `LifecycleSupportStartDate`— Questo parametro elenca la data di inizio del supporto standard di Aurora o del supporto RDS Extended per la versione principale del motore, a seconda del valore di. `LifecycleSupportName`
- `LifecycleSupportEndDate`— Questo parametro elenca la data di fine del supporto standard di Aurora o del supporto RDS Extended per la versione principale del motore, a seconda del valore di. `LifecycleSupportName`

Esempio

L'esempio di risposta mostra le date di inizio e fine per i cicli di vita del motore supportati `open-source-rds-standard-support` e `open-source-rds-extended-support` per Aurora MySQL versione 2 (MySQL 5.7). RDS Extended Support è disponibile per Aurora MySQL versione 2 (MySQL 5.7).

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "aurora-mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2018-02-06T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-10-31T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-11-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        ...
    ]
}
```

API RDS

Per visualizzare le date di inizio e fine del supporto standard di Aurora e del supporto RDS Extended per le versioni principali del motore utilizzando l'API RDS, utilizza l'operazione Descrivi. DBMajor EngineVersions

Se RDS Extended Support è disponibile per una versione del motore, la risposta include il parametro `SupportedEngineLifeCycles` come matrice con due oggetti. Un oggetto include le date di inizio e fine per il supporto Aurora. Il secondo oggetto include le date di inizio e fine di RDS Extended Support.

Se RDS Extended Support non è disponibile per una versione del motore, la risposta include solo il parametro `SupportedEngineLifeCycles` come array con un singolo oggetto. Questo oggetto include le date di inizio e fine per il supporto standard di Aurora.

Ripristino di Aurora DB o di un cluster globale con Amazon Extended Support RDS

Quando ripristini Aurora DB o un cluster globale, seleziona Enable RDS Extended Support nella console oppure utilizza l'opzione Extended Support in o AWS CLI il parametro in. RDS API Quando si registra un' Aurora DB o un cluster globale in Extended RDS Support, questo viene registrato permanentemente in RDS Extended Support per tutta la durata dell' globale.

L'impostazione predefinita per l'impostazione RDS Extended Support dipende dal fatto che si utilizzi la console AWS CLI, il o il RDS API per ripristinare il database. Se usi AWS CLI o RDS API e non specifichi l'impostazione RDS Extended Support, Amazon abilita per impostazione RDS predefinita Extended RDS Support. Quando si esegue l'automazione utilizzando AWS CloudFormationo altri servizi, questo comportamento predefinito mantiene la disponibilità del database oltre la data di fine del supporto standard di Aurora. È possibile disabilitare RDS Extended Support utilizzando AWS CLI o il RDSAPI.

Argomenti

- RDSComportamento dell'Extended Support
- Considerazioni per l'RDSExtended Support
- Ripristina del cluster Aurora DB o un cluster RDS globale con Extended Support

RDSComportamento dell'Extended Support

La tabella seguente riassume cosa succede quando una versione del motore principale di Aurora DB o un cluster globale che si sta ripristinando ha raggiunto la fine del supporto standard di Aurora .

RDSStato di Extended Support*	Comportamento
Abilitato	Amazon ti RDS addebita i costi dell'RDSExtended Support.
	Al termine del ripristino, Amazon aggiorna RDS automaticamente l' DB o il cluster globale a una versione più recente del motore (in una finestra di manutenzione futura).

* Nella RDS console, lo stato RDS Extended Support appare come Sì o No. In AWS CLI o RDSAPI, lo stato RDS Extended Support appare come `open-source-rds-extended-support` o`open-source-rds-extended-support-disabled`.

Considerazioni per l'RDSExtended Support

Prima di ripristinare Aurora DB o un cluster globale, considera i seguenti elementi:

- Una volta trascorsa la data di fine del supporto standard di Aurora, se desideri ripristinare un' cluster Aurora DB o un cluster globale da Amazon S3, puoi farlo solo utilizzando o il. AWS CLI RDS API Utilizzare l'`--engine-lifecycle-support`opzione nel AWS CLI comando restore-db-cluster-from-s3 o il `EngineLifecycleSupport` parametro nell'operazione R estoreDBCluster RDS API fromS3.
- Se si desidera impedire ad Aurora di ripristinare i database alle versioni Extended RDS Support, specificare `open-source-rds-extended-support-disabled` in o. AWS CLI RDS API In questo modo, eviterai i costi associati all'RDSExtended Support.

  Se specifichi questa impostazione, Aurora aggiornerà automaticamente il database ripristinato a una versione principale più recente e supportata. Se l'upgrade non supera i controlli precedenti all'aggiornamento, Aurora ripristinerà in modo sicuro la versione del motore RDS Extended Support. Questo database rimarrà in modalità RDS Extended Support e Aurora addebiterà il costo del RDS supporto Extended Support fino all'aggiornamento manuale del database.

- RDSL'Extended Support è impostato a livello di cluster. I membri di un cluster avranno sempre la stessa impostazione per RDS Extended Support `--engine-lifecycle-support` nella RDS console, in AWS CLI, e `EngineLifecycleSupport` in RDSAPI.

Per ulteriori informazioni, consulta Versioni di Amazon Aurora.

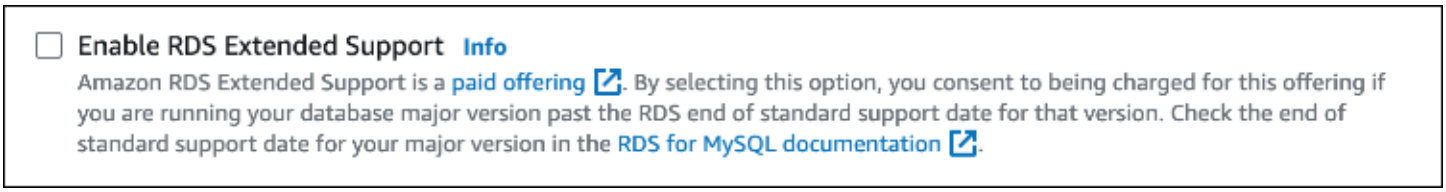
Ripristina del cluster Aurora DB o un cluster RDS globale con Extended Support

È possibile ripristinare Aurora DB o un cluster globale con una versione RDS Extended Support utilizzando AWS Management Console il, il, o AWS CLI il. RDS API

Console

Quando ripristini un cluster Aurora DB o un cluster globale, un'istanza DB o un cluster Extended RDS Support nella sezione Opzioni del motore. Se non selezioni questa impostazione e la versione principale del motore che stai ripristinando ha superato la fine del supporto standard di Aurora, Aurora aggiorna automaticamente il cluster RDS Aurora DB o l' a una versione con supporto standard Aurora.

L'immagine seguente mostra l'impostazione Enable RDS Extended Support:



AWS CLI

Quando esegui il AWS CLI comando restore-db-cluster-from-snapshot restore-db-instance-from-db-snapshot , seleziona RDS Extended Support specificando l'opzione. `open-source-rds-extended-support` `--engine-lifecycle-support`

Se desideri evitare gli addebiti associati all'RDSExtended Support, imposta l'`--engine-lifecycle-support`opzione su`open-source-rds-extended-support-disabled`. Per impostazione predefinita, questa opzione è impostata su`open-source-rds-extended-support`.

È inoltre possibile specificare questo valore utilizzando i seguenti AWS CLI comandi:

- restore-db-cluster-from-s3

- restore-db-cluster-to-point-in-time

RDS API

Quando utilizzi l'RDSAPIoperazione R estoreDBCluster FromSnapshot R estoreDBInstance Amazon, seleziona RDS Extended Support impostando il `EngineLifecycleSupport` parametro su`open-source-rds-extended-support`.

Per evitare gli addebiti associati a RDS Extended Support, imposta il `EngineLifecycleSupport` parametro su`open-source-rds-extended-support-disabled`. Questo parametro è impostato su `open-source-rds-extended-support` per impostazione predefinita.

È inoltre possibile specificare questo valore utilizzando le seguenti RDS API operazioni:

- R di estoreDBCluster S3
- RestoreDBClusterToPointInTime

Per ulteriori informazioni sul ripristino di un cluster Aurora DB, segui le istruzioni per il tuo motore DB in. Backup e ripristino di un cluster DB Amazon Aurora

Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database

Un'implementazione blu/verde copia un ambiente di database di produzione in un ambiente di gestione temporanea separato e sincronizzato. Utilizzando Aurora Blue/Green Deployments, puoi apportare modifiche al database nell'ambiente di staging senza influire sull'ambiente di produzione. Ad esempio, puoi aggiornare la versione principale o secondaria del motore DB o modificare i parametri del database nell'ambiente di staging. Quando sei pronto, puoi promuovere l'ambiente di staging come nuovo ambiente di database di produzione, con tempi di inattività in genere inferiori a un minuto.

Amazon Aurora crea l'ambiente di staging clonando il volume di storage Aurora sottostante nell'ambiente di produzione. Il volume del cluster nell'ambiente di staging memorizza solo le modifiche incrementali apportate a quell'ambiente.

> **Note**
>
> Attualmente, le implementazioni Blue/Green sono supportate solo per RDS Aurora MySQL e Aurora PostgreSQL. Per la disponibilità del motore Amazon RDS, consulta Using Amazon RDS Blue/Green Deployments per gli aggiornamenti del database nella Amazon RDS User Guide.

Argomenti

- Panoramica delle implementazioni di Amazon Aurora Blue/Green
- Creazione di una distribuzione blu/verde in Amazon
- Visualizzazione di una distribuzione blu/verde in Amazon
- Cambiare una distribuzione blu/verde in Amazon
- Eliminazione di una distribuzione blu/verde in Amazon RDS

Panoramica delle implementazioni di Amazon Aurora Blue/Green

Utilizzando Aurora Blue/Green Deployments, puoi apportare e testare modifiche al database prima di implementarle in un ambiente di produzione. Un'implementazione blu/verde crea un ambiente di gestione temporanea che copia l'ambiente di produzione. In un'implementazione blu/verde, l'ambiente blu è l'ambiente di produzione corrente. L'ambiente verde è l'ambiente di staging e rimane sincronizzato con l'ambiente di produzione corrente.

È possibile apportare modifiche al cluster di database Aurora nell'ambiente verde senza influire sui carichi di lavoro di produzione. Ad esempio, è possibile aggiornare la versione principale o secondaria del motore di database o modificare i parametri di database nell'ambiente di gestione temporanea. È possibile testare le modifiche nell'ambiente verde. Quando sei pronto, puoi passare da un ambiente all'altro per trasformare l'ambiente verde nel nuovo ambiente di produzione. Lo switchover richiede in genere meno di un minuto senza perdita di dati e senza la necessità di modificare l'applicazione.

Poiché l'ambiente verde è una copia della topologia dell'ambiente di produzione, il cluster di database e tutte le relative istanze database vengono copiati nell'implementazione. L'ambiente verde include anche le funzionalità utilizzate dal cluster DB, come le istantanee del cluster DB, Performance Insights, Enhanced Monitoring e Aurora Serverless v2.

> **Note**
>
> Le implementazioni Blue/Green sono supportate per Aurora MySQL e Aurora PostgreSQL. Per la disponibilità di Amazon RDS, consulta Panoramica delle distribuzioni Blue/Green di Amazon RDS nella Guida per l'utente di Amazon RDS.

Argomenti

- Disponibilità di regioni e versioni
- Vantaggi dell'utilizzo delle implementazioni blu/verde Amazon RDS
- Flusso di lavoro di un'implementazione blu/verde
- Autorizzazione dell'accesso alle operazioni di Amazon Aurora
- Limitazioni e considerazioni per le distribuzioni blu/verdi di Amazon Aurora
- Best practice per le distribuzioni Amazon Aurora

Disponibilità di regioni e versioni

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni, consulta the section called “Distribuzioni blu/verdi”.

Vantaggi dell'utilizzo delle implementazioni blu/verde Amazon RDS

Utilizzando Amazon RDS, le Blue/Green Deployments, you can stay current on security patches, improve database performance, and adopt newer database features with short, predictable downtime. Blue/green implementazioni riducono i rischi e i tempi di inattività per gli aggiornamenti del database, come gli aggiornamenti principali o secondari delle versioni del motore.

Le implementazioni blu/verde offrono i seguenti vantaggi:

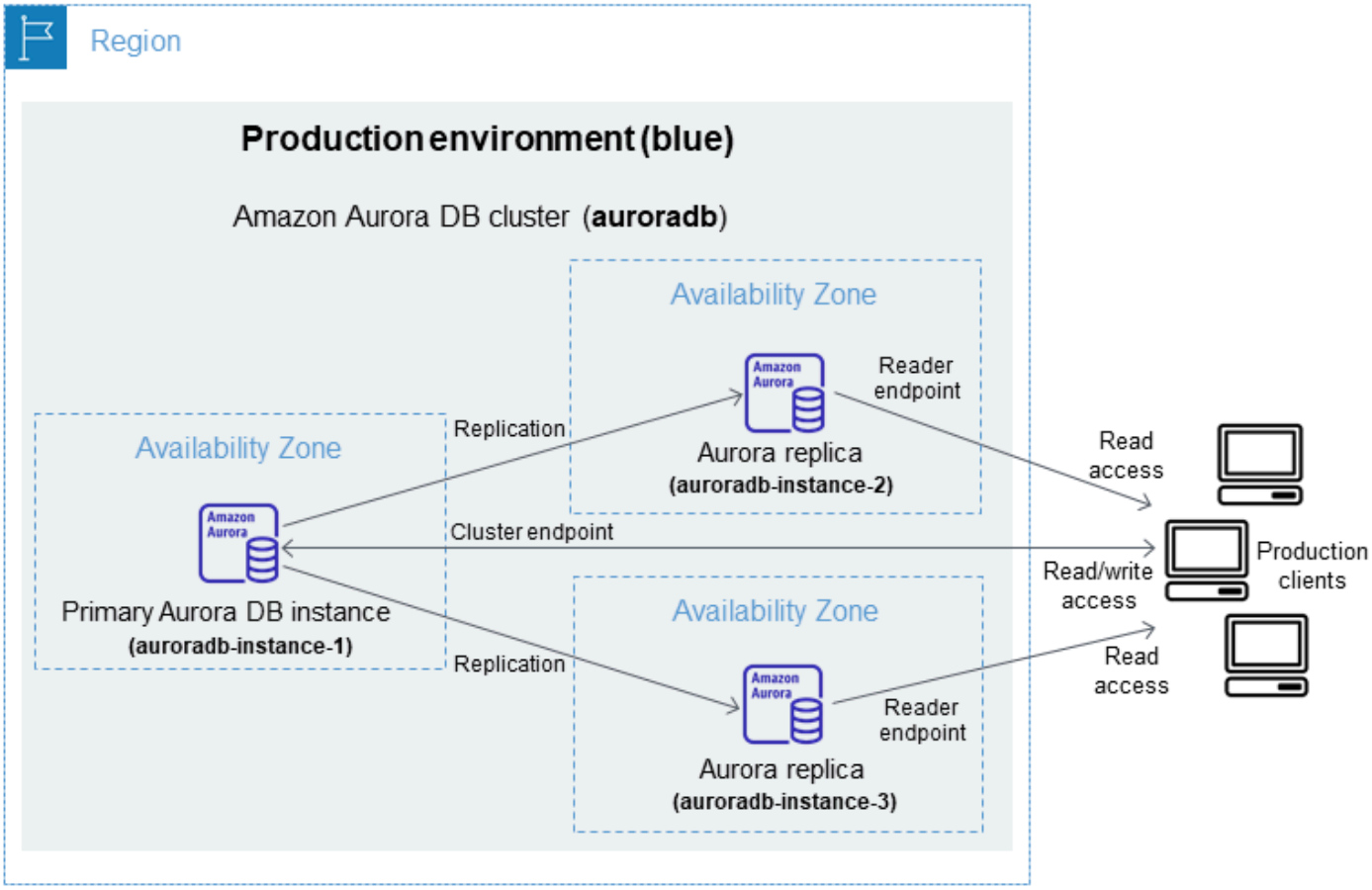
- Crea facilmente un ambiente di gestione temporanea pronto per la produzione.
- Replica automaticamente le modifiche del database dall'ambiente di produzione all'ambiente di gestione temporanea.
- Esegui il test delle modifiche del database in un ambiente di gestione temporanea sicuro, senza influire sull'ambiente di produzione.
- Rimani aggiornato con le patch del database e gli aggiornamenti di sistema.
- Implementa ed esegui il test delle nuove funzionalità del database.
- Esegui lo switchover dell'ambiente di gestione temporanea in un nuovo ambiente di produzione senza modificare l'applicazione.
- Esegui lo switchover in sicurezza usando i guardrail di switchover integrati.
- Elimina la perdita di dati durante lo switchover.
- Esegui lo switchover rapidamente, in genere in meno di un minuto a seconda del carico di lavoro.

Flusso di lavoro di un'implementazione blu/verde

Completa i seguenti passaggi principali quando utilizzi un'implementazione blu/verde per gli aggiornamenti del cluster di database Aurora.

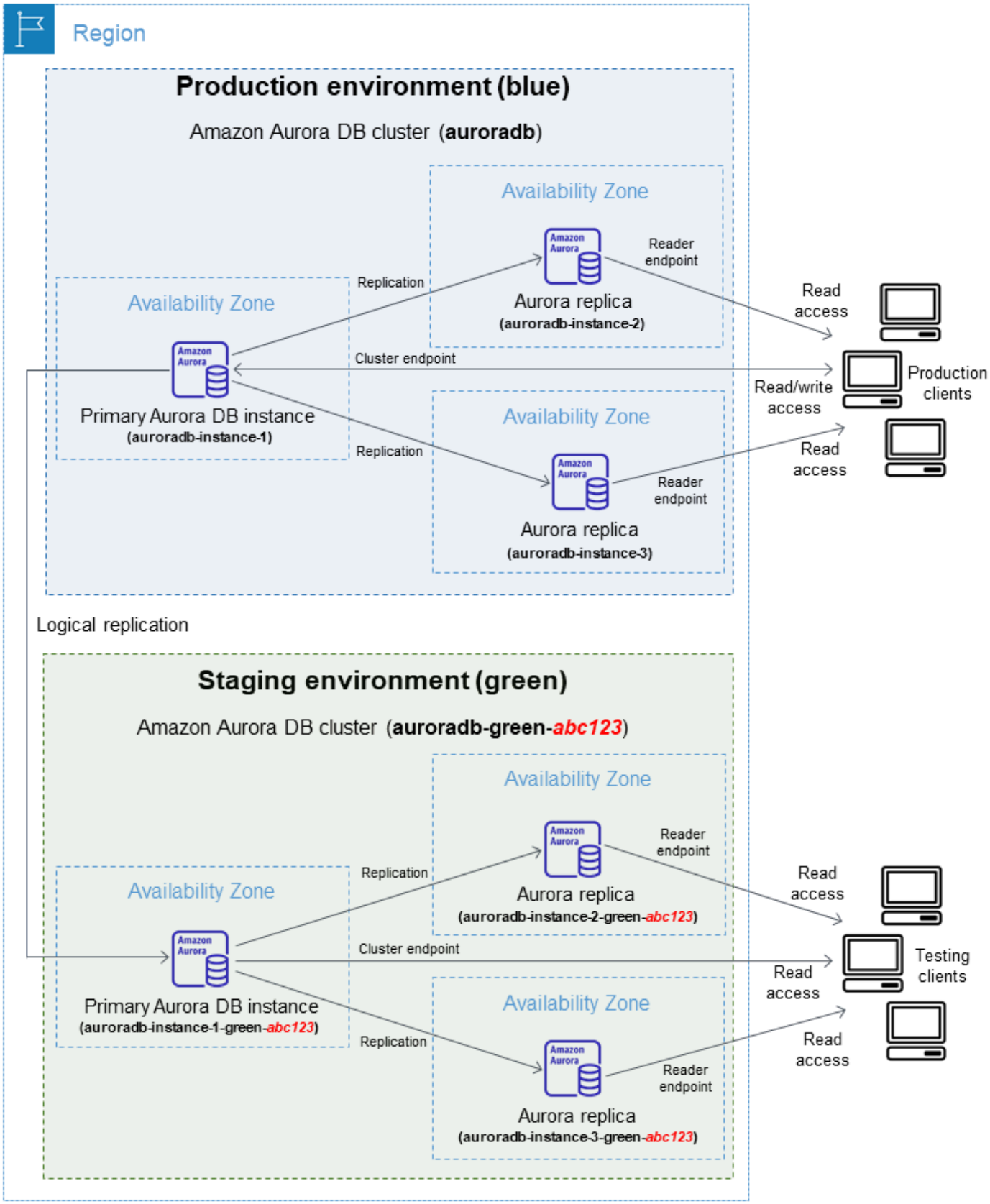
1. Identifica un cluster di database di produzione che richieda aggiornamenti.

   L'immagine seguente mostra un esempio di cluster di database di produzione.



2. Crea l'implementazione blu/verde. Per istruzioni, consultare Creazione di una distribuzione blu/verde in Amazon .

   L'immagine seguente mostra un esempio di blue/green deployment of the production environment from step 1. While creating the blue/green implementazione, RDS copia la topologia e la configurazione complete del cluster Aurora DB per creare l'ambiente verde. Ai nomi del cluster di database e delle istanze database copiati viene aggiunto `-green-`*`random-characters`*. L'ambiente di staging nell'immagine contiene il cluster DB (auroradb-green-). *`abc123`* Contiene anche le tre istanze DB nel cluster DB (auroradb-instance1-green-, auroradb-instance2-green- e auroradb-instance3-green-). *`abc123 abc123 abc123`*


Region
Production environment (blue)
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2)
Replication
Availability Zone
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Logical replication
Staging environment (green)
Amazon Aurora DB cluster (auroradb-green-abc123)
Availability Zone
Amazon Aurora
Reader endpoint
Aurora replica
(auroradb-instance-2-green-abc123)
Replication
Availability Zone
Read access
Cluster endpoint
Testing clients
Read access
Primary Aurora DB instance
(auroradb-instance-1-green-abc123)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3-green-abc123)

Quando si crea l'implementazione blu/verde, è possibile specificare una versione successiva del motore di database e un gruppo di parametri del·cluster di database diverso per il cluster di database nell'ambiente verde. È anche possibile specificare un gruppo di parametri database diverso per le istanze database nel cluster di database.

RDS configura anche la replica dall'istanza database primaria dell'ambiente blu all'istanza database primaria dell'ambiente verde.

> **Important**
>
> Per Aurora MySQL versione 3, dopo aver creato la distribuzione blu/verde, il cluster DB nell'ambiente verde non consente operazioni di scrittura per impostazione predefinita. Tuttavia, ciò non si applica agli utenti che dispongono del `CONNECTION_ADMIN` privilegio, incluso l'utente master Aurora. Gli utenti con questo privilegio possono ignorare il comportamento. `read_only` Per ulteriori informazioni, consulta Privilegio basato sui ruoli.

3. Apporta le modifiche all'ambiente di gestione temporanea.

   Ad esempio, è possibile modificare la classe di istanza DB utilizzata da una o più istanze DB nell'ambiente verde.

   Per ulteriori informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

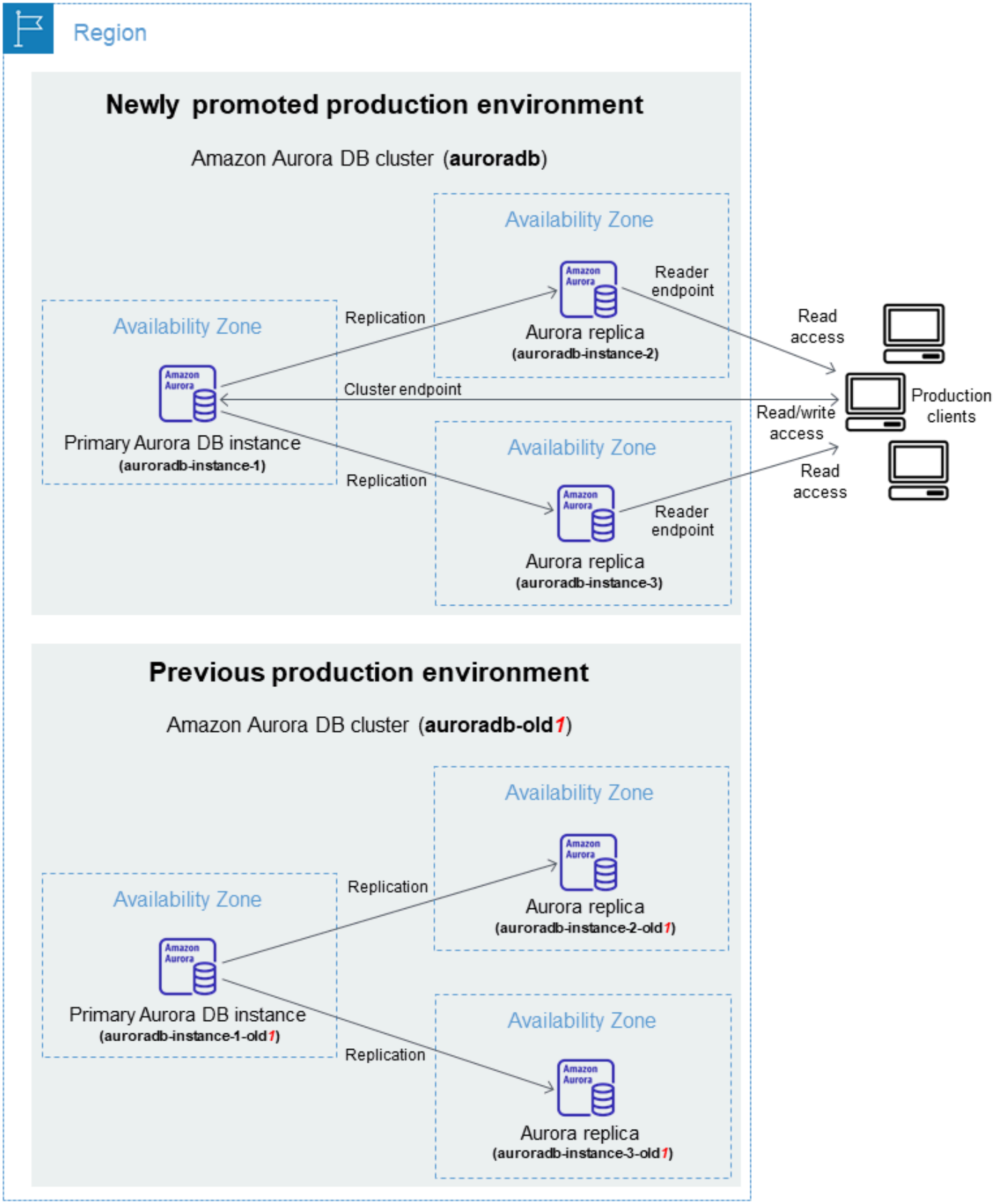
4. Esegui il test dell'ambiente di gestione temporanea.

   Durante i test, ti consigliamo di mantenere i database in un ambiente verde di sola lettura. Abilita le operazioni di scrittura sull'ambiente verde con cautela perché possono causare conflitti di replica. Possono inoltre generare dati non previsti nei database di produzione dopo lo switchover. Per abilitare le operazioni di scrittura per Aurora MySQL, imposta il `read_only` parametro su, quindi riavvia l'istanza DB. `0` Per Aurora PostgreSQL, imposta il parametro su a livello di sessione. `default_transaction_read_only` `off`

5. Quando sei pronto, passa alla transizione dall'ambiente di staging al nuovo ambiente di produzione. Per istruzioni, consultare Cambiare una distribuzione blu/verde in Amazon .

   Lo switchover comporta tempi di inattività. I tempi di inattività sono in genere inferiori al minuto, ma possono essere più lunghi a seconda del carico di lavoro.

   L'immagine seguente mostra i cluster di database dopo lo switchover.


Region
Newly promoted production environment
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Availability Zone
Replication
Aurora replica
(auroradb-instance-2)
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Previous production environment
Amazon Aurora DB cluster (auroradb-old1)
Availability Zone
Availability Zone
Replication
Aurora replica
(auroradb-instance-2-old1)
Primary Aurora DB instance
(auroradb-instance-1-old1)
Availability Zone
Replication
Aurora replica
(auroradb-instance-3-old1)

Dopo lo switchover, il cluster di database Aurora dell'ambiente verde diventa il nuovo cluster di database di produzione. I nomi e gli endpoint dell'ambiente di produzione corrente vengono assegnati all'ambiente di produzione appena convertito, senza che sia necessario apportare modifiche all'applicazione. Di conseguenza, il traffico di produzione ora viene indirizzato al nuovo ambiente di produzione. Il cluster di database e le istanze database nell'ambiente blu vengono rinominati aggiungendo `-old`*n* al nome corrente, dove *n* è un numero. Ad esempio, supponi che il nome dell'istanza database nell'ambiente blu sia `auroradb-instance-1`. Dopo lo switchover, il nome dell'istanza database diventa `auroradb-instance-1-old1`.

Nell'esempio dell'immagine, durante lo switchover si verificano le seguenti modifiche:

- Il cluster di database dell'ambiente verde `auroradb-green-abc123` diventa il cluster di database di produzione denominato `auroradb`.
- L'istanza database dell'ambiente verde denominata `auroradb-instance1-green-abc123` diventa l'istanza database di produzione `auroradb-instance1`.
- L'istanza database dell'ambiente verde denominata `auroradb-instance2-green-abc123` diventa l'istanza database di produzione `auroradb-instance2`.
- L'istanza database dell'ambiente verde denominata `auroradb-instance3-green-abc123` diventa l'istanza database di produzione `auroradb-instance3`.
- Il cluster di database dell'ambiente blu denominato `auroradb` diventa `auroradb-old1`.
- L'istanza database dell'ambiente blu denominata `auroradb-instance1` diventa `auroradb-instance1-old1`.
- L'istanza database dell'ambiente blu denominata `auroradb-instance2` diventa `auroradb-instance2-old1`.
- L'istanza database dell'ambiente blu denominata `auroradb-instance3` diventa `auroradb-instance3-old1`.

6. Se non hai più bisogno di un'implementazione blu/verde, puoi eliminarla. Per istruzioni, consultare Eliminazione di una distribuzione blu/verde in Amazon RDS .

   Dopo lo switchover, l'ambiente di produzione precedente non viene eliminato, quindi è possibile utilizzarlo per i test di regressione, se necessario.

Autorizzazione dell'accesso alle operazioni di Amazon Aurora

Gli utenti devono disporre delle autorizzazioni necessarie per eseguire operazioni relative alle implementazioni blu/verde. Puoi creare le policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse indicate di cui hanno bisogno. Puoi quindi collegare tali policy ai ruoli o ai set di autorizzazioni IAM che richiedono le autorizzazioni. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.

L'utente che crea un'implementazione blu/verde deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterEndpoint`

L'utente che esegue lo switchover a un'implementazione blu/verde deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBCluster`
- `rds:PromoteReadReplicaDBCluster`

L'utente che elimina un'implementazione blu/verde deve disporre delle autorizzazioni per eseguire le seguenti operazioni RDS:

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBCluster`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterEndpoint`

Aurora effettua il provisioning e modifica le risorse nell'ambiente di staging per tuo conto. Queste risorse includono istanze DB che utilizzano una convenzione di denominazione definita internamente. Pertanto, le policy IAM allegate non possono contenere modelli di nomi di risorse parziali come. `my-`

`db-prefix-*` Sono supportati solo i caratteri jolly (*). In generale, consigliamo di utilizzare i tag delle risorse e altri attributi supportati per controllare l'accesso a queste risorse, anziché i caratteri jolly. Per ulteriori informazioni, consulta Azioni, risorse e chiavi di condizione per Amazon RDS.

Limitazioni e considerazioni per le distribuzioni blu/verdi di Amazon Aurora

Le implementazioni blu/green in Amazon RDS richiedono un'attenta considerazione di fattori come gli slot di replica, la gestione delle risorse, il dimensionamento delle istanze e i potenziali impatti sulle prestazioni del database. Le sezioni seguenti forniscono indicazioni per aiutarti a ottimizzare la tua strategia di distribuzione per garantire tempi di inattività minimi, transizioni senza interruzioni e una gestione efficace dell'ambiente di database.

Argomenti

- Limitazioni per le implementazioni blu/verde
- Considerazioni sulle implementazioni blu/verde

Limitazioni per le implementazioni blu/verde

Le seguenti limitazioni si applicano alle implementazioni blu/verde.

Argomenti

- Limitazioni generali per le implementazioni blu/verde
- Limitazioni di Aurora MySQL per MySQL per implementazioni blu/verdi
-

Limitazioni generali per le implementazioni blu/verde

Le seguenti limitazioni generali si applicano alle implementazioni blu/verde:

- Non è possibile interrompere e avviare un cluster che fa parte di una distribuzione blu/verde.
- Le distribuzioni blu/verde non supportano la gestione delle password degli utenti principali con. AWS Secrets Manager
- Se si tenta di forzare un backtrack sul cluster DB blu, la distribuzione blu/verde si interrompe e lo switchover viene bloccato.
- Durante il passaggio, gli ambienti blu e verdi non possono avere integrazioni Zero-ETL con Amazon Redshift. Occorre prima eliminare l'integrazione ed eseguire il passaggio, quindi ricreare l'integrazione.

- Il pianificatore eventi (parametro `event_scheduler`) deve essere disabilitato nell'ambiente verde quando si crea un'implementazione blu/verde. Ciò impedisce che si generino eventi nell'ambiente verde e conseguentemente incongruenze.
- Le politiche di Auto Scaling configurate sul cluster DB blu non vengono copiate nell'ambiente verde. È necessario riconfigurarle dopo lo switchover, indipendentemente dal fatto che siano state inizialmente configurate nell'ambiente blu o verde.
- Non è possibile modificare un cluster di database decrittografato in un cluster di database crittografato.
- Non è possibile modificare un cluster di DB blu con una versione del motore superiore rispetto al corrispondente cluster di DB verde.
- Le risorse nell'ambiente blu e nell'ambiente verde devono trovarsi nello stesso Account AWS.
- Le implementazioni blu/verde non sono supportate per le seguenti funzionalità:
  - Server proxy per Amazon RDS
  - Repliche di lettura tra regioni diverse
  - Cluster di database Aurora Serverless v1
  - Cluster database che fanno parte di un database globale Aurora
  - AWS CloudFormation

Limitazioni di Aurora MySQL per MySQL per implementazioni blu/verdi

Le seguenti limitazioni si applicano alle implementazioni blu/verdi di Aurora MySQL RDS per MySQL

- Il cluster DB di origine non può contenere alcun database denominato. `tmp` I database con questo nome non verranno copiati nell'ambiente verde.
- Il cluster di DB blu non può essere una replica binlog esterna.
- Se il cluster DB di origine con backtrack è abilitato, il cluster DB verde viene creato senza supporto per il backtrack. Questo perché il backtracking non funziona con la replica dei log binari (binlog), necessaria per le implementazioni blu/green. Per ulteriori informazioni, consulta the section called “Backtrack di un cluster database”.
- Le distribuzioni blu/green non supportano il driver AWS JDBC per MySQL. Per ulteriori informazioni, consulta Limitazioni note su. GitHub

- Le tabelle non registrate non vengono replicate nell'ambiente verde a meno che il `rds.logically_replicate_unlogged_tables` parametro non sia impostato su nel cluster DB blu. 1 Non modificate il valore di questo parametro dopo aver creato una distribuzione blu/verde per evitare possibili errori di replica su tabelle non registrate.
- Il cluster di DB blu non può essere una fonte logica (editore) o una replica (sottoscrittore).
- Se il cluster di database blu è configurato come server esterno di un'estensione FDW (Foreign Data Wrapper), è necessario utilizzare il nome dell'endpoint del cluster anziché gli indirizzi IP. Ciò consente alla configurazione di rimanere funzionale dopo lo switchover.
- In una distribuzione blu/verde, ogni database richiede uno slot di replica logico. Con l'aumentare del numero di database, il sovraccarico di risorse aumenta e può potenzialmente causare ritardi nella replica, soprattutto se l'istanza DB del cluster non è sufficientemente scalata. L'impatto dipende da fattori quali il carico di lavoro del database e il numero di connessioni.
- Le distribuzioni blu/green sono supportate per Babelfish for Aurora PostgreSQL solo per la versione 15.7 e successive 15 versioni e 16.3 e 16 versioni successive.
- Se desideri acquisire i piani di esecuzione nelle repliche di Aurora, devi fornire l'endpoint blu del cluster di database quando chiami la funzione `apg_plan_mgmt.create_replica_plan_capture`. Ciò garantisce che le acquisizioni dei piani continuino a funzionare dopo lo switchover. Per ulteriori informazioni, consulta the section called “Acquisizione dei piani di esecuzione Aurora PostgreSQL nelle repliche”.
- Il processo di applicazione della replica logica nell'ambiente verde è a thread singolo. Se l'ambiente blu genera un volume elevato di traffico di scrittura, l'ambiente verde potrebbe non essere in grado di tenere il passo. Ciò può causare ritardi o errori di replica, in particolare per i carichi di lavoro che producono un throughput di scrittura continuo elevato. Assicurati di testare a fondo i tuoi carichi di lavoro. Per gli scenari che richiedono aggiornamenti di versione importanti e la gestione di carichi di lavoro di scrittura ad alto volume, prendi in considerazione approcci alternativi come l'utilizzo di AWS Database Migration Service (AWS DMS)o la replica logica autogestita.
- Alle estensioni PostgreSQL si applicano le seguenti limitazioni:
  - L'`pg_partman`estensione deve essere disabilitata nell'ambiente blu quando si crea una distribuzione blu/verde. L'estensione esegue operazioni DDL come `CREATE TABLE` che interrompono la replica logica dall'ambiente blu all'ambiente verde.
  - L'estensione `pg_cron` deve rimanere disabilitata su tutti i database verdi dopo la creazione dell'implementazione blu/verde. L'estensione dispone di worker in background che vengono eseguiti come superutente e aggirano l'impostazione di sola lettura dell'ambiente verde, il che potrebbe causare conflitti di replica.

- L'estensione `apg_plan_mgmt` deve avere il parametro `apg_plan_mgmt.capture_plan_baselines` impostato su `off` in tutti i database verdi per evitare conflitti di chiave primaria se un piano identico viene acquisito nell'ambiente blu. Per ulteriori informazioni, consulta the section called “Panoramica della gestione del piano di query per Aurora PostgreSQL”.
- Le estensioni `pglogical` e `pgactive` devono essere disabilitate nell'ambiente blu quando si crea un'implementazione blu/verde. Dopo aver convertito l'ambiente verde nel nuovo ambiente di produzione, puoi abilitare nuovamente le estensioni. Inoltre, il database blu non può essere un abbonato logico di un'istanza esterna.
- Se utilizzi l'`pgAudit`estensione, deve rimanere nelle librerie condivise (`shared_preload_libraries`) nei gruppi di parametri DB personalizzati sia per le istanze DB blu che per quelle verdi. Per ulteriori informazioni, consulta the section called “Configurazione dell' pgAudit estensione”.

Limitazioni specifiche della replica logica per le distribuzioni blu/verdi

PostgreSQL presenta alcune restrizioni relative alla replica logica, che si traducono in limitazioni durante la creazione di implementazioni blu/verdi per i cluster di database Aurora PostgreSQL.

La tabella seguente descrive le limitazioni della replica logica che si applicano alle implementazioni blu/verde per Aurora PostgreSQL. Per ulteriori informazioni, consulta Restrictions nella documentazione della replica logica di PostgreSQL.

Limitazione	Spiegazione
Le istruzioni DDL (Data Definitio n Language), come `CREATE TABLE` e `CREATE SCHEMA`, non vengono replicate dall'ambiente blu nell'ambiente verde.	Se Aurora rileva una modifica DDL nell'ambiente blu, i database verdi entrano nello stato di replica degradata. È necessario eliminare l'impleme ntazione blu/verde e tutti i database verdi, quindi ricrearla.

Limitazione	Spiegazione
Le operazioni `NEXTVAL` sugli oggetti della sequenza non sono sincronizzate tra l'ambiente blu e l'ambiente verde.	Durante lo switchover, Aurora incrementa i valori di sequenza nell'ambiente verde in modo che corrispondano a quelli nell'ambiente blu. Se hai migliaia di sequenze, lo switchover può subire un ritardo.
Gli oggetti di grandi dimensioni nell'ambiente blu non vengono replicati nell'ambiente verde. Ciò include sia gli oggetti di grandi dimensioni esistenti che tutti gli oggetti di grandi dimensioni appena creati o modificati durante il processo di distribuzione blu/verde.	Se Aurora rileva la creazione o la modifica di oggetti di grandi dimensioni nell'ambiente blu archiviati nella tabella di sistema `pg_largeobject` , i database verdi entrano nello stato di replica degradata. È necessario eliminare l'implementazione blu/verde e tutti i database verdi, quindi ricrearla.
L'aggiornamento delle viste materializzate interrompe la replica.	L'aggiornamento delle viste materializzate nell'ambiente blu interrompe la replica nell'ambiente verde. Evita di aggiornare le viste materializzate nell'ambiente blu. Dopo il passaggio, è possibile aggiornarle manualmente utilizzando il comando REFRESH MATERIALIZED VIEW o pianificare un aggiornamento.

Limitazione	Spiegazione
Le operazioni UPDATE e DELETE non sono consentite nelle tabelle che non dispongono di una chiave primaria.	Prima di creare una distribuzione blu/verde, assicurati che tutte le tabelle abbiano una chiave o un utilizzo primari. `REPLICA IDENTITY FULL` Tuttavia, utilizzatela solo `REPLICA IDENTITY FULL` se non esiste una chiave primaria o unica, poiché influisce sulle prestazioni di replica. Per ulteriori informazioni, consultare la documentazione di PostgreSQL.

Considerazioni sulle implementazioni blu/verde

Amazon RDS traccia le risorse nelle implementazioni blu/verde con `DbiResourceId` e `DbClusterResourceId` di ciascuna risorsa. Questo ID di risorsa è un Regione AWS identificatore univoco e immutabile della risorsa.

Ciascuno è elencato nella configurazione del database nella console RDS.

Il nome (ID cluster) di una risorsa viene modificato quando effettui lo switchover a un'implementazione blu/verde, ma ogni risorsa mantiene lo stesso ID risorsa. Ad esempio, un identificatore di cluster di database potrebbe essere `mycluster` nell'ambiente blu. Dopo lo switchover, lo stesso cluster di database potrebbe essere rinominato in `mycluster-old1`. Tuttavia, l'ID risorsa del cluster di database non viene modificato durante lo switchover. Quindi, quando sostituisci le risorse verdi con le nuove risorse di produzione, le loro risorse IDs non corrispondono alla risorsa blu IDs che era precedentemente in produzione.

Dopo il passaggio a una distribuzione blu/verde, valuta la possibilità di aggiornare la risorsa IDs con quelle delle risorse di produzione appena trasferite per funzionalità e servizi integrati utilizzati con le risorse di produzione. In particolare, considera i seguenti aggiornamenti:

- Se esegui il filtraggio utilizzando l'API e la risorsa RDS IDs, modifica la risorsa IDs utilizzata nel filtraggio dopo il passaggio.
- Se utilizzate risorse CloudTrail per il controllo, impostate i consumatori della risorsa in modo che tengano traccia della nuova risorsa dopo CloudTrail il passaggio. IDs Per ulteriori informazioni, consulta Monitoraggio delle chiamate API Aurora in AWS CloudTrail.

- Se utilizzi i flussi di attività del database per le risorse dell'ambiente blu, regola l'applicazione per monitorare gli eventi del database per il nuovo flusso dopo lo switchover. Per ulteriori informazioni, consulta Regioni supportate e motori Aurora DB per i flussi di attività del database.
- Se utilizzi l'API Performance Insights, modifica la risorsa IDs nelle chiamate all'API dopo lo switchover. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

  Dopo lo switchover è possibile monitorare un database con lo stesso nome, ma non contiene i dati precedenti allo lo switchover.
- Se utilizzi una risorsa IDs nelle policy IAM, assicurati di aggiungere la risorsa IDs delle risorse appena trasferite quando necessario. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.
- Se hai ruoli IAM associati all', assicurati di riassociarli dopo il passaggio. I ruoli allegati non vengono copiati automaticamente nell'ambiente verde.
- Se si esegue l'autenticazione nel cluster database utilizzando l'autenticazione del database IAM, assicurarsi che la policy IAM utilizzata per l'accesso al database contenga sia i database blu che quelli verdi elencati sotto l'elemento `Resource` della policy. Ciò è necessario per connettersi al database verde dopo il passaggio. Per ulteriori informazioni, consulta the section called “Creazione e utilizzo di una policy IAM per l'accesso al database IAM”.
- Se desideri ripristinare uno snapshot di cluster di database manuale per un cluster di database che faceva parte di un'implementazione blu/verde, assicurati di ripristinare lo snapshot del cluster di database corretto esaminando l'ora in cui è stato creato. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database.
- Dopo il passaggio, le attività di replica AWS Database Migration Service (AWS DMS) non possono riprendere perché il checkpoint dall'ambiente blu non è valido nell'ambiente verde. È necessario ricreare l'attività DMS con un nuovo checkpoint per continuare la replica.
- Amazon Aurora crea l'ambiente verde clonando il volume di archiviazione di Aurora sottostante nell'ambiente blu. Il volume del cluster verde memorizza solo le modifiche incrementali apportate all'ambiente verde. Se elimini il cluster di database nell'ambiente blu, la dimensione del volume di archiviazione di Aurora sottostante nell'ambiente verde aumenta fino a raggiungere la dimensione completa. Per ulteriori informazioni, consulta the section called “Clonazione di un volume per un cluster database Aurora”.
- Quando aggiungi un'istanza database al cluster di database nell'ambiente verde di un'implementazione blu/verde, la nuova istanza database non sostituisce un'istanza database nell'ambiente blu al momento dello switchover. Tuttavia, la nuova istanza database viene

mantenuta nel cluster di database e diventa un'istanza database nel nuovo ambiente di produzione.

- Quando si elimina un'istanza DB nel cluster DB nell'ambiente verde di una blue/green deployment, you can't create a new DB instance to replace it in the blue/green distribuzione.

  Se crei una nuova istanza database con lo stesso nome e nome della risorsa Amazon (ARN) dell'istanza database eliminata, ha un `DbiResourceId` diverso e quindi non fa parte dell'ambiente verde.

  Se elimini un'istanza database nel cluster di database nell'ambiente verde, si verifica il seguente comportamento:

  - Se l'istanza database con lo stesso nome esiste nell'ambiente blu, non verrà eseguito lo switchover all'istanza database dell'ambiente verde. Questa istanza database non verrà rinominata aggiungendo `-old`*n* al suo nome.
  - Qualsiasi applicazione che punti all'istanza database nell'ambiente blu continua a utilizzare la stessa istanza database dopo lo switchover.

Best practice per le distribuzioni Amazon Aurora

Di seguito sono riportate le best practice per le distribuzioni blu/verdi.

Argomenti

- Le migliori pratiche generali per le implementazioni blu/verdi
- per implementazioni blu/green
- Le migliori pratiche di Aurora PostgreSQL per implementazioni blu/green

Le migliori pratiche generali per le implementazioni blu/verdi

Prendi in considerazione le seguenti best practice generali quando crei una distribuzione blu/verde.

- Esegui accuratamente il test del cluster di database Aurora nell'ambiente verde prima di effettuare lo switchover.
- Mantieni i tuoi database nell'ambiente verde di sola lettura. Si consiglia di abilitare le operazioni di scrittura nell'ambiente verde con cautela perché possono causare conflitti di replica nell'ambiente verde. Possono inoltre generare dati non previsti nei database di produzione dopo lo switchover.

- Se utilizzi una distribuzione blu/verde per implementare le modifiche allo schema, apporta solo modifiche compatibili con la replica.

  Ad esempio, è possibile aggiungere nuove colonne alla fine di una tabella senza interrompere la replica dalla distribuzione blu a quella verde. Tuttavia, le modifiche dello schema, come la ridenominazione delle colonne o delle tabelle, interrompono la replica nell'implementazione verde.

  Per ulteriori informazioni sulle modifiche compatibili con la replica, consulta Replication with Differing Table Definitions on Source and Replica nella documentazione MySQL e Restrictions nella documentazione della replica logica PostgreSQL.
- Usa l'endpoint del cluster, l'endpoint di lettura o l'endpoint personalizzato per tutte le connessioni in entrambi gli ambienti. Non utilizzare endpoint di istanza o endpoint personalizzati con gli elenchi statici o di esclusione.
- Quando si effettua lo switchover in una implementazione blu/verde, attenersi alle best practice relative allo switchover. Per ulteriori informazioni, consulta the section called “Best practice per lo switchover”.

per implementazioni blu/green

Prendi in considerazione le seguenti best practice quando crei una distribuzione blu/verde da un cluster Aurora .

- Se l'ambiente verde presenta un ritardo di replica, considera quanto segue:
  - Disabilita la conservazione dei log binari se non è necessaria o disabilitala temporaneamente fino a quando la replica non viene ripristinata. A tale scopo, reimposta il parametro del cluster `binlog_format` DB su `0` e riavvia l'istanza DB green writer.
  - Imposta temporaneamente il `innodb_flush_log_at_trx_commit` parametro nel gruppo di parametri DB verde. Una volta ripristinata la replica, ripristinate il valore predefinito di prima dello switchover. 1 Se si verifica un arresto imprevisto o un arresto anomalo con il valore temporaneo del parametro, ricostruisci l'ambiente verde per evitare il danneggiamento dei dati non rilevato. Per ulteriori informazioni, consulta the section called “Configurazione della frequenza di svuotamento del buffer dei registri”.

Le migliori pratiche di Aurora PostgreSQL per implementazioni blu/green

Prendi in considerazione le seguenti best practice quando crei una distribuzione blu/verde da un cluster Aurora PostgreSQL DB.

- Monitora la cache write-through della replica logica di Aurora PostgreSQL e apporta modifiche al buffer della cache, se necessario. Per ulteriori informazioni, consulta the section called “Monitoraggio della cache write-through della replica logica”.
- Aumenta il valore del parametro `logical_decoding_work_mem` DB nell'ambiente blu. In questo modo si riduce la decodifica su disco e si utilizza invece la memoria. Per ulteriori informazioni, consulta the section called “Regolazione della memoria di lavoro per la decodifica logica”.
  - È possibile monitorare l'overflow delle transazioni in fase di scrittura su disco utilizzando la `ReplicationSlotDiskUsage` CloudWatch metrica. Questa metrica offre informazioni sull'utilizzo del disco degli slot di replica, aiutando a identificare quando i dati delle transazioni superano la capacità di memoria e vengono archiviati su disco. È possibile monitorare la memoria liberabile con la metrica. `FreeableMemory` CloudWatch Per ulteriori informazioni, consulta the section called “Parametri a livello di istanza per Amazon Aurora”.
  - In Aurora PostgreSQL versione 14 e successive, è possibile monitorare la dimensione dei file di overflow logico utilizzando la vista di sistema. `pg_stat_replication_slots`
- Aggiorna tutte le estensioni di PostgreSQL all'ultima versione prima di creare un'implementazione blu/verde. Per ulteriori informazioni, consulta the section called “Aggiornamento estensioni PostgreSQL”.
- Se utilizzi l'`aws_s3`estensione, consenti al cluster DB verde l'accesso ad Amazon S3 tramite un ruolo IAM dopo la creazione dell'ambiente verde. In tal modo i comandi di importazione ed esportazione continuano a funzionare dopo lo switchover. Per istruzioni, consultare the section called “Configurazione dell'accesso a un bucket Amazon S3”.
- Se specifichi una versione superiore del motore per l'ambiente verde, esegui l'`ANALYZE`operazione su tutti i database per aggiornare la `pg_statistic` tabella. Le statistiche di Optimizer non vengono trasferite durante l'aggiornamento di una versione principale, quindi è necessario rigenerare tutte le statistiche per evitare problemi di prestazioni. Per ulteriori procedure consigliate durante gli aggiornamenti delle versioni principali, consulta. the section called “Esecuzione di un aggiornamento della versione principale”
- Evita di configurare i trigger `ENABLE ALWAYS` se `ENABLE REPLICA` o se il trigger viene utilizzato sulla fonte per manipolare i dati. In caso contrario, il sistema di replica propaga le modifiche ed esegue il trigger, che porta alla duplicazione.
- Le transazioni di lunga durata possono causare un notevole ritardo nella replica. Per ridurre il ritardo di replica, prova a fare quanto segue:
  - Riduci le transazioni di lunga durata e le sottotransazioni che possono essere ritardate fino a quando l'ambiente verde non raggiunge il livello dell'ambiente blu.

- Riduci le operazioni di massa sull'ambiente blu fino a quando l'ambiente verde non raggiungerà il livello dell'ambiente blu.
- Avvia un'operazione manuale di congelamento sottovuoto su tavoli occupati prima di creare la distribuzione blu/verde. Per istruzioni, vedere Esecuzione di un congelamento sottovuoto manuale.
- Nella versione 12 di PostgreSQL e successive, disabilita `index_cleanup` il parametro su tabelle di grandi dimensioni o occupate per migliorare l'efficienza della manutenzione regolare dei database blu. Per ulteriori informazioni, consulta Aspirare una tabella il più rapidamente possibile.

  > **Note**
  >
  > Saltare regolarmente la pulizia dell'indice durante l'aspirazione può causare un aumento dell'indice, che potrebbe compromettere le prestazioni di scansione. Come procedura ottimale, utilizzate questo approccio solo quando utilizzate una distribuzione blu/verde. Una volta completata la distribuzione, consigliamo di riprendere la manutenzione e la pulizia regolari dell'indice.

- Il ritardo nella replica può verificarsi se le istanze DB blu e verdi sono sottodimensionate per il carico di lavoro. Assicurati che le tue istanze DB non raggiungano i limiti di risorse per il tipo di istanza. Per ulteriori informazioni, consulta the section called “Analizza l'utilizzo delle risorse con i parametri CloudWatch ”.

- Una replica lenta può causare il riavvio frequente di mittenti e destinatari, il che ritarda la sincronizzazione. Per assicurarvi che rimangano attivi, disattivate i timeout impostando il `wal_sender_timeout` parametro su `0` nell'ambiente blu e il parametro su nell'ambiente verde. `wal_receiver_timeout` `0`
- Esamina le prestazioni delle istruzioni UPDATE e DELETE e valuta se la creazione di un indice sulla colonna utilizzata nella clausola WHERE può ottimizzare queste query. Ciò può migliorare le prestazioni quando le operazioni vengono ripetute nell'ambiente verde. Per ulteriori informazioni, consulta the section called “Controlla i filtri predicati per le query che generano attese”.
- Se utilizzi i trigger, assicurati che non interferiscano con la creazione, l'aggiornamento e l'eliminazione di `pg_catalog.pg_publication` `pg_catalog.pg_replication_slots` oggetti i cui nomi iniziano con «rds». `pg_catalog.pg_subscription`

- Se Babelfish è abilitato sul cluster DB di origine, i seguenti parametri devono avere le stesse impostazioni nel gruppo di parametri del cluster DB di destinazione per l'ambiente verde come nel gruppo di parametri del cluster DB di origine:
  - rds.babelfish_status
  - babelfishpg_tds.tds_default_numeric_precision
  - babelfishpg_tds.tds_default_numeric_scale
  - babelfishpg_tsql.default_locale
  - babelfishpg_tsql.migration_mode
  - babelfishpg_tsql.server_collation_name

  Per ulteriori informazioni su questi parametri, consultare the section called “Impostazioni del gruppo di parametri del cluster database per Babelfish ”.

Creazione di una distribuzione blu/verde in Amazon

RDS copia la topologia e le caratteristiche dell'ambiente blu in un'area di staging. Se l'istanza DB blu ha delle repliche lette, queste vengono copiate come repliche dell'istanza verde. Lo storage allocato di tutte le repliche verdi corrisponde all'istanza principale verde, mentre gli altri parametri di archiviazione vengono ereditati dalle repliche blu.

Quando si crea un'implementazione blu/verde, si specifica il cluster di database da copiare nell'implementazione. Il cluster scelto è il cluster di database di produzione e diventa il cluster di database dell'ambiente blu. RDS copia la topologia dell'ambiente blu in un'area di gestione temporanea, insieme alle funzionalità configurate. Il cluster di database viene copiato nell'ambiente verde e RDS configura la replica dal cluster di database dell'ambiente blu nel cluster di database dell'ambiente verde. RDS copia anche tutte le istanze database del cluster di database.

Argomenti

- Preparazione di una implementazione blu/verde
- Specifica delle modifiche durante la creazione di un'implementazione blu/verde
- Creazione di un'implementazione blu/verde
- Impostazioni per la creazione di distribuzioni blu/verdi

Preparazione di una implementazione blu/verde

Esistono alcuni passaggi da eseguire prima di creare una distribuzione blu/verde, a seconda del motore su cui è in esecuzione l'istanza database del cluster Aurora .

Argomenti

- Preparazione di un cluster Aurora MySQL DB per una distribuzione blu/verde
- Preparazione di un cluster Aurora PostgreSQL DB per una distribuzione blu/verde

Preparazione di un cluster Aurora MySQL DB per una distribuzione blu/verde

Prima di creare un'implementazione blu/verde per un cluster di database Aurora MySQL, il cluster deve essere associato a un gruppo di parametri del·cluster di database personalizzato con la registrazione binaria (`binlog_format`) attivata. Il log binario è necessario per la replica dall'ambiente blu nell'ambiente verde. Sebbene qualsiasi formato binlog funzioni, consigliamo `ROW` per ridurre il rischio di incongruenze di replica. Per informazioni sulla creazione di un gruppo di parametri del·cluster di database personalizzato e sull'impostazione dei parametri, consulta the section called “Gruppi di parametri di cluster database”.

> **Note**
>
> L'abilitazione della registrazione dei log binari aumenta il numero delle operazioni I/O di scrittura sul disco nel cluster DB. È possibile monitorare l'utilizzo degli IOPS con la metrica. `VolumeWriteIOPs` CloudWatch

Dopo aver abilitato la registrazione binaria, assicuratevi di riavviare il cluster DB in modo che le modifiche abbiano effetto. Affinché la creazione di implementazioni blu/verde abbia esito positivo, l'istanza di scrittura deve essere sincronizzata con il gruppo di parametri del cluster database. Per ulteriori informazioni, consulta Riavvio di un'istanza database in un cluster Aurora.

Inoltre, si consiglia di modificare il periodo di conservazione dei log binari impostando un valore diverso da quello `NULL` per impedire l'eliminazione dei file di log binari. Per ulteriori informazioni, consulta the section called “Impostazione e visualizzazione della configurazione del registro binario”.

Preparazione di un cluster Aurora PostgreSQL DB per una distribuzione blu/verde

Prima di creare una distribuzione blu/verde per un cluster Aurora PostgreSQL DB, assicurati di fare quanto segue.

- Associa il cluster a un gruppo di parametri del·cluster di database personalizzato con la replica logica (`rds.logical_replication`) abilitata. La replica logica è necessaria per la replica dall'ambiente blu nell'ambiente verde.

  Quando abiliti la replica logica, devi anche ottimizzare alcuni parametri del cluster, ad esempio, e. `max_replication_slots` `max_logical_replication_workers` `max_worker_processes` Per istruzioni su come abilitare la replica logica e ottimizzare questi parametri, vedere. the section called “Configurazione della replica logica”

  Inoltre, assicuratevi che il `synchronous_commit` parametro sia impostato su. on

  Dopo aver configurato i parametri richiesti, riavviate il cluster DB in modo che le modifiche abbiano effetto. Affinché la creazione di implementazioni blu/verde abbia esito positivo, l'istanza di scrittura deve essere sincronizzata con il gruppo di parametri del cluster database. Per ulteriori informazioni, consulta Riavvio di un'istanza database in un cluster Aurora.
- Verifica che il tuo cluster DB stia eseguendo una versione di Aurora PostgreSQL compatibile con Blue/Green Deployments. Per l'elenco delle versioni compatibili, consulta the section called “Implementazioni blu/verdi con Aurora Postgre SQL”.
- Assicurati che tutte le tabelle del cluster di database abbiano una chiave primaria. La replica logica di PostgreSQL non consente operazioni UPDATE o DELETE su tabelle che non dispongono di una chiave primaria.

Specifica delle modifiche durante la creazione di un'implementazione blu/verde

È possibile apportare le seguenti modifiche al cluster di database nell'ambiente verde quando si crea l'implementazione blu/verde:

È possibile apportare altre modifiche al cluster di database e alle relative istanze database nell'ambiente verde dopo l'implementazione. Ad esempio, è possibile specificare una versione del motore superiore o un gruppo di parametri diverso.

Per ulteriori informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

Argomenti

- Specifica di una versione successiva del motore
- Specifica di un gruppo di parametri di database

Specifica di una versione successiva del motore

È possibile specificare una versione superiore del motore se si desidera testare un aggiornamento del motore di database. Al momento dello switchover, il database viene aggiornato alla versione principale o secondaria specificata del motore di database.

Specifica di un gruppo di parametri di database

Specifica un gruppo di parametri del·cluster di database diverso da quello utilizzato dal cluster di database. È possibile verificare in che modo le modifiche ai parametri influiscono sul cluster di database nell'ambiente verde o specificare un gruppo di parametri per una nuova versione principale del motore di database in caso di aggiornamento.

Se si specifica un gruppo di parametri del·cluster di database diverso, il gruppo specificato viene associato al cluster di database nell'ambiente verde. Se non si specifica un gruppo di parametri del·cluster di database diverso, il cluster dell'ambiente verde è associato allo stesso gruppo di parametri del·cluster di database dell'ambiente blu.

Creazione di un'implementazione blu/verde

È possibile creare una distribuzione blu/verde utilizzando l'API AWS Management Console AWS CLI, the o RDS.

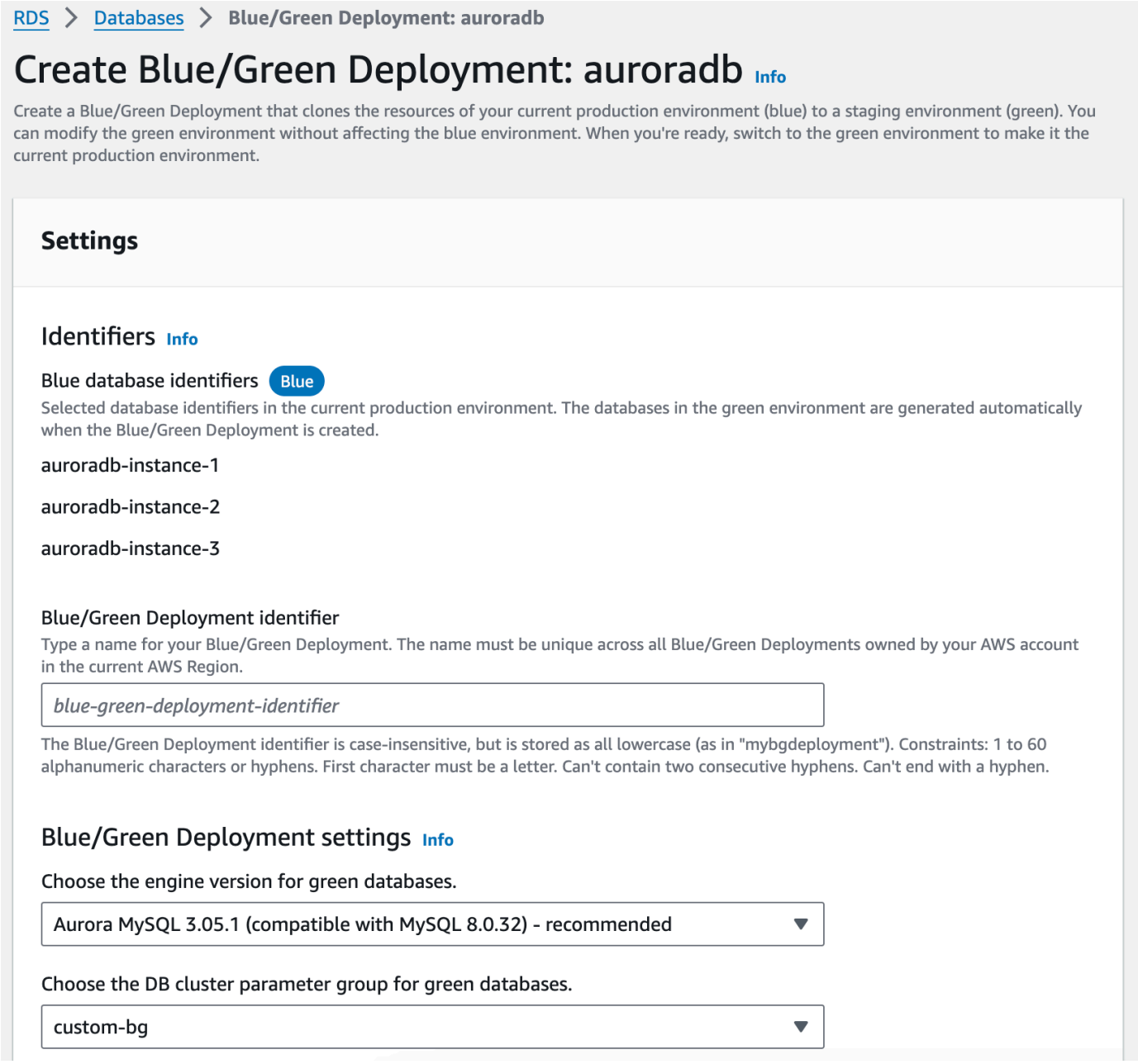
Console

Per creare un'implementazione blu/verde

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database) quindi seleziona il cluster di database da copiare nell'ambiente verde.

3. Scegli Azioni, Crea distribuzione blu/verde.

   Viene visualizzata la pagina Crea distribuzione blu/verde.



4. Controlla gli identificatori blu del database. Assicurati che corrispondano alle istanze DB che ti aspetti nell'ambiente blu. In caso contrario, scegli Cancel (Annulla).
5. Per il nome della distribuzione blu/verde, inserisci un nome per la tua distribuzione blu/verde.
6. Nelle sezioni rimanenti, specifica le impostazioni per l'ambiente verde. Per informazioni su ciascuna impostazione, consulta the section called “Impostazioni disponibili”.

È possibile apportare altre modifiche ai database nell'ambiente verde dopo che è stato implementato.

7. Scegli Create (Crea).

AWS CLI

Per creare una distribuzione blu/verde utilizzando il AWS CLI, usa il create-blue-green-deploymentcomando. Per informazioni su tutte le opzioni disponibili, vedere. the section called "Impostazioni disponibili"

Example

In Linux, macOS, oppure Unix:

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name aurora-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb \
    --target-engine-version 8.0 \
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

In Windows:

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name aurora-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb ^
    --target-engine-version 8.0 ^
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

API RDS

Per creare una distribuzione blu/verde utilizzando l'API Amazon RDS, utilizza l'operazione. `CreateBlueGreenDeployment` Per ulteriori informazioni su ciascuna opzione, consulta the section called "Impostazioni disponibili".

Impostazioni per la creazione di distribuzioni blu/verdi

La tabella seguente spiega le impostazioni che è possibile scegliere quando si crea una distribuzione blu/verde. Per ulteriori informazioni sulle AWS CLI opzioni, vedere. create-blue-green-deployment Per ulteriori informazioni sui parametri dell'API RDS, vedere CreateBlueGreenDeployment.

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS

Identificatore di distribuzione blu/verde	Un nome per la distribuzione blu/verde.	Opzione CLI:
		`--blue-green-deployment-name`
		Parametro API:
		`BlueGreenDeploymentName`

Identificatore blu del database	L'identificatore del cluster di che si desidera copiare nell'ambiente verde. Quando utilizzi la CLI o l'API, specifica il cluster di Amazon Resource Name (ARN).	Opzione CLI:
		`--source`
		Parametro API:
		`Source`

Gruppo di parametri del cluster DB per database verdi	Un gruppo di parametri da associare ai database nell'ambiente verde.	Opzione CLI:
		`--target-db-cluster-parameter-group-name`
		Parametro API:
		`TargetDBClusterParameterGroupName`

Versione del motore per database ecologici	Aggiorna il cluster di nell'ambiente verde alla versione del motore DB specificata.
	Se scegli un'istanza database Aurora PostgreSQL DB RDS per PostgreSQL che utilizza la replica . Per ulteriori informazioni, consulta the section called “Limitazioni
		Opzione CLI:
		`--target-engine-version`
		Parametro API RDS:
		`TargetEngineVersion`

Impostazione della console	Descrizione impostazione	Opzione CLI e parametro API di RDS
	specifiche della replica logica per le distribuzioni blu/verdi”.	

Visualizzazione di una distribuzione blu/verde in Amazon

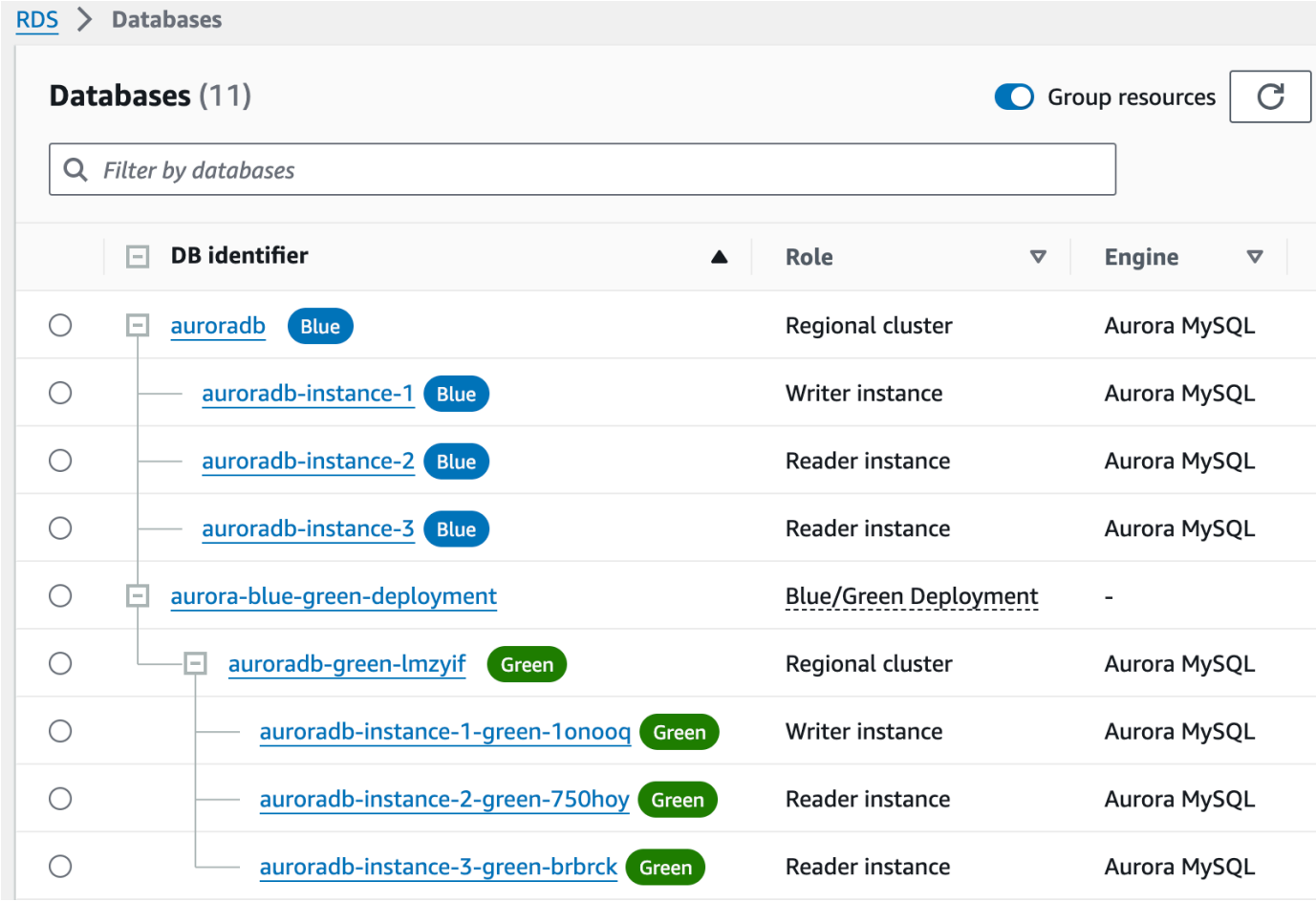
È possibile visualizzare i dettagli su una distribuzione blu/verde utilizzando l'API AWS Management Console AWS CLI, the o RDS.

È anche possibile visualizzare e sottoscrivere gli eventi per informazioni su un'implementazione blu/verde. Per ulteriori informazioni, consulta Eventi di implementazione blu/verde.

Console

Per visualizzare i dettagli di un'implementazione blu/verde

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database), quindi trova l'implementazione blu/verde nell'elenco.

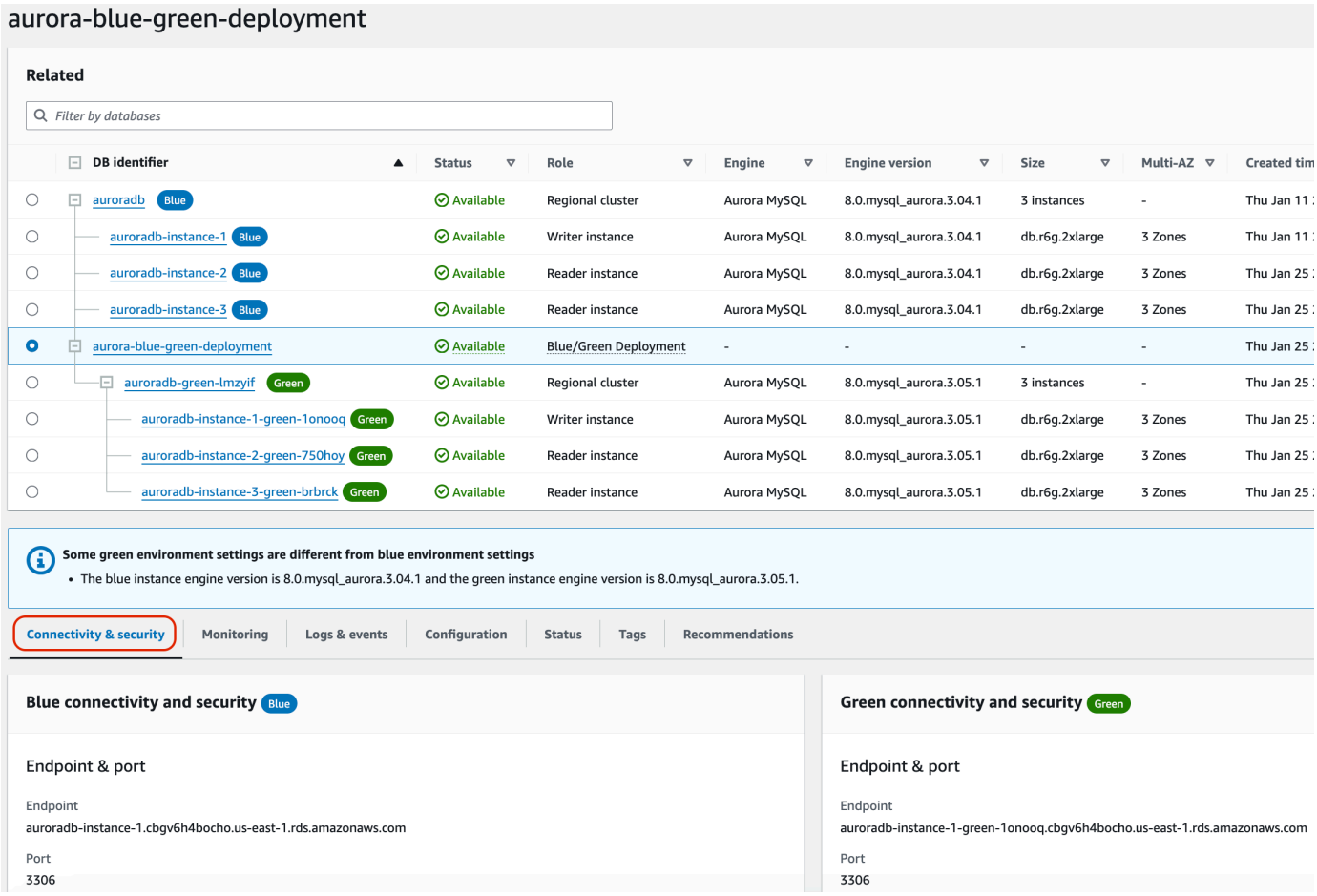


Il valore Role (Ruolo) per l'implementazione blu/verde è Blue/Green Deployment (Implementazione blu/verde).

3. Scegli il nome dell'implementazione blu/verde di cui desideri visualizzarne i dettagli.

   Ogni scheda ha una sezione per l'implementazione blu e una sezione per l'implementazione verde. Ad esempio, nella scheda Configurazione, la versione del motore DB potrebbe essere diversa nell'ambiente blu e nell'ambiente verde se stai aggiornando la versione del motore DB nell'ambiente verde.

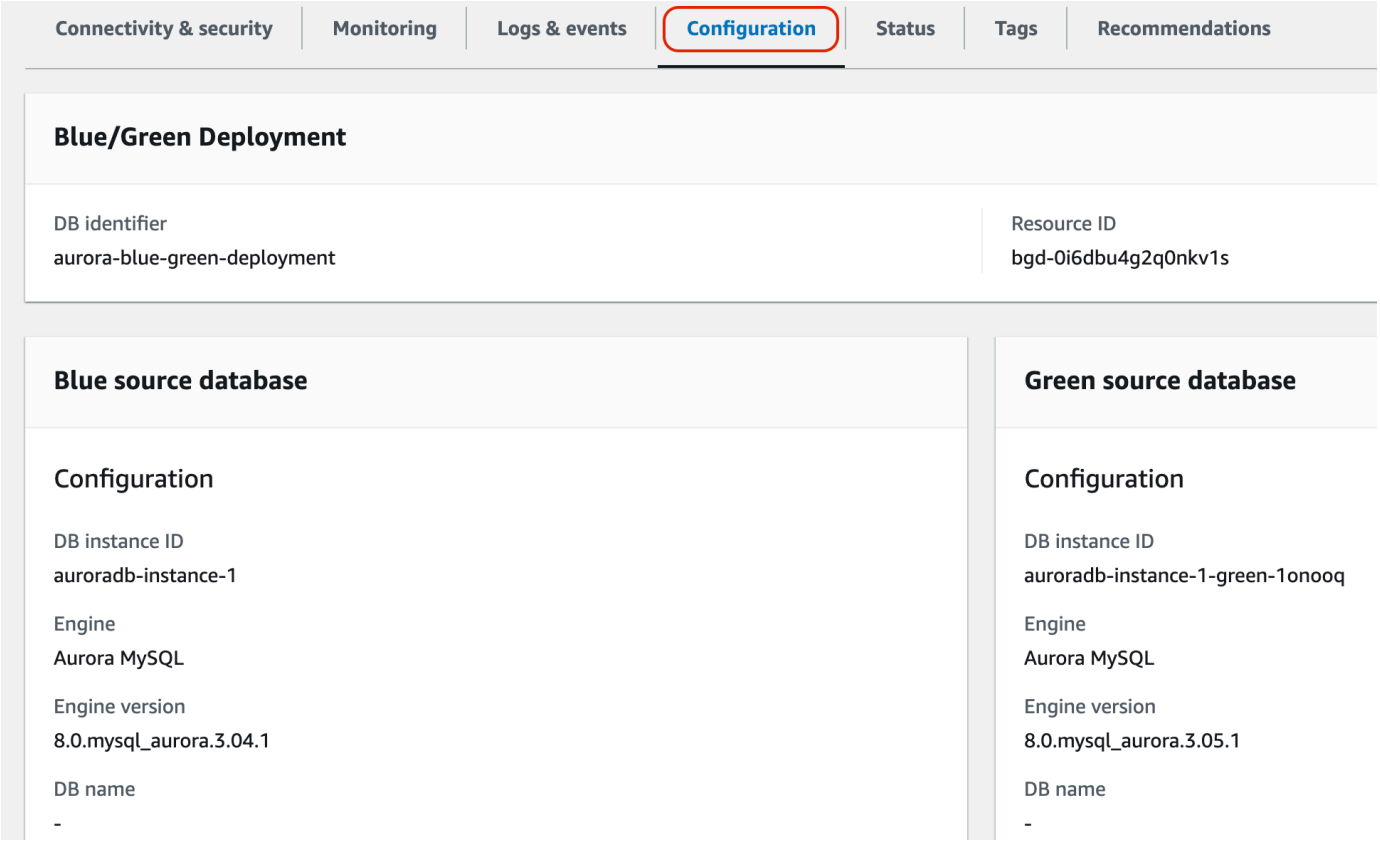
   L'immagine seguente mostra un esempio della scheda Connettività e sicurezza:



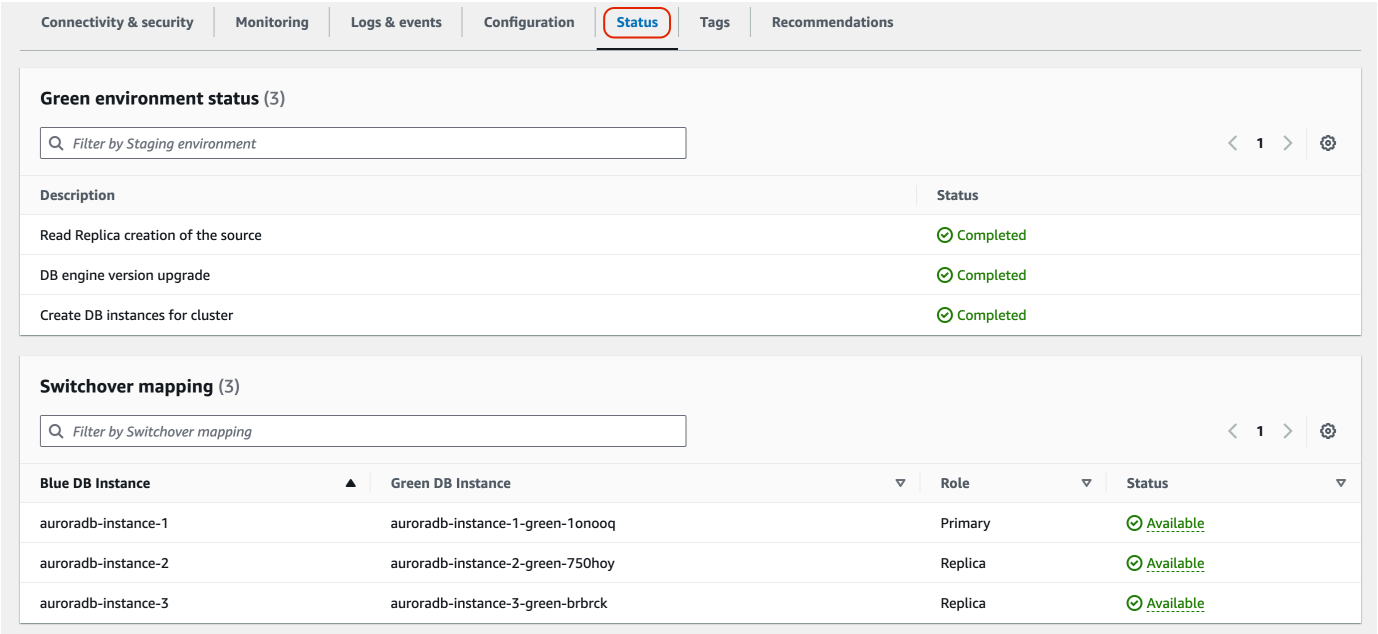
La scheda Connettività e sicurezza include anche una sezione chiamata Replica, che mostra lo stato attuale della replica e il ritardo di replica tra gli ambienti blu e verde. Se lo stato di replica è `Replicating`, l'implementazione blu/verde viene replicata correttamente.

Per le implementazioni blu/verdi di Aurora PostgreSQL RDS per PostgreSQL non supportate o a oggetti di grandi dimensioni nell'ambiente blu. `Replication degraded` Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blu/verdi”.

L'immagine seguente mostra un esempio della scheda Configurazione:



L'immagine seguente mostra un esempio della scheda Stato:



AWS CLI

Per visualizzare i dettagli su una distribuzione blu/verde utilizzando il AWS CLI, utilizzare il describe-blue-green-deploymentscomando.

Example Visualizzazione dei dettagli di un'implementazione blu/verde filtrando per nome

Quando si utilizza il describe-blue-green-deploymentscomando, è possibile filtrare in base a. `--blue-green-deployment-name`

L'esempio seguente mostra i dettagli per un'implementazione blu/verde denominata *`my-blue-green-deployment`*.

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example Visualizzazione dei dettagli di un'implementazione blu/verde specificando l'identificatore

Quando si utilizza il describe-blue-green-deploymentscomando, è possibile specificare l'`--blue-green-deployment-identifier`opzione.

L'esempio seguente mostra i dettagli per un'implementazione blu/verde con l'identificatore *`bgd-1234567890abcdef`*.

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

API RDS

Per visualizzare i dettagli su un'implementazione blu/verde utilizzando l'API Amazon RDS, utilizza l'operazione `DescribeBlueGreenDeployments` e specifica `BlueGreenDeploymentIdentifier`.

Cambiare una distribuzione blu/verde in Amazon

Uno switchover trasforma il cluster DB, comprese le relative istanze DB, nell'ambiente verde per diventare il cluster DB di produzione. Prima dello switchover, il traffico di produzione viene indirizzato al cluster nell'ambiente blu. Dopo lo switchover, il traffico di produzione viene indirizzato al cluster di database nell'ambiente verde.

Il passaggio a una distribuzione blu/verde non equivale a promuovere il cluster DB di verdi all'interno della distribuzione blu/verde. Se promuovi manualmente il cluster DB di verdi selezionando Promuovi dal menu Azioni, la replica tra gli ambienti blu e verde si interrompe e la distribuzione blu/verde entra in uno stato di configurazione non valida.

Argomenti

- Timeout dello switchover
- Guardrail dello switchover
- Azioni dello switchover
- Best practice per lo switchover
- Verifica CloudWatch delle metriche prima del passaggio al digitale
- Monitoraggio del ritardo nella replica prima del passaggio al digitale
- Switchover di un'implementazione blu/verde
- Dopo lo switchover

Timeout dello switchover

È possibile specificare un timeout per lo switchover compreso tra 30 secondi e 3.600 secondi (un'ora). Se lo switchover richiede più tempo della durata specificata, viene eseguito il rollback di tutte le modifiche e non viene apportata alcuna modifica agli ambienti. L'impostazione predefinita del timeout è 300 secondi (cinque minuti).

Guardrail dello switchover

Quando avvii uno switchover, Amazon RDS esegue alcuni controlli di base per verificare la preparazione degli ambienti blu e verdi per lo switchover. Questi controlli sono noti come guardrail dello switchover e impediscono lo switchover se gli ambienti non sono pronti per farlo. Pertanto, evitano tempi di inattività più lunghi del previsto e impediscono la perdita di dati tra gli ambienti blu e quelli verdi che potrebbe verificarsi se lo switchover venisse avviato.

Amazon RDS esegue i seguenti controlli guardrail sull'ambiente verde:

- Integrità della replica: verifica se lo stato della replica del cluster di database verde è integro. Il cluster di database verde è una replica del cluster di database blu.
- Ritardo della replica: verifica se il ritardo della replica del cluster di database verde rientra nei limiti consentiti per lo switchover. I limiti consentiti si basano sul periodo di timeout specificato. Il ritardo

della replica indica il ritardo del cluster di database verde rispetto al cluster di database blu. Per ulteriori informazioni, consulta the section called “Monitoraggio del ritardo nella replica prima del passaggio al digitale”.

- Scritture attive: assicura che non vi siano scritture attive nel cluster di database verde.

Amazon RDS esegue i seguenti controlli guardrail sull'ambiente blu:

- Replica esterna: per Aurora , assicura che l'ambiente blu non sia una fonte logica autogestita (editore) o una replica (sottoscrittore). In tal caso, si consiglia di eliminare gli slot e gli abbonamenti di replica autogestiti su tutti i database nell'ambiente blu, procedere con il passaggio al digitale e quindi ricrearli per riprendere la replica. Per Aurora MySQL RDS per MySQL , verifica se il database blu non è una replica binlog esterna. In caso affermativo, assicurati che non si stia replicando attivamente.
- Scritture attive di lunga durata: assicura che non vi siano scritture attive di lunga durata nel cluster di database blu perché possono aumentare il ritardo della replica.
- Istruzioni DDL di lunga durata: assicura che non vi siano istruzioni DLL di lunga durata nel cluster di database blu perché possono aumentare il ritardo della replica.
- Modifiche a PostgreSQL non supportate: per le . Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blu/verdi”.

  Se Amazon RDS rileva modifiche PostgreSQL non supportate, modifica lo stato di replica e notifica all'utente che lo switchover non è disponibile `Replication degraded` per la distribuzione e per tutti i database verdi. blue/green deployment. To proceed with switchover, we recommend that you delete and recreate the blue/green A tale scopo, scegli Operazioni, Elimina con database verdi.

Azioni dello switchover

Quando si effettua lo switchover per un'implementazione blu/verde, RDS esegue le seguenti azioni:

1. Esegue controlli guardrail per verificare se gli ambienti blu e verdi sono pronti per lo switchover.
2. Interrompe le nuove operazioni di scrittura nel cluster di database in entrambi gli ambienti.
3. Elimina le connessioni alle istanze database in entrambi gli ambienti e non consente nuove connessioni.
4. Attende che la replica recuperi l'ambiente verde in modo che sia sincronizzato con l'ambiente blu.
5. Rinomina il cluster di database e le istanze database in entrambi gli ambienti.

RDS rinomina il cluster di database e le istanze database nell'ambiente verde in modo che corrispondano al cluster di database e alle istanze database nell'ambiente blu. Ad esempio, supponi che il nome dell'istanza database nell'ambiente blu sia `mydb`. Supponi anche che il nome dell'istanza database corrispondente nell'ambiente verde sia `mydb-green-abc123`. Durante lo switchover, il nome dell'istanza database nell'ambiente verde viene modificato in `mydb`.

RDS rinomina il cluster di database e le istanze database nell'ambiente blu aggiungendo `-old`*n* al nome corrente, dove *n* è un numero. Ad esempio, supponi che il nome dell'istanza database nell'ambiente blu sia `mydb`. Dopo lo switchover, il nome dell'istanza database diventa `mydb-old1`.

RDS rinomina anche gli endpoint nell'ambiente verde in modo che corrispondano agli endpoint nell'ambiente blu, per non apportare modifiche all'applicazione.

6. Consente le connessioni ai database in entrambi gli ambienti.
7. Consente le operazioni di scrittura nel cluster di database nel nuovo ambiente di produzione.

   Dopo lo switchover, il precedente cluster DB dell'istanza DB di produzione consente operazioni di lettura solo e si riavvia l'istanza DB. `0` Anche se abiliti le scritture sul cluster DB, questo rimane di sola lettura finché non elimini la distribuzione blu/verde.

Puoi monitorare lo stato di uno switchover utilizzando Amazon. EventBridge Per ulteriori informazioni, consulta the section called “Eventi di implementazione blu/verde”.

Se hai dei tag configurati nell'ambiente blu, questi tag vengono copiati nel nuovo ambiente di produzione durante lo switchover. Per ulteriori informazioni sui tag, consulta Etichettatura di Amazon Aurora e risorse Amazon RDS.

Se lo switchover inizia e poi si interrompe prima del termine per un qualsiasi motivo, viene eseguito il rollback di tutte le modifiche e non viene apportata alcuna modifica agli ambienti.

Best practice per lo switchover

Prima di effettuare lo switchover, ti consigliamo vivamente di seguire le best practice completando le seguenti attività:

- Esegui accuratamente il test delle risorse nell'ambiente verde. Assicurati che funzionino correttamente ed efficacemente.
- Monitora le CloudWatch metriche Amazon pertinenti. Per ulteriori informazioni, consulta the section called “Verifica CloudWatch delle metriche prima del passaggio al digitale”.

- Identifica il momento migliore per lo switchover.

  Durante lo switchover, le scritture dei database vengono interrotte in entrambi gli ambienti. Identifica il momento in cui il traffico è più basso nell'ambiente di produzione. Le transazioni a lunga durata, come quelle attive DDLs, possono aumentare i tempi di passaggio al digitale, con conseguenti tempi di inattività più lunghi per i carichi di lavoro di produzione.

  Se è presente un numero elevato di connessioni sulle , sul cluster di database e sulle istanze DB, valuta la possibilità di ridurle manualmente al minimo necessario per l'applicazione prima di passare alla blue/green deployment. One way to achieve this is to create a script that monitors the status of the blue/green distribuzione e iniziare a ripulire le connessioni quando rileva che lo stato è cambiato in. `SWITCHOVER_IN_PROGRESS`
- Assicurati che il cluster di database e le istanze database siano nello stato `Available` in entrambi gli ambienti.
- Assicurati che il cluster di database nell'ambiente verde sia nello stato integro e in grado di replicare.
-

  Altrimenti, le applicazioni continueranno a inviare traffico di scrittura all'ambiente blu dopo lo switchover.
- Per le implementazioni Blue/Green di Aurora PostgreSQL RDS per PostgreSQL blu/green che utilizzano la replica logica come segue:
  - Esamina i limiti della replica logica e intraprendi le azioni necessarie prima del passaggio. Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blu/verdi”.
  - Eseguire l'operazione `ANALYZE` per aggiornare la tabella `pg_statistics`. Ciò riduce il rischio di problemi di prestazioni dopo il passaggio al digitale.

> **Note**
>
> Durante uno switchover non è possibile modificare il cluster di database incluso nello switchover.

Verifica CloudWatch delle metriche prima del passaggio al digitale

Prima di passare a una distribuzione blu/verde, ti consigliamo di verificare i valori delle seguenti metriche all'interno di Amazon. CloudWatch

- `DatabaseConnections`: utilizza questo parametro per stimare il livello di attività dell'implementazione blu/verde e assicurarti che il valore sia a un livello accettabile per la tua implementazione prima dello switchover. Se Approfondimenti sulle prestazioni è attivato, `DBLoad` è un parametro più accurato.
- `ActiveTransactions`: se `innodb_monitor_enable` è impostato su `all` nel gruppo di parametri del database per una qualsiasi istanza database, usa questo parametro per vedere se c'è un numero elevato di transazioni attive che possono bloccare lo switchover.

Per ulteriori informazioni, consulta the section called “CloudWatch metriche per Aurora”.

Monitoraggio del ritardo nella replica prima del passaggio al digitale

Prima di passare a un'implementazione blu/verde, assicurati che il ritardo della replica sia vicino allo zero per ridurre i tempi di inattività.

Aurora MySQL RDS per MySQL e per MariaDB

Per le implementazioni blu/green di MariadB, `AuroraBinlogReplicaLag` CloudWatch controlla la metrica nell'ambiente verde per identificare l'attuale ritardo della replica. Per ulteriori informazioni, consulta the section called “Diagnosi e risoluzione del ritardo tra repliche di lettura”.

Per le l'attuale ritardo `OldestReplicationSlotLag` CloudWatch della replica. Per ulteriori informazioni, consulta the section called “Parametri a livello di istanza per Amazon Aurora”.

Inoltre, puoi eseguire la seguente query SQL nell'ambiente blu:

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
WHERE slot_type = 'logical';
```

```
slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

`confirmed_flush_lsn`Rappresenta l'ultimo numero di sequenza di registro (LSN) inviato alla replica. `pg_current_wal_lsn`Rappresenta la posizione attuale del database. Un valore `lsn_distance` pari a 0 indica che la replica è stata recuperata.

Switchover di un'implementazione blu/verde

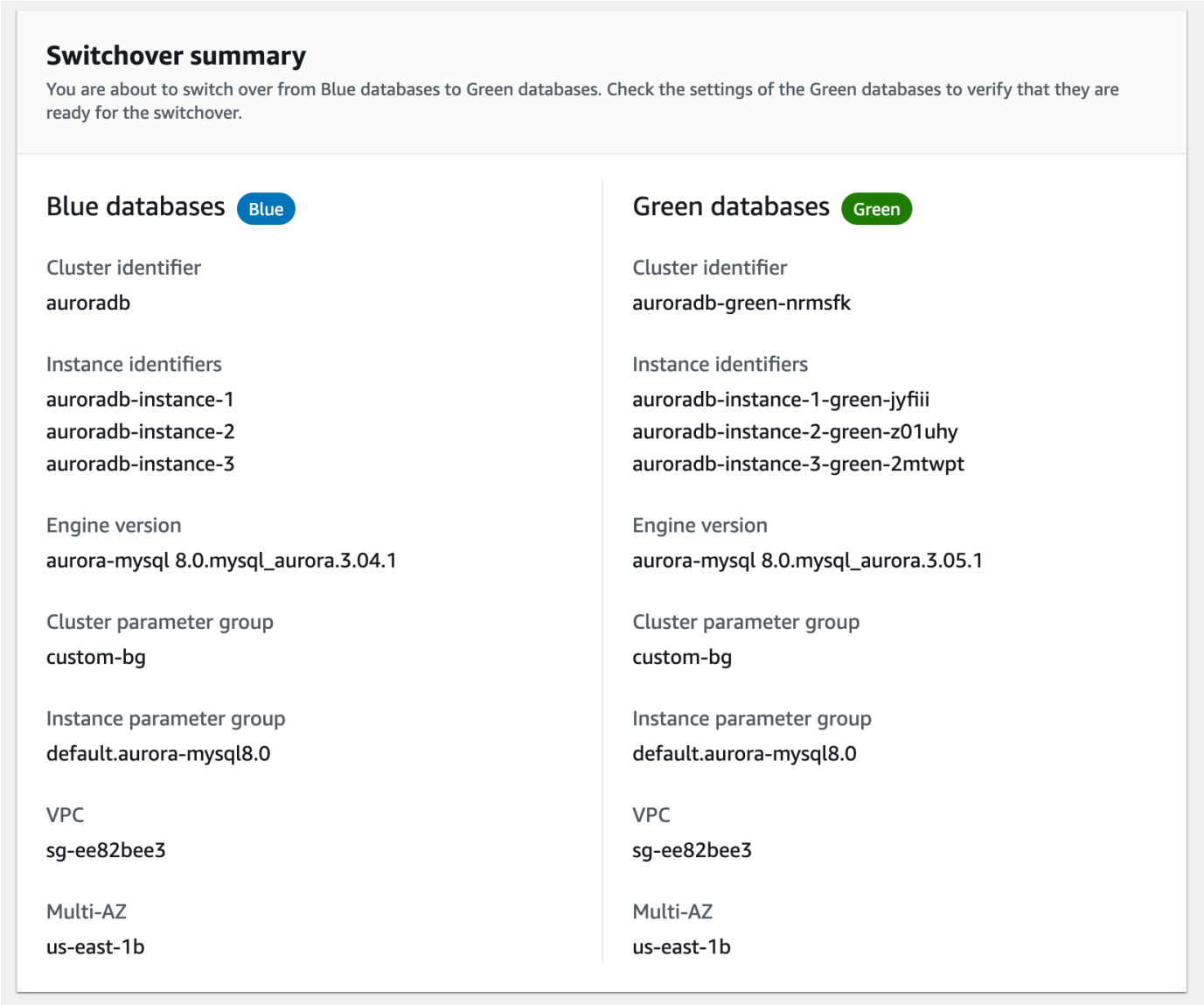
È possibile passare da una distribuzione blu/verde utilizzando l'API, the o RDS. AWS Management Console AWS CLI

Console

Per eseguire lo switchover di un'implementazione blu/verde

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database) e seleziona l'implementazione blu/verde di cui eseguire lo switchover.
3. In Actions (Operazioni), scegli Switch over (Esegui switchover).

   Viene visualizzata la pagina Switch over (Switchover).



4. Nella pagina Switch over (Switchover), consulta il riepilogo dello switchover. Assicurati che le risorse in entrambi gli ambienti corrispondano a quelle previste. In caso contrario, scegli Cancel (Annulla).
5. In Impostazioni di timeout, inserisci il limite di tempo per lo switchover.
6. Se sul cluster è in esecuzione Aurora PostgreSQL, esamina e verifica i suggerimenti prima dello switchover. Per ulteriori informazioni, consulta the section called “Limitazioni specifiche della replica logica per le distribuzioni blu/verdi”.
7. Seleziona Switch over (Switchover).

AWS CLI

Per passare da una distribuzione blu/verde utilizzando il AWS CLI, usa il switchover-blue-green-deploymentcomando con le seguenti opzioni:

- `--blue-green-deployment-identifier`— Specificare l'ID della risorsa della distribuzione blu/verde.
- `--switchover-timeout`: specifica il limite di tempo per lo switchover, in secondi. Il valore predefinito è 300.

Example Switchover di un'implementazione blu/verde

In Linux, macOS, oppure Unix:

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

In Windows:

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

API RDS

Per eseguire lo switchover di una implementazione blu/verde utilizzando l'API Amazon RDS, usa l'operazione `SwitchoverBlueGreenDeployment` con i seguenti parametri:

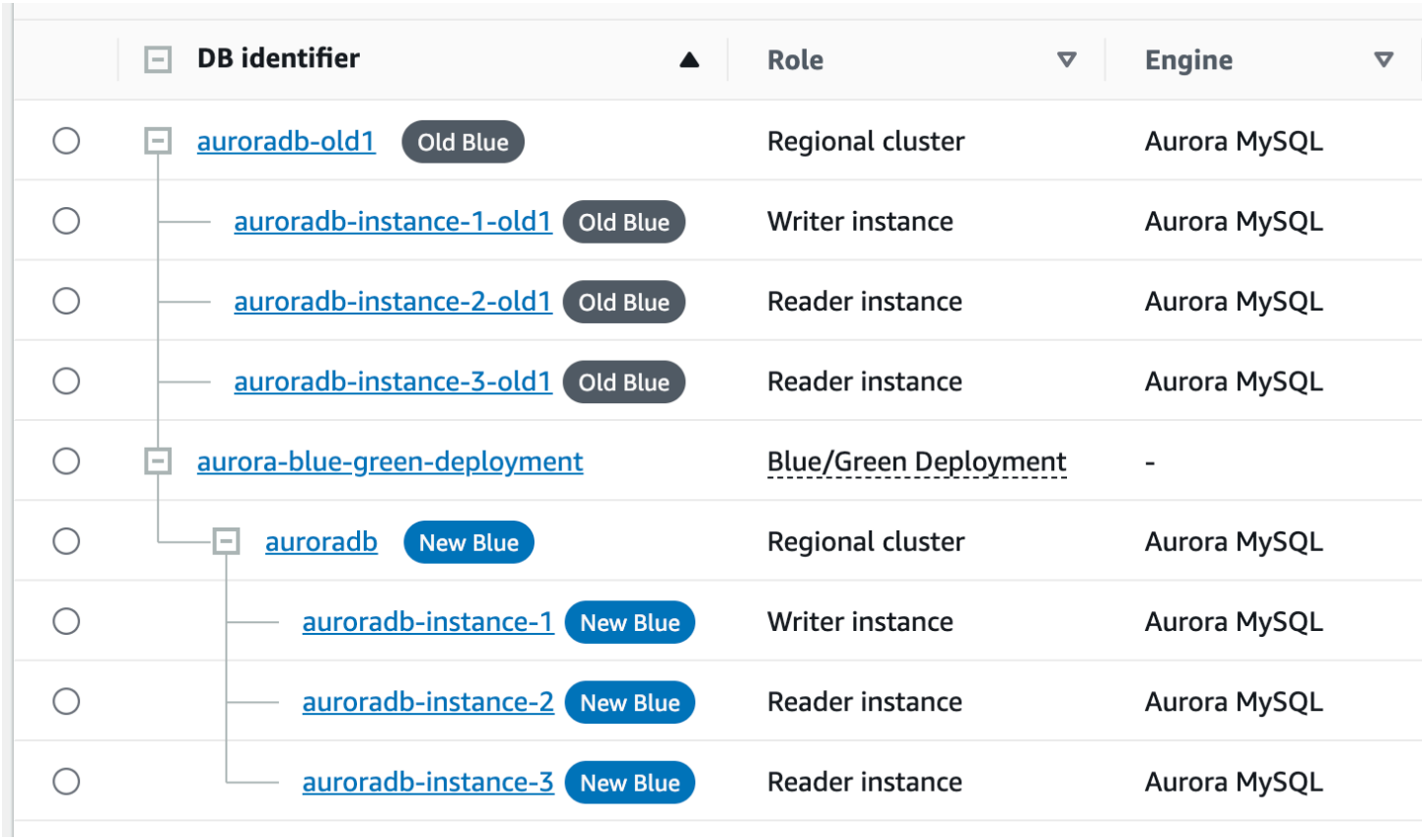
- `BlueGreenDeploymentIdentifier`— Specificare l'ID della risorsa della distribuzione blu/verde.
- `SwitchoverTimeout`: specifica il limite di tempo per lo switchover, in secondi. Il valore predefinito è 300.

Dopo lo switchover

Dopo uno switchover, il cluster di database e le istanze database vengono mantenuti nell'ambiente blu precedente. A queste risorse si applicano i costi standard. La replica e il log binario tra gli ambienti blu e verde vengono arrestati.

RDS rinomina il cluster di database e le istanze database nell'ambiente blu aggiungendo `-old`*n* al nome corrente, dove *n* è un numero. Il cluster DB blu viene forzato in uno stato di sola lettura. Anche se abiliti le operazioni di scrittura, rimane di sola lettura finché non elimini la distribuzione blu/verde. RDS nomina il cluster di istanze DB e DB nell'ambiente verde. `-new`*n*

Se si elimina la risorsa di distribuzione blu/verde, RDS conserva le risorse and. `-old`*n* `-new`*n*



Aggiornamento del nodo principale per i consumatori

RDS offre repliche di lettura completamente gestite. Tuttavia, offre anche la possibilità di configurare repliche autogestite, note anche come repliche esterne. Le repliche esterne consentono di utilizzare risorse di terze parti come obiettivi di replica.

Dopo aver cambiato una distribuzione blu/verde, se il cluster DB di istanze DB relativo nodo principale dopo il passaggio per mantenere la continuità della replica.

Per aggiornare il nodo principale

1. Dopo il passaggio, l'istanza Writer DB che in precedenza si trovava nell'ambiente verde emette un evento che contiene il nome del file di registro principale e la posizione del registro principale.

Per individuare l'evento, accedi alla console RDS e scegli Eventi dal riquadro di navigazione a sinistra.

2. Filtra per eventi la cui fonte è il nome della vecchia istanza di Green Writer DB, prima dello switchover.
3. Individua l'evento che contiene le coordinate del log binario. Il messaggio dell'evento è simile a:`Binary log coordinates in green environment after switchover: file mysql-bin-changelog.`*`000003`* `and position` *`40134574`*.
4. Assicurati che il consumatore o la replica abbiano applicato tutti i log binari del vecchio ambiente blu. Quindi, utilizzate le coordinate del log binario fornite per riprendere la replica sui consumatori. Ad esempio, se stai eseguendo una replica MySQL EC2 su, puoi usare i seguenti comandi:

   MySQL 8.0.22 e versioni principali e secondarie precedenti

   ```
   CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-changelog.000003', MASTER_LOG_POS=40134574;
   ```

   MySQL 8.0.23 e versioni successive principali e secondarie

   ```
   CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
    SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
   ```

Eliminazione di una distribuzione blu/verde in Amazon RDS

È possibile eliminare l'implementazione blu/verde prima o dopo lo switchover.

Quando elimini un'implementazione blu/verde prima dello switchover, Amazon RDS elimina facoltativamente il cluster di database nell'ambiente verde:

- Se scegli di eliminare il cluster di database nell'ambiente verde (`--delete-target`), per tale cluster la protezione dall'eliminazione deve essere disattivata.
- Se non elimini il cluster di database nell'ambiente verde (`--no-delete-target`), il cluster viene mantenuto ma non fa più parte di un'implementazione blu/verde. Per , la replica continua tra gli ambienti. Per Aurora PostgreSQL, l'ambiente verde viene promosso a ambiente autonomo, quindi la replica si interrompe.

L'opzione per eliminare i database verdi non è disponibile nella console dopo lo switchover. Quando si eliminano le distribuzioni blu/verdi utilizzando, non è possibile specificare l'opzione se lo stato della distribuzione è. AWS CLI`--delete-target` `SWITCHOVER_COMPLETED`

> **Important**
>
> Dopo aver eliminato una distribuzione blu/verde, RDS rimuove le protezioni di sola lettura dal precedente cluster DB di produzione. Se il `read_only` parametro è disabilitato per il cluster DB, riprende a consentire le operazioni di scrittura.

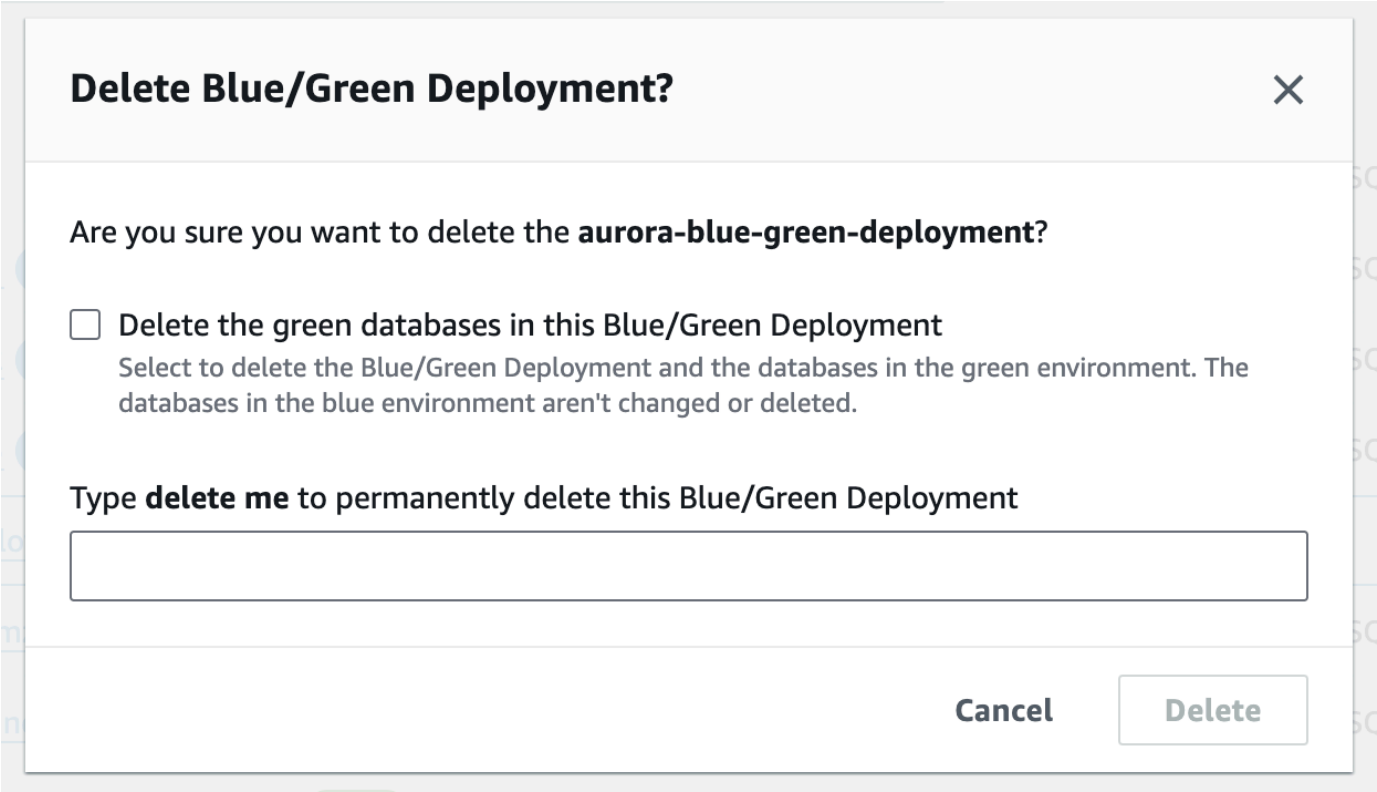
È possibile eliminare una distribuzione blu/verde utilizzando l'API AWS Management Console AWS CLI, the o RDS.

Console

Per eliminare un'implementazione blu/verde

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database) e seleziona l'implementazione blu/verde da eliminare.
3. In Actions (Azioni), scegliere Delete (Elimina).

   La distribuzione Blue/Green di Delete? Viene visualizzata la finestra.



Per eliminare i database verdi, seleziona Elimina i database verdi in questa distribuzione blu/verde.

4. Immettere **`delete me`** nella casella.
5. Scegliere Delete (Elimina).

AWS CLI

Per eliminare una distribuzione blu/verde utilizzando il AWS CLI, usa il delete-blue-green-deploymentcomando con le seguenti opzioni:

- `--blue-green-deployment-identifier`— L'ID della risorsa della distribuzione blu/verde da eliminare.
- `--delete-target`: specifica che il cluster di database nell'ambiente verde viene eliminato. Non è possibile specificare questa opzione se lo stato dell'implementazione blu/verde è `SWITCHOVER_COMPLETED`.
- `--no-delete-target`: specifica che il cluster di database nell'ambiente verde viene mantenuto.

Example Eliminazione di un'implementazione blu/verde e del cluster di database nell'ambiente verde

In Linux, macOS, oppure Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

In Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Example Eliminazione di un'implementazione blu/verde, mantenendo il cluster di database nell'ambiente verde

In Linux, macOS, oppure Unix:

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

In Windows:

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

API RDS

Per eliminare un'implementazione blu/verde con l'API Amazon RDS, utilizza l'operazione `DeleteBlueGreenDeployment` con i seguenti parametri:

- `BlueGreenDeploymentIdentifier`— L'ID della risorsa della distribuzione blu/verde da eliminare.
- `DeleteTarget`: specifica `TRUE` per eliminare il cluster di database nell'ambiente verde o `FALSE` per mantenerlo. Non può essere `TRUE` se lo stato dell'implementazione blu/verde è `SWITCHOVER_COMPLETED`.

Backup e ripristino di un cluster DB Amazon Aurora

Questi argomenti forniscono informazioni sul backup e sul ripristino dei cluster database Amazon Aurora.

> **Tip**
>
> Le funzioni di Aurora a elevata disponibilità e le funzionalità di backup automatico consentono di proteggere i dati senza richiedere una configurazione completa. Prima di implementare una strategia di backup, scopri i modi in cui Aurora conserva più copie dei tuoi dati e ti aiuta ad accedervi su più istanze e regioni DB. AWS Per informazioni dettagliate, consultare Elevata disponibilità di Amazon Aurora.

Argomenti

- Panoramica di backup e ripristino di un cluster di database Aurora
- Mantenimento dei backup automatici
- Informazioni sull'utilizzo dello storage di backup Amazon Aurora
- Creazione di uno snapshot del cluster database
- Ripristino da uno snapshot cluster database
- Copia di snapshot del cluster DB
- Condivisione di uno snapshot cluster database
- Esportazione dei dati del cluster database in Amazon S3
- Esportazione dei dati dello snapshot del cluster di database in Amazon S3
- Ripristino di un cluster di database a un determinato momento
- Eliminazione di una snapshot del cluster di database
- Tutorial: Ripristino di un cluster database Amazon Autora da uno snapshot cluster DB

Panoramica di backup e ripristino di un cluster di database Aurora

Negli argomenti seguenti vengono descritti i backup Aurora e come ripristinare il cluster database Aurora.

Indice

- Backup
  - Utilizzo AWS Backup
- Finestra di backup
- Ripristino dei dati
- Clonazione dei database per Aurora
- Backtrack

Backup

Aurora esegue automaticamente il backup del volume del cluster e conserva i dati per la durata del tempo di conservazione del backup. I backup automatizzati Aurora sono continui e incrementali, in modo da poter eseguire rapidamente un ripristino da qualsiasi point-in-time incluso nel periodo di conservazione del backup. Durante la scrittura dei dati di backup, non si verifica alcun impatto sulle prestazioni o interruzione del funzionamento del servizio del database. È possibile specificare un periodo di conservazione del backup (da 1 a 35 giorni) quando un cluster database viene creato o modificato. I backup automatizzati Aurora sono archiviati in Amazon S3. Per ulteriori informazioni sulla conservazione dei backup automatici, vedere Mantenimento dei backup automatici

Per mantenere i dati oltre il tempo di conservazione del backup, è possibile eseguire uno snapshot dei dati del volume del cluster. Gli snapshot dei cluster Aurora non scadono. Puoi creare un nuovo cluster database dallo snapshot. Per ulteriori informazioni, consulta Creazione di uno snapshot del cluster database.

> **Note**
>
> - Per i cluster di database Amazon Aurora, il periodo di retention dei backup predefinito è di un giorno indipendentemente dal modo in cui viene creato il cluster DB.
> - Non puoi disabilitare i backup automatici su Aurora. Il periodo di retention dei backup per Aurora viene gestito dal cluster di database.

I costi dello storage di backup dipendono dalla quantità di dati di backup e snapshot Aurora mantenuti e dalla durata di mantenimento. Per informazioni sullo storage associato ai backup e agli snapshot Aurora, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora. Per informazioni sui prezzi dello storage di backup Aurora, consulta i prezzi di Amazon for RDS Aurora. Dopo che un cluster Aurora associato a uno snapshot viene eliminato, per l'archiviazione di tale snapshot verranno applicati i costi dello storage standard di Aurora.

Utilizzo AWS Backup

È possibile utilizzare… AWS Backup per gestire i backup dei cluster Amazon Aurora DB.

Istantanee gestite da AWS Backup sono considerate istantanee manuali del cluster DB, ma non vengono conteggiate ai fini della quota di snapshot del cluster DB per Aurora. Istantanee create con AWS Backup hanno nomi con`awsbackup:job-`*`AWS-Backup-job-number`*. Per ulteriori informazioni sull' AWS Backup, vedi il AWS Guida per gli sviluppatori di Backup .

Puoi anche usare AWS Backup per gestire i backup automatici dei cluster Amazon Aurora DB. Se il cluster DB è associato a un piano di backup in AWS Backup, è possibile utilizzare tale piano di backup per point-in-time il ripristino. Backup automatici (continui) gestiti da AWS Backup hanno nomi con`continuous:cluster-`*`AWS-Backup-job-number`*. Per ulteriori informazioni, consulta Ripristino di un cluster DB a un orario specificato utilizzando AWS Backup.

Finestra di backup

I backup automatici vengono effettuati quotidianamente durante la finestra di backup scelta. Se il backup richiede più tempo rispetto alla finestra di backup prevista, il backup continua dopo il termine della finestra, finché non viene completato. La finestra di backup non può sovrapporsi con la finestra di manutenzione settimanale per il cluster database.

I backup automatici di Aurora sono continui e incrementali, tuttavia si utilizza una finestra di backup per creare un backup giornaliero del sistema che viene conservato per tutto il periodo di conservazione del backup. È possibile copiare il backup per conservarlo oltre il periodo di conservazione.

> **Note**
>
> Quando si crea un cluster DB utilizzando il AWS Management Console, non è possibile specificare una finestra di backup. Tuttavia, è possibile specificare una finestra di backup quando si crea un cluster DB utilizzando il AWS CLI oppure RDSAPI.

Se non si specifica una finestra di backup preferita al momento della creazione del cluster, Aurora assegna una finestra di backup predefinita di 30 minuti. Questa finestra viene selezionata a caso da un intervallo di tempo di 8 ore per ciascuna Regione AWS. La tabella seguente elenca i blocchi temporali per ciascuno Regione AWS da cui vengono assegnate le finestre di backup predefinite.

Nome della regione	Regione	Periodo di tempo
US East (N. Virginia)	us-east-1	03:00 — 11:00 UTC
Stati Uniti orientali (Ohio)	us-east-2	03:00 — 11:00 UTC
US West (N. Californi a)	us-west-1	06:00 — 14:00 UTC
US West (Oregon)	us-west-2	06:00 — 14:00 UTC
Africa (Cape Town)	af-south-1	03:00 — 11:00 UTC
Asia Pacifico (Hong Kong)	ap-east-1	06:00 — 14:00 UTC
Asia Pacific (Hyderabad)	ap-south-2	06:30 — 14:30 UTC
Asia Pacifico (Giacarta)	ap-southeast-3	08:00 — 16:00 UTC
Asia Pacifico (Malesia)	ap-southeast-5	09:00 — 17:00 UTC
Asia Pacifico (Melbourne)	ap-southeast-4	11:00 — 19:00 UTC
Asia Pacific (Mumbai)	ap-south-1	16:30 — 00:30 UTC
Asia Pacifico (Osaka-Locale)	ap-northeast-3	00:00 — 08:00 UTC
Asia Pacifico (Seul)	ap-northeast-2	13:00 — 21:00 UTC

Nome della regione	Regione	Periodo di tempo
Asia Pacific (Singapore)	ap-southeast-1	14:00 — 22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	12:00 — 20:00 UTC
Asia Pacifico (Tokyo)	ap-northeast-1	13:00 — 21:00 UTC
Canada (Central)	ca-central-1	03:00 — 11:00 UTC
Canada occidentale (Calgary)	ca-west-1	18:00 — 02:00 UTC
China (Beijing)	cn-north-1	06:00 — 14:00 UTC
Cina (Ningxia)	cn-northwest-1	06:00 — 14:00 UTC
Europe (Frankfurt)	eu-central-1	20:00 — 04:00 UTC
Europa (Irlanda)	eu-west-1	22:00 — 06:00 UTC
Europe (London)	eu-west-2	22:00 — 06:00 UTC
Europa (Milano)	eu-south-1	02:00 — 10:00 UTC
Europe (Paris)	eu-west-3	07:29 — 14:29 UTC
Europa (Spagna)	eu-south-2	02:00 — 10:00 UTC
Europa (Stoccolma)	eu-north-1	23:00 — 07:00 UTC
Europa (Zurigo)	eu-central-2	02:00 — 10:00 UTC
Israele (Tel Aviv)	il-central-1	03:00 — 11:00 UTC
Medio Oriente (Bahrein)	me-south-1	06:00 — 14:00 UTC
Medio Oriente () UAE	me-central-1	05:00 — 13:00 UTC

Nome della regione	Regione	Periodo di tempo
Sud America (São Paulo)	sa-east-1	23:00 — 07:00 UTC
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	17:00 — 01:00 UTC
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	06:00 — 14:00 UTC

Ripristino dei dati

È possibile recuperare i dati creando un nuovo cluster database Aurora dai dati di backup mantenuti da Aurora, da uno snapshot del cluster database salvato o da un backup automatico mantenuto. Puoi ripristinare rapidamente una nuova copia di un cluster DB creato dai dati di backup a un momento qualsiasi del periodo di retention dei backup. Data la natura continua e incrementale dei backup Aurora durante il tempo di conservazione del backup, non sarà necessario effettuare snapshot frequenti dei dati per migliorare i tempi di ripristino.

L'ultimo orario di ripristino di un cluster database corrisponde al punto più recente in corrispondenza del quale è possibile ripristinare il cluster database. In genere, ciò avviene entro 5 minuti dall'ora corrente per un cluster database attivo o 5 minuti dall'orario di eliminazione del cluster per un backup automatico mantenuto.

Il primo orario di ripristino indica il punto all'interno del periodo di conservazione dei backup dai cui è possibile ripristinare il volume del cluster.

Per determinare l'ora di ripristino più recente o più antica per un cluster di database, cerca i `Earliest restorable time` valori `Latest restorable time` o sulla console. RDS Per informazioni sulla visualizzazione di tali valori consulta Visualizzazione dei backup automatici conservati per Amazon Aurora.

Puoi stabilire il completamento del ripristino di un cluster DB verificando i valori `Latest restorable time` ed `Earliest restorable time`. Questi valori vengono restituiti NULL fino al completamento dell'operazione di ripristino. Non è possibile richiedere un'operazione di backup o ripristino se una delle due `Latest restorable time` viene `Earliest restorable time` restituitaNULL.

Per informazioni sul ripristino di un cluster di database a un punto temporale specifico, consulta Ripristino di un cluster di database a un determinato momento.

Clonazione dei database per Aurora

Anziché ripristinare la snapshot di un cluster DB, puoi anche clonare i database del cluster DB Aurora in un nuovo cluster DB. I database dei cloni utilizzano un ridottissimo spazio aggiuntivo al momento della creazione. I dati vengono copiati solo come modifiche nel database di origine o in quelli clone. Puoi effettuare più cloni dello stesso cluster DB oppure creare cloni aggiuntivi da altri cloni. Per ulteriori informazioni, consulta Clonazione di un volume per un cluster di database Amazon Aurora.

Backtrack

Aurora My SQL ora supporta il «riavvolgimento» di un cluster DB a un orario specifico, senza ripristinare i dati da un backup. Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.

Mantenimento dei backup automatici

Quando si elimina un oggetto predisposto o Aurora Serverless v2 Cluster DB, è possibile conservare i backup automatici. Ciò consente di ripristinare un cluster database in un punto nel tempo specifico all'interno del periodo di conservazione del backup, anche dopo l'eliminazione del cluster.

I backup automatici mantenuti contengono snapshot di sistema e log delle transazioni di un cluster database. Includono anche le proprietà del cluster database, come la classe di istanza database, necessarie per ripristinarlo in un cluster attivo.

È possibile ripristinare o rimuovere i backup automatici conservati utilizzando AWS Management Console, e RDS API AWS CLI.

> **Note**
>
> Non è possibile conservare i backup automatici per Aurora Serverless v1 Cluster DB.

Argomenti

- Periodo di conservazione
- Costi di retention
- Impedire l'eliminazione automatica dei backup
- Limitazioni
- Visualizzazione dei backup automatici conservati per Amazon Aurora
- Eliminazione dei backup automatici conservati per Amazon Aurora

Periodo di conservazione

Gli snapshot di sistema e i log delle transazioni di un backup automatico mantenuto scadono allo stesso modo del cluster database di origine. Le impostazioni per il periodo di conservazione del cluster di origine si applicano anche ai backup automatici. Poiché non vengono creati nuovi snapshot o log per questo cluster, i backup automatici mantenuti scadranno. Al termine del periodo di conservazione, si continua a mantenere gli snapshot del cluster database manuali, ma tutti i backup automatici scadranno.

È possibile rimuovere i backup automatici conservati utilizzando la console, AWS CLI oppure. RDS API Per ulteriori informazioni, consulta Eliminazione dei backup automatici conservati per Amazon Aurora.

A differenza di un backup automatico mantenuto, uno snapshot finale non scade. Ti suggeriamo di acquisire uno snapshot finale anche se mantieni i backup automatici, perché i backup automatici mantenuti prima o poi scadono.

Costi di retention

Per lo storage di backup non vengono addebitati costi aggiuntivi fino al 100% dello storage di database Aurora totale per ogni cluster database Aurora. Inoltre, non sono previsti costi aggiuntivi fino a un giorno se si mantengono i backup automatici dopo l'eliminazione di un cluster database. I backup mantenuti per più di un giorno vengono addebitati.

Non sono previsti costi aggiuntivi per i log delle transazioni o i metadati dell'istanza. Tutte le altre regole di prezzo per i backup si applicano ai cluster ripristinabili. Per ulteriori informazioni, consulta la pagina Prezzi di Amazon Aurora.

Impedire l'eliminazione automatica dei backup

Amazon RDS elimina i backup automatici in diverse situazioni:

- Al termine del periodo di conservazione.
- Quando si disabilitano i backup automatici per un cluster di database.
- Quando si elimina un cluster di database.

Se si desidera mantenere uno snapshot automatico per un periodo più lungo, è possbile copiarlo e creare uno snapshot manuale che sarà conservato finché non lo si elimina personalmente. I costi RDS di storage di Amazon potrebbero essere applicati agli snapshot manuali se superano lo spazio di archiviazione predefinito.

Per ulteriori informazioni sulla copia di uno snapshot del cluster DB, consulta. Copia di snapshot del cluster DB

Per ulteriori informazioni sui costi dello storage di backup, consulta i RDSprezzi di Amazon.

Limitazioni

Le seguenti limitazioni si applicano ai backup automatici mantenuti:

- Il numero massimo di backup automatici conservati in uno AWS La regione è 40. Non è incluso nella quota per i cluster database. È possibile avere fino a 40 cluster database in esecuzione, 40 istanze database in esecuzione e 40 backup automatici mantenuti per i cluster database contemporaneamente.

  Per ulteriori informazioni, consulta Quote in Amazon Aurora.
- I backup automatici mantenuti non contengono informazioni relative ai parametri o ai gruppi di opzioni.
- È possibile ripristinare un cluster eliminato in un punto temporale che si trova all'interno del periodo di conservazione al momento dell'eliminazione.
- Non è possibile modificare un backup automatico mantenuto perché è costituito da backup di sistema, log delle transazioni e proprietà del cluster database presenti al momento dell'eliminazione del cluster di origine.

Visualizzazione dei backup automatici conservati per Amazon Aurora

Per visualizzare i backup automatici conservati nella RDS console, scegli Backup automatici nel riquadro di navigazione, quindi scegli Conservati. Per visualizzare singoli snapshot associati a un backup automatico mantenuto, scegli Snapshot nel pannello di navigazione. In alternativa, puoi descrivere singoli snapshot associati a un backup automatico mantenuto. Da qui, puoi ripristinare un'istanza database direttamente da uno di tali snapshot.

Per descrivere i backup automatici conservati utilizzando il AWS CLI, utilizzare il seguente comando:

```
aws rds describe-db-cluster-automated-backups --db-cluster-resource-
id DB_cluster_resource_ID
```

Per descrivere i backup automatici conservati utilizzando il RDSAPI, richiamate l'`DescribeDBClusterAutomatedBackups`azione con il `DbClusterResourceId` parametro.

Eliminazione dei backup automatici conservati per Amazon Aurora

Puoi eliminare i backup automatici mantenuti quando non servono più. Per eliminare un backup automatico conservato utilizzando l' AWS Management Console API Amazon RDS o Amazon RDS, utilizza le seguenti procedure. AWS CLI

Console

Per eliminare i backup automatici mantenuti

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Automated backups (Backup automatici).
3. Scegli la scheda Conservato.

4. Scegliere il backup automatico mantenuto da eliminare.
5. In Actions (Azioni), selezionare Delete (Elimina).
6. Nella pagina di conferma, immetti **`delete me`** e seleziona Elimina.

AWS CLI

È possibile eliminare un backup automatico conservato utilizzando il AWS CLI comando delete-db-cluster-automated-backup con la seguente opzione:

- `--db-cluster-resource-id`: l'identificatore della risorsa per il cluster database di origine.

  È possibile trovare l'identificatore di risorsa per il cluster DB di origine di un backup automatizzato mantenuto eseguendo il comando -backups. AWS CLI describe-db-cluster-automated

Example

Questo esempio elimina il backup automatico mantenuto per il cluster database di origine con l'ID risorsa `cluster-123ABCEXAMPLE`.

PerLinux, o: macOS Unix

```
aws rds delete-db-cluster-automated-backup \
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

Per Windows:

```
aws rds delete-db-cluster-automated-backup ^
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

API RDS

Puoi eliminare un backup automatico mantenuto utilizzando l'operazione Delete dell'API Amazon RDS DBCluster AutomatedBackup con il seguente parametro:

- `DbClusterResourceId`: l'identificatore della risorsa per il cluster database di origine.

  Puoi trovare l'identificatore di risorsa per l'istanza DB di origine di un backup automatizzato mantenuto utilizzando l'operazione Descrivi dell'API Amazon RDS. DBCluster AutomatedBackups

Informazioni sull'utilizzo dello storage di backup Amazon Aurora

Amazon Aurora gestisce due tipi di backup: backup automatici (continui) e istantanee.

Storage di backup automatici

Il backup automatico (continuo) per un cluster archivia in modo incrementale tutte le modifiche apportate al database in un periodo di conservazione specificato in modo da poter eseguire il ripristino in qualsiasi momento entro il periodo di conservazione definito. I periodi di conservazione possono variare da 1 a 35 giorni. I backup automatici sono incrementali e il relativo costo viene addebitato in base alla quantità di storage necessaria per il ripristino in qualsiasi momento entro il periodo di conservazione.

Aurora offre anche una quantità di utilizzo gratuito del backup. Questa quantità di utilizzo gratuita è pari alla dimensione del volume del cluster più recente (come rappresentata dal CloudWatch parametro `VolumeBytesUsed` Amazon). Questa quantità viene sottratta dall'utilizzo del backup automatico calcolato. Inoltre, non è previsto alcun costo per backup automatici il cui periodo di conservazione è solo di 1 giorno.

Ad esempio, il backup automatico ha un periodo di conservazione di 7 giorni e quindi puoi ripristinare il cluster allo stato di quattro giorni fa. Aurora utilizza i dati incrementali archiviati nel backup automatico per ricreare lo stato del cluster all'ora esatta di quattro giorni fa.

Il backup automatico archivia tutte le informazioni necessarie per poter ripristinare il cluster in qualsiasi momento compreso nella finestra di conservazione. Ciò significa che archivia tutte le modifiche durante il periodo di conservazione, incluse le operazioni di scrittura di nuove informazioni o di eliminazione delle informazioni esistenti. Per i database a cui vengono apportate molte modifiche, la dimensione del backup automatico aumenta nel tempo. Non appena un database non viene più modificato, puoi aspettarti che la dimensione del backup automatico diminuisca, man mano che le modifiche precedentemente archiviate non rientrano più nella finestra di conservazione.

L'utilizzo totale fatturato per il backup automatico non supera mai la dimensione cumulativa del volume del cluster durante il periodo di conservazione. Ad esempio, se il periodo di conservazione è di 7 giorni e il volume del cluster è di 100 GB al giorno, l'utilizzo fatturato del backup automatico non supera mai 700 GB (100 GB*7).

Archiviazione delle istantanee

Le istantanee dei cluster DB sono sempre backup completi che acquisiscono le dimensioni del volume del cluster al momento della creazione. Sia che si scattino istantanee manualmente o tramite un piano di AWS backup, Aurora le tratta come istantanee manuali. Aurora offre spazio di archiviazione gratuito illimitato per le istantanee entro il periodo di conservazione dei backup automatizzati. Una volta superata tale periodo, un'istantanea manuale comporta addebiti per GB al mese. Le istantanee di sistema automatizzate rimangono gratuite a meno che non vengano copiate. Poiché i backup automatici non coprono le copie delle istantanee, le fatturano AWS sempre per conto proprio.

Per informazioni generali sui backup Aurora, consulta Backup. Per informazioni sui prezzi dello storage di backup Aurora, consulta la pagina dei prezzi di Amazon Aurora.

CloudWatch Parametri Amazon per lo storage di backup Aurora

Puoi monitorare i tuoi cluster Aurora e creare report utilizzando i CloudWatch parametri di Amazon tramite la console. CloudWatch È possibile utilizzare le seguenti CloudWatch metriche per esaminare e monitorare la quantità di storage utilizzata dai backup Aurora. Questi parametri vengono calcolati in modo indipendente per ogni cluster DB Aurora.

- `BackupRetentionPeriodStorageUsed`: rappresenta la quantità di spazio di archiviazione di backup, in byte, utilizzato al momento per l'archiviazione dei backup automatizzati.
  - Il valore dipende dalla dimensione del volume del cluster e dal numero di modifiche (scritture e aggiornamenti) apportate al cluster database durante il periodo di conservazione. Questo è dovuto al fatto che il backup automatico deve archiviare tutte le modifiche incrementali apportate al cluster per poterle ripristinare in qualsiasi momento.
  - Il valore di questa metrica non viene sottratto dal livello gratuito di utilizzo del backup fornito da Aurora.
  - Questa metrica restituisce un unico punto dati giornaliero per l'utilizzo del backup automatico registrato in un giorno specifico.
- `SnapshotStorageUsed`: rappresenta la quantità di archiviazione dei backup utilizzata, in byte, per l'archiviazione di istantanee manuali oltre il periodo di conservazione dei backup.
  - Il valore dipende dal numero di istantanee conservate oltre il periodo di conservazione del backup automatico e dalle dimensioni di ciascuna istantanea.
  - Ogni snapshot corrisponde, per dimensioni, al volume del cluster al momento della sua acquisizione.

- Gli snapshot sono backup completi, non incrementali.
- Questa metrica restituisce un punto dati giornaliero per ogni istantanea addebitata. Per recuperare l'utilizzo totale giornaliero delle istantanee, calcola la somma di questa metrica in un periodo di 1 giorno.

- `TotalBackupStorageBilled`: rappresenta le metriche per tutti gli utilizzi di backup fatturati, in byte, per il cluster specificato:

  `BackupRetentionPeriodStorageUsed + SnapshotStorageUsed - free tier`

  - Questa metrica restituisce un punto dati giornaliero per il valore `BackupRetentionPeriodStorageUsed` meno il livello gratuito di utilizzo del backup fornito da Aurora. Questo livello gratuito corrisponde all'ultima dimensione registrata del volume del cluster database. Questo punto dati rappresenta l'utilizzo effettivo fatturato per il backup automatico.
  - Questa metrica restituisce singoli punti dati giornalieri per tutti i valori `SnapshotStorageUsed`.
  - Per recuperare l'utilizzo totale giornaliero fatturato del backup, calcola la somma di questa metrica in un periodo di 1 giorno. L'utilizzo fatturato delle istantanee viene sommato all'utilizzo del backup automatico fatturato per ottenere l'utilizzo totale del backup fatturato.

Per ulteriori informazioni su come utilizzare le CloudWatch metriche, consulta. Disponibilità delle metriche Aurora nella console Amazon RDS

Calcolo dell'utilizzo dello storage dei backup

L'utilizzo di un backup automatico viene calcolato facendo riferimento a tutti i record incrementali che devono essere archiviati, per poterli ripristinare in qualsiasi momento durante il periodo di conservazione del backup.

Si supponga, ad esempio, di avere un backup automatico con periodo di conservazione di 7 giorni. La dimensione del volume del cluster prima del periodo di conservazione è pari a 100 GB, quindi questo valore è la quantità minima che Aurora deve archiviare. Nei successivi 7 giorni viene eseguita la seguente attività, in cui la dimensione incrementale dei record corrisponde alla quantità di storage necessaria per archiviare i record delle modifiche provenienti dalle operazioni di scrittura e aggiornamento del database.

Day (Giorno)	Dimensioni incrementali dei record (GB)
1	10
2	15
3	25
4	20
5	10
6	25
7	30
Totale	135

Questi dati indicano che l'utilizzo del backup automatico calcolato per il backup è il seguente:

```
100 GB (volume size before retention period) + 135 GB (size of incremental records) =
 235 GB total backup usage
```

Dall'utilizzo fatturato viene quindi sottratto il livello di utilizzo gratuito. Supponiamo che la dimensione più recente del volume sia 200 GB:

```
235 GB total backup usage - 200 GB (latest volume size) = 35 GB billed backup usage
```

FAQs

Quando vengono fatturate le istantanee?

Vengono fatturate le istantanee manuali che non rientrano nel periodo di conservazione del backup automatico (ovvero precedenti al periodo indicato).

Cos'è un'istantanea manuale?

Un'istantanea manuale è un'istantanea per la quale è valida una delle seguenti condizioni:

- Viene richiesta manualmente dall'utente

- Preso da un servizio di backup automatizzato come AWS Backup
- Viene copiata da un'istantanea di sistema automatica per essere conservata oltre il periodo di conservazione

Cosa succede alle istantanee manuali se elimino il cluster database?

Le istantanee manuali non scadono finché non scegli di eliminarle.

Quando elimini il cluster di database, le istantanee manuali acquisite in precedenza continuano a esistere. Se in precedenza queste istantanee non venivano fatturate perché rientravano nel periodo di conservazione dei backup automatici, ora non sono più coperte e iniziano a essere fatturate per intero in base al loro utilizzo.

Come posso ridurre i costi dell'archiviazione dei backup?

Esistono alcuni modi per ridurre i costi relativi all'utilizzo dei backup:

- Elimina le istantanee manuali che non rientrano nel periodo di conservazione dei backup automatici. Ciò include le istantanee che hai scattato, nonché le istantanee che il tuo piano di AWS Backup potrebbe aver scattato. Controlla il piano di Backup AWS per assicurarti che non conservi le istantanee al di fuori del periodo di conservazione previsto.
- Valuta le operazioni di scrittura e aggiornamento eseguite sul database per vedere se puoi ridurre il numero di modifiche che stai apportando. Poiché il backup automatico archivia tutte le modifiche incrementali nel il periodo di conservazione definito, la riduzione del numero di aggiornamenti da effettuare riduce anche i costi del backup automatico.
- Valuta se è ragionevole ridurre il periodo di conservazione del backup automatico. La riduzione del periodo di conservazione significa che il backup archivia meno giorni di dati incrementali, il che potrebbe ridurre il costo complessivo del backup. Tuttavia, la riduzione di questo periodo potrebbe anche comportare la fatturazione di alcune istantanee in quanto non rientrano più nel periodo di conservazione. Assicurati di controllare tutti i costi aggiuntivi per le istantanee che potresti dover sostenere prima di decidere se questa è la linea d'azione più idonea.

Come viene fatturato lo storage dei backup?

Lo storage di backup è fatturato per GB/mese.

Ciò significa che l'utilizzo dello storage di backup viene addebitato come la media ponderata dell'utilizzo durante il mese specificato. Ecco alcuni esempi riferiti a un mese di 30 giorni:

- L'utilizzo fatturato del backup è pari a 100 GB per tutti i 30 giorni del mese. Il costo viene addebitato nel seguente modo:

```
(100 GB * 30) / 30 = 100 GB-month
```

- L'utilizzo fatturato del backup è pari a 100 GB per i primi 15 giorni del mese, quindi a 0 GB per gli ultimi 15. Il costo viene addebitato nel seguente modo:

```
(100 GB * 15 + 0 GB * 15) / 30 = 50 GB-month
```

- L'utilizzo fatturato del backup è pari a 50 GB per i primi 10 giorni del mese, a 100 GB per i successivi 10 giorni, quindi a 150 GB per gli ultimi 10. Il costo viene addebitato nel seguente modo:

```
(50 GB * 10 + 100 GB * 10 + 150 GB * 10) / 30 = 100 GB-month
```

In che modo l'impostazione del backtracking per il cluster database influisce sull'utilizzo dello spazio di archiviazione del backup?

L'impostazione di backtrack per un cluster DB Aurora non influenza il volume dei dati di backup per tale cluster. Aurora fattura separatamente l'archiviazione per dati di backtracking. Per informazioni sui prezzi del backtracking Aurora, consulta la pagina Prezzi di Amazon Aurora.

Come si applicano i costi di archiviazione alle istantanee condivise?

Se si condivide uno snapshot con un altro utente, si continua a essere il proprietario di tale snapshot. I costi dello storage spettano al proprietario della snapshot. Se elimini una snapshot condivisa di tua proprietà, nessuno potrà accedervi.

Per mantenere l'accesso a una snapshot condivisa di proprietà di un altro utente, puoi copiarla. In questo modo diventerai il proprietario della nuova snapshot. Eventuali costi di storage per la snapshot copiata si applicano al tuo account.

Per ulteriori informazioni sugli snapshot di condivisione, consulta Condivisione di uno snapshot cluster database. Per ulteriori informazioni sulla copia di snapshot, consulta Copia di snapshot del cluster DB.

Creazione di uno snapshot del cluster database

Amazon RDS crea uno snapshot del volume di storage del tuo cluster DB, eseguendo il backup dell'intero cluster DB e non solo dei singoli database. Quando crei uno snapshot di cluster database è necessario identificare qual è il cluster database di cui stai effettuando il backup e dare un nome alla snapshot di cluster database in modo da poterlo usare successivamente per il ripristino. La quantità di tempo necessaria per creare una snapshot di cluster di database varia a seconda della dimensione dei database. Poiché lo snapshot include l'intero volume d'archiviazione, la dimensione dei file, come i file temporanei, influisce sulla quantità di tempo necessaria per creare lo snapshot.

> **Note**
>
> Il tuo cluster database deve essere nello stato `available` per poter acquisire uno snapshot del cluster database.

A differenza dei backup automatizzati, gli snapshot manuali non sono soggetti al periodo di retention dei backup. Gli snapshot non scadono.

Per i backup a lungo termine, si consiglia di esportare i dati delle snapshot in Amazon S3. Se la versione principale del motore DB non è più supportata, non è possibile ripristinare tale versione da un'istantanea. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Puoi creare uno snapshot del cluster DB utilizzando AWS Management Console AWS CLI, the o. RDS API

Console

Per creare uno snapshot del cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).

   Viene visualizzato l'elenco Snapshot manuali.
3. Seleziona Acquisisci snapshot.

   Viene visualizzata la finestra Acquisizione di snapshot DB.

4. Per il tipo di snapshot, seleziona DB cluster.
5. Scegli il cluster DB per il quale desideri scattare un'istantanea.
6. Immettete il nome dello snapshot.
7. Seleziona Acquisisci snapshot.

   Viene visualizzato l'elenco delle istantanee manuali, con lo stato della nuova istantanea del cluster DB visualizzato come. `Creating` Dopo che lo stato è diventato `Available`, potrai vedere il tempo di creazione.

AWS CLI

Quando si crea uno snapshot del cluster DB utilizzando il AWS CLI, è necessario identificare il cluster di DB di cui eseguire il backup e quindi assegnare un nome allo snapshot del cluster DB in modo da poterlo ripristinare in un secondo momento. È possibile eseguire questa operazione utilizzando il AWS CLI `create-db-cluster-snapshot`comando con i seguenti parametri:

- `--db-cluster-identifier`
- `--db-cluster-snapshot-identifier`

In questo esempio, si crea uno snapshot del cluster DB denominato *`mydbclustersnapshot`* per un cluster DB chiamato *`mydbcluster`*.

Example

Per LinuxmacOS, oUnix:

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mydbcluster \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Per Windows:

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

RDS API

Quando crei uno snapshot del cluster DB utilizzando Amazon RDSAPI, devi identificare il cluster di DB di cui vuoi eseguire il backup e quindi assegnare un nome allo snapshot del cluster DB in modo da poterlo ripristinare in un secondo momento. Puoi farlo utilizzando il RDS API `CreateDBClusterSnapshot`comando Amazon con i seguenti parametri:

- DBClusterIdentifier
- DBClusterSnapshotIdentifier

Determinare se lo snapshot del cluster di database è disponibile

Puoi verificare che lo snapshot del cluster DB sia disponibile cercando in Snapshot nella scheda Manutenzione e backup nella pagina di dettaglio del cluster in AWS Management Console, utilizzando il `describe-db-cluster-snapshots`CLIcomando o utilizzando l'azione. `DescribeDBClusterSnapshots`API

È inoltre possibile utilizzare il `wait db-cluster-snapshot-available`CLIcomando per eseguire il polling API ogni 30 secondi fino a quando lo snapshot non è disponibile.

Ripristino da uno snapshot cluster database

Amazon RDS crea una snapshot dei volumi di storage del cluster di database eseguendo il backup dell'intero cluster anziché dei singoli database. È possibile creare un nuovo cluster di database eseguendo il ripristino da uno snapshot di database. Devi fornire il nome dello snapshot del cluster di database da cui eseguire il ripristino e quindi un nome per il nuovo cluster di database che viene creato dal ripristino. Non è possibile eseguire il ripristino da una snapshot di cluster di database in un cluster di database esistente. Al momento del ripristino, viene creato un nuovo cluster di database.

> **Important**
>
> Se si tenta di ripristinare un'istantanea su una versione obsoleta del motore DB, verrà eseguito un aggiornamento immediato alla versione più recente del motore. Un cluster DB ripristinato da una versione obsoleta del motore non sarà accessibile fino a quando non verrà aggiornato a una versione principale più recente.
> Inoltre, potrebbero essere applicati costi di Extended Support se la versione è in Extended Support o ha raggiunto la fine del supporto standard. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con .

È possibile utilizzare l'istanza del cluster di database ripristinato non appena lo stato diventa `available`.

È possibile utilizzare AWS CloudFormation per ripristinare un cluster DB da un'istantanea del cluster DB. Per ulteriori informazioni, consulta AWS: :RDS:: DBCluster nella Guida per l'AWS CloudFormation utente.

> **Note**
>
> La condivisione manuale di un'istantanea del cluster DB, crittografata o non crittografata, consente agli AWS account autorizzati di ripristinare direttamente un cluster DB dalla snapshot invece di prenderne una copia ed eseguire il ripristino da quella. Per ulteriori informazioni, consulta Condivisione di uno snapshot cluster database.

Per informazioni sul ripristino di un cluster Aurora DB o di un cluster globale con una versione RDS Extended Support, vedere. Ripristino di Aurora DB o di un cluster globale con Amazon Extended Support RDS

Considerazioni sui gruppi di parametri

È consigliabile mantenere il gruppo parametri del database e il gruppo di parametri del cluster di database per tutti gli snapshot del cluster di database creati, in modo da poter associare il cluster di database ripristinato ai gruppi di parametri corretti.

Il gruppo parametri del database di default e il gruppo di parametri del cluster di database sono associati al cluster ripristinato, a meno che non si scelgano gruppi diversi. Non sono disponibili impostazioni personalizzate per i parametri nei gruppi di parametri di default.

È possibile specificare i gruppi di parametri al momento del ripristino del cluster di database.

Per ulteriori informazioni sui gruppi di parametri del database e i gruppi di parametri del cluster di database, consulta .

Considerazioni relative al gruppo di sicurezza

Quando ripristini un cluster di database, il cloud privato virtuale (VPC) di default, il gruppo di sottoreti di database e il gruppo di sicurezza VPC sono associati all'istanza ripristinata, a meno che non si scelgano altri gruppi.

- Se utilizzi la console Amazon RDS, puoi specificare un gruppo di sicurezza VPC personalizzato da associare al cluster o creare un nuovo gruppo di sicurezza VPC.
- Se utilizzi il AWS CLI, puoi specificare un gruppo di sicurezza VPC personalizzato da associare al cluster includendo l'`--vpc-security-group-id`opzione nel comando. `restore-db-cluster-from-snapshot`
- Se utilizzi l'API di Amazon RDS, puoi includere il parametro `VpcSecurityGroupIds.VpcSecurityGroupId.N` nell'operazione `RestoreDBClusterFromSnapshot`.

Non appena il ripristino è completo e il nuovo cluster di database è disponibile, puoi anche cambiare le impostazioni del VPC modificando il cluster di database. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Considerazioni Amazon Aurora

Con Aurora, si ripristina uno snapshot cluster DB in un cluster DB.

Con Aurora MySQL e Aurora PostgreSQL, è anche possibile ripristinare uno snapshot cluster database in un cluster database Aurora Serverless. Per ulteriori informazioni, consulta Ripristino di un Aurora Serverless v1 cluster di database.

Con Aurora MySQL, è anche possibile ripristinare uno snapshot cluster database da un cluster senza query parallela a un cluster con query parallela. Poiché la query parallela viene usata in genere con tabelle di dimensioni molto grandi, il meccanismo di snapshot è il più veloce per inserire volumi elevati di dati in un cluster Aurora MySQL abilitato per la query parallela. Per ulteriori informazioni, consulta Interrogazione parallela per Amazon Aurora My SQL.

Ripristino da uno snapshot

È possibile ripristinare un cluster di database da uno snapshot del cluster di database tramite la AWS Management Console, la AWS CLI o l'API RDS.

Console

Per ripristinare un cluster database da una snapshot cluster DB

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere lo snapshot cluster database da cui eseguire il ripristino.
4. Per Actions (Operazioni), selezionare Restore Snapshot (Ripristina snapshot).

   Viene visualizzata la pagina Ripristina snapshot.
5. Scegli la versione del motore del database in cui desideri ripristinare il cluster database.

   Per impostazione predefinita, lo snapshot viene ripristinato nella stessa versione del motore del database del cluster database di origine, se tale versione è disponibile.
6. Per DB Instance Identifier (Identificatore di istanza database), immetti il nome dell'istanza database ripristinata. Tieni presente che Amazon RDS ricava l'identificatore del cluster DB dall'identificatore dell'istanza DB specificato.
7. Specificare altre impostazioni, ad esempio la configurazione dell'archiviazione del cluster database.

   Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

8. Scegliere Restore DB Cluster (Ripristina cluster di database).

AWS CLI

Per ripristinare un cluster DB da uno snapshot del cluster DB, usa il comando -snapshot. AWS CLI restore-db-cluster-from

In questo esempio il ripristino avviene da uno snapshot del cluster DB creato precedentemente e denominato `mydbclustersnapshot`. Viene ripristinato un nuovo cluster DB denominato `mynewdbcluster`.

Puoi specificare altre impostazioni, come la versione del motore del database. Se non specifichi una versione del motore, il cluster database viene ripristinato alla versione del motore predefinita.

Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Example

PerLinux, o: macOS Unix

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine aurora-mysql|aurora-postgresql
```

Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewdbcluster ^
    --snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql|aurora-postgresql
```

Dopo il ripristino del cluster di database, devi aggiungere quest'ultimo al gruppo di sicurezza utilizzato dal cluster di database per creare la snapshot cluster di database, se desideri le stesse funzionalità del cluster di database precedente.

> **Important**
>
> Se si utilizza la console per ripristinare un cluster database, Amazon RDS crea automaticamente l'istanza database primaria (istanza di scrittura) per il cluster database.

Se si utilizza AWS CLI per ripristinare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non si crea l'istanza database primaria, lo stato degli endpoint del cluster database rimange impostato su `creating`.
Chiama il create-db-instance AWS CLI comando per creare l'istanza principale per il tuo cluster DB. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

API RDS

Per ripristinare un cluster DB da un'istantanea del cluster DB, richiama l'operazione Restore dell'API RDS DBCluster FromSnapshot con i seguenti parametri:

- `DBClusterIdentifier`
- `SnapshotIdentifier`

Important

Se si utilizza la console per ripristinare un cluster database, Amazon RDS crea automaticamente l'istanza database primaria (istanza di scrittura) per il cluster database.
Se si utilizza API RDS per ripristinare un cluster di database, è necessario creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database. Se non si crea l'istanza database primaria, lo stato degli endpoint del cluster database rimange impostato su `creating`.
Chiama l'operazione Create dell'API RDS DBInstance per creare l'istanza principale per il cluster DB. Includi il nome del cluster di database come valore del parametro `DBClusterIdentifier`.

Copia di snapshot del cluster DB

Con Amazon Aurora è possibile copiare backup automatici o snapshot manuali del cluster di database. Dopo aver copiato uno snapshot, la copia è uno snapshot manuale. È possibile creare più copie di un backup automatico o di uno snapshot manuale, ma ogni copia deve avere un identificatore univoco.

È possibile copiare un'istantanea all'interno della stessa Regione AWS, copiare un'istantanea dall'altra parte Regioni AWS e copiare istantanee condivise.

Non puoi copiare uno snapshot del cluster di database tra regioni e account in una fase singola. Esegui una fase per ognuna di queste operazioni di copia. In alternativa alla copia, puoi anche condividere istantanee manuali con altri account. AWS Per ulteriori informazioni, consulta Condivisione di uno snapshot cluster database.

> **Note**
>
> Amazon ti invia una fattura in base alla quantità dei dati di backup e snapshot Amazon Aurora mantenuti e alla durata di mantenimento. Per informazioni sullo storage associato ai backup e agli snapshot Aurora, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora. Per informazioni sui prezzi relativi allo storage Aurora, consulta prezzi di Amazon RDS for Aurora.

Esamina le limitazioni e le considerazioni relative alla copia degli snapshot del cluster DB. Per copiare le istantanee dei cluster DB, consulta uno dei seguenti argomenti.

- Copiare uno snapshot del cluster DB con AWS Management Console
- Copiare uno snapshot del cluster DB non crittografato utilizzando il AWS CLI o Amazon RDS API
- Copiare uno snapshot crittografato del cluster DB utilizzando AWS CLI o Amazon RDS API
- Copia di una snapshot cluster database tra account

Copiare uno snapshot del cluster DB con AWS Management Console

Utilizza le procedure in questo argomento, per copiare una snapshot cluster database. Se il motore di database di origine è Aurora, lo snapshot è uno snapshot cluster database.

Per ogni AWS account, è possibile copiare fino a cinque istantanee del cluster DB alla volta da uno Regione AWS all'altro. Viene supportata la copia di snapshot cluster database crittografate e non crittografate. Se si copia uno snapshot del cluster DB su un altro Regione AWS, si crea uno snapshot manuale del cluster di DB che viene conservato in quello stesso. Regione AWS La copia di uno snapshot del cluster DB dall'origine comporta costi di Regione AWS trasferimento dati Amazon RDS.

Per ulteriori informazioni sui prezzi del trasferimento dati, consulta Prezzi di Amazon RDS.

Dopo aver creato la copia dello snapshot del cluster DB nel nuovo database Regione AWS, la copia dello snapshot del cluster DB si comporta come tutte le altre istantanee del cluster DB. Regione AWS

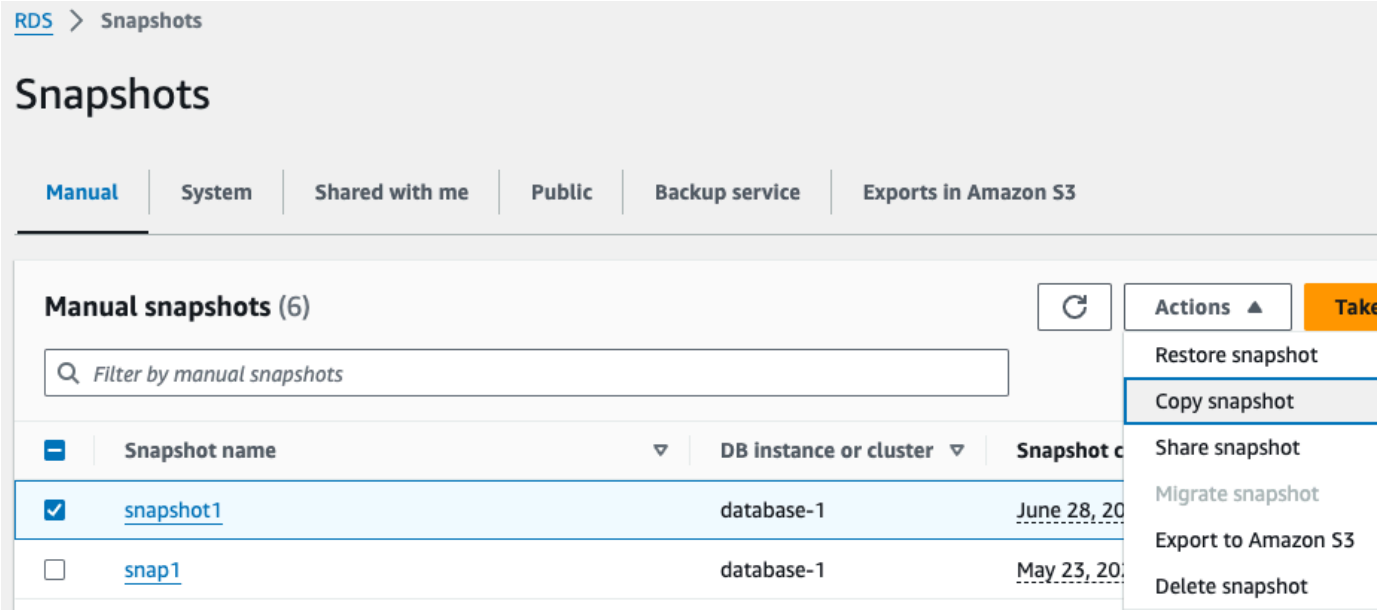
Questa procedura funziona per snapshot del cluster di database crittografati o non crittografati, nella stessa Regione AWS o tra regioni.

Per cancellare un'operazione di copia quando è in corso, elimina la snapshot cluster database target mentre la snapshot cluster database è nello stato copying (copia in corso).

Prima di copiare uno snapshot del cluster DB, esamina e. Limitazioni Considerazioni sulla copia delle istantanee

Per copiare una snapshot cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Snapshots (Snapshot).
3. Seleziona lo snapshot del cluster DB che desideri copiare.
4. Scegli Azioni, quindi scegli Copia istantanea.



5. (Facoltativo) Per copiare l'istantanea del cluster DB in un'altra Regione AWS, selezionala in Regione Regione AWS di destinazione.
6. Digita il nome della copia di snapshot di cluster di database in New DB Snapshot Identifier (Nuovo identificatore snapshot DB).
7. Per copiare i tag e i valori dalla snapshot alla copia della snapshot, seleziona Copy Tags (Copia tag).
8. Seleziona Copy Snapshot (Copia snapshot).

Limitazioni

Di seguito sono riportate alcune limitazioni che si applicano quando si copiano le snapshot:

- Non è possibile copiare uno snapshot da o verso i seguenti siti: Regioni AWS
  - Cina (Pechino)
  - Cina (Ningxia)
- È possibile copiare un'istantanea tra AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali). Tuttavia, non puoi copiare un'istantanea tra queste AWS GovCloud (US) regioni e quelle commerciali. Regioni AWS
- Se elimini una snapshot origine prima che la snapshot target diventi disponibile, la copia della snapshot potrebbe non riuscire. Verifica che la snapshot target abbia lo stato di `AVAILABLE` prima di eliminare una snapshot origine.

- Puoi avere un massimo di cinque richieste di copia di snapshot in corso in una singola regione di destinazione per account
- Quando vengono richieste più copie di snapshot per la stessa istanza database di origine, vengono accodate internamente. Le copie richieste in seguito non verranno avviate fino al completamento delle copie di snapshot precedenti. Per ulteriori informazioni, consulta Perché la creazione di snapshot EC2 AMI o EBS è lenta? nel AWS Knowledge Center.
- A seconda del soggetto Regioni AWS coinvolto e della quantità di dati da copiare, il completamento di una copia istantanea tra diverse regioni può richiedere ore. In alcuni casi, potrebbe esserci un numero elevato di richieste di copia di snapshot tra regioni da una determinata regione di origine. In questi casi, Amazon RDS potrebbe mettere in coda le richieste di copia tra regioni provenienti dalla regione di origine fino al completamento di alcune copie in corso. Nessuna informazione di progresso viene visualizzata sulle richieste di copia mentre sono in coda. Le informazioni sul progresso vengono visualizzate quando inizia la copia.
- Aurora non supporta le istantanee incrementali. Le copie degli snapshot del cluster Aurora DB vengono sempre archiviate come copie complete. La copia di uno snapshot completa contiene tutti i dati e i metadati necessari per archiviare l'istanza database.

Considerazioni sulla copia delle istantanee

Di seguito sono riportate le considerazioni relative alla copia delle istantanee.

Argomenti

- Considerazioni sulla copia condivisa di istantanee
- Considerazioni sulla copia crittografata degli snapshot del cluster DB
- Considerazioni sulla copia di istantanee tra regioni
- Considerazioni per i gruppi di parametri

Considerazioni sulla copia condivisa di istantanee

Puoi copiare istantanee condivise con te da altri account. AWS In alcuni casi, potresti copiare un'istantanea crittografata che è stata condivisa da un altro AWS account. In questi casi, è necessario avere accesso allo AWS KMS key strumento utilizzato per crittografare l'istantanea.

Puoi copiare solo uno snapshot condiviso del cluster di database, crittografato o meno, nella stessa Regione AWS. Per ulteriori informazioni, consulta Condivisione di snapshot crittografati.

Considerazioni sulla copia crittografata degli snapshot del cluster DB

Puoi copiare una snapshot che è stata crittografata utilizzando una chiave KMS. Se la copia di una snapshot crittografata, la copia della snapshot deve anche essere crittografata. Se si copia un'istantanea crittografata all'interno dello stesso Regione AWS, è possibile crittografare la copia con la stessa chiave KMS dell'istantanea originale. Oppure puoi specificare una chiave KMS diversa.

Se copi uno snapshot crittografato tra regioni, devi specificare una chiave KMS valida nella Regione AWS di destinazione. Può essere una chiave KMS specifica per la regione o una chiave multi-regione. Per ulteriori informazioni sulle chiavi multi-regione, consulta Utilizzo delle chiavi multi-regione in AWS KMS.

Per ulteriori informazioni sulla gestione delle AWS KMS chiavi per Amazon RDS, consultaAWS KMS key gestione.

La snapshot di origine resta crittografata nel processo di copia. Per ulteriori informazioni, consulta Limiti relativi a istanze database crittografate Amazon Aurora.

> **Note**
>
> Per le snapshot di cluster di database Amazon Aurora, non puoi crittografare una snapshot di cluster di database quando copi la snapshot.

Per copiare istantanee crittografate dei cluster DB, consulta i seguenti argomenti.

- Copiare uno snapshot crittografato del cluster DB utilizzando AWS CLI o Amazon RDS API
- Copia di una snapshot cluster database tra account

Considerazioni sulla copia di istantanee tra regioni

È possibile copiare snapshot di cluster di database tra Regioni AWS. Tuttavia, esistono alcuni vincoli e considerazioni per la copia di snapshot tra le regioni.

A seconda del soggetto Regioni AWS coinvolto e della quantità di dati da copiare, il completamento di una copia istantanea tra aree geografiche può richiedere ore.

In alcuni casi, potrebbe esserci un numero elevato di richieste di copia di snapshot tra regioni da una determinata Regione AWS di origine. In questi casi, Amazon RDS potrebbe mettere in coda

nuove richieste di copia interregionali provenienti da tale fonte fino al completamento di alcune copie Regione AWS in corso. Nessuna informazione di progresso viene visualizzata sulle richieste di copia mentre sono in coda. Le informazioni sul progresso vengono visualizzate quando inizia la copia.

La copia di istantanee tra regioni crea copie complete nei dati di destinazione, ma i costi di trasferimento dei dati sono incrementali. I dati incrementali includono sia i nuovi dati che sono stati aggiunti al database di un cliente dopo l'ultima copia, sia tutte le modifiche apportate ai dati esistenti. Per ulteriori informazioni, consulta Creazione di copie di backup Regioni AWS nella AWS Backup Developer Guide.

> **Note**
>
> Aurora copia la quantità minima di dati richiesta per creare una copia completa di un'istantanea nella regione di destinazione. Le tariffe per il trasferimento dei dati si applicano quando si copiano istantanee tra regioni.

Considerazioni per i gruppi di parametri

Quando copi uno snapshot tra regioni, la copia non include il gruppo di parametri usato dal cluster di database originale. Quando si ripristina un'istantanea per creare un nuovo cluster DB, a tale cluster di database viene assegnato il gruppo di parametri predefinito in cui Regione AWS è stato creato. Per assegnare al nuovo cluster di database gli stessi parametri dell'originale, completa la seguente procedura:

1. Nella destinazione Regione AWS, crea un gruppo di parametri del cluster DB con le stesse impostazioni del cluster DB originale. Se ne esiste già uno nel nuovo Regione AWS, puoi usare quello.
2. Dopo aver ripristinato l'istantanea nella destinazione Regione AWS, modifica il nuovo cluster DB e aggiungi il gruppo di parametri nuovo o esistente del passaggio precedente.

Copiare uno snapshot del cluster DB non crittografato utilizzando il AWS CLI o Amazon RDS API

Utilizza le procedure descritte nelle sezioni seguenti per copiare uno snapshot del cluster DB non crittografato utilizzando il AWS Management Console, AWS CLI o Amazon RDSAPI.

Per cancellare un'operazione di copia quando è in corso, elimina lo snapshot del cluster di database target identificato da `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mentre lo stato dello snapshot è Copia in corso.

Console

Per copiare uno snapshot del cluster DB utilizzando il AWS Management Console, consulta Copiare uno snapshot del cluster DB con AWS Management Console.

AWS CLI

Per copiare uno snapshot del cluster DB, utilizzare il AWS CLI copy-db-cluster-snapshotcomando. Se stai copiando l'istantanea su un'altra Regione AWS, esegui il comando in Regione AWS in cui verrà copiata l'istantanea.

Le seguenti opzioni vengono utilizzate per copiare una snapshot cluster database non crittografata:

- `--source-db-cluster-snapshot-identifier` – Identificatore per lo snapshot cluster database da copiare. Se stai copiando l'istantanea su un'altra Regione AWS, questo identificatore deve essere nel ARN formato dell'origine Regione AWS.
- `--target-db-cluster-snapshot-identifier` – Identificatore per la nuova copia dello snapshot cluster database.

Il codice seguente crea una copia dello snapshot del cluster DB `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` denominato `myclustersnapshotcopy` in Regione AWS in cui viene eseguito il comando. Quando viene creata la copia, tutti i tag della snapshot originale vengono copiati nella copia della snapshot.

Example

In Linux, macOS, oppure Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --copy-tags
```

In Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --copy-tags
```

RDS API

Per copiare uno snapshot del cluster DB, utilizza l'operazione Amazon RDS API C opyDBCluster Snapshot. Se stai copiando lo snapshot su un altro Regione AWS, eseguire l'azione nella Regione AWS in cui verrà copiata l'istantanea.

I seguenti parametri vengono utilizzati per copiare una snapshot cluster database non crittografata:

- `SourceDBClusterSnapshotIdentifier` – Identificatore per lo snapshot cluster database da copiare. Se stai copiando l'istantanea su un'altra Regione AWS, questo identificatore deve essere nel ARN formato dell'origine Regione AWS.
- `TargetDBClusterSnapshotIdentifier` – Identificatore per la nuova copia dello snapshot cluster database.

Il seguente codice crea una copia di uno snapshot `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` denominata `myclustersnapshotcopy` nella regione Stati Uniti occidentali (California settentrionale). Quando viene creata la copia, tutti i tag della snapshot originale vengono copiati nella copia della snapshot.

Example

```
https://rds.us-west-1.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20130805
   &TargetDBSnapshotIdentifier=myclustersnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
```

```
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copiare uno snapshot crittografato del cluster DB utilizzando AWS CLI o Amazon RDS API

Utilizza le procedure illustrate nelle seguenti sezioni per copiare uno snapshot crittografato del cluster DB utilizzando AWS Management Console, AWS CLI o Amazon RDSAPI.

Per cancellare un'operazione di copia quando è in corso, elimina lo snapshot del cluster di database target identificato da `--target-db-cluster-snapshot-identifier` o `TargetDBClusterSnapshotIdentifier` mentre lo stato dello snapshot è Copia in corso.

Console

Per copiare uno snapshot del cluster DB utilizzando AWS Management Console, consulta Copiare uno snapshot del cluster DB con AWS Management Console.

AWS CLI

Per copiare uno snapshot del cluster DB, utilizzare il AWS CLI copy-db-cluster-snapshotcomando. Se stai copiando l'istantanea su un'altra Regione AWS, esegui il comando in Regione AWS in cui verrà copiata l'istantanea.

Le seguenti opzioni vengono utilizzate per copiare una snapshot cluster database crittografata:

- `--source-db-cluster-snapshot-identifier` – Identificatore per lo snapshot cluster database crittografato da copiare. Se stai copiando l'istantanea su un'altra Regione AWS, questo identificatore deve essere nel ARN formato dell'origine Regione AWS.
- `--target-db-cluster-snapshot-identifier` – Identificatore per la nuova copia dello snapshot cluster database crittografato.
- `--kms-key-id`— L'identificatore della KMS chiave da utilizzare per crittografare la copia dello snapshot del cluster DB.

  Facoltativamente, è possibile utilizzare questa opzione se lo snapshot del cluster DB è crittografato, si copia l'istantanea nello stesso Regione AWS e si desidera specificare una nuova KMS chiave per crittografare la copia. In caso contrario, la copia dello snapshot del cluster DB viene crittografata con la stessa KMS chiave dello snapshot del cluster DB di origine.

È necessario utilizzare questa opzione se lo snapshot del cluster DB è crittografato e si sta copiando lo snapshot su un altro Regione AWS. In tal caso, è necessario specificare una KMS chiave per la destinazione Regione AWS.

Il seguente esempio di codice copia lo snapshot del cluster DB crittografato dalla regione Stati Uniti occidentali (Oregon) alla regione US East (N. Virginia). Il comando viene chiamato nella regione US East (N. Virginia).

Example

In Linux, macOS, oppure Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --kms-key-id my-us-east-1-key
```

In Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --kms-key-id my-us-east-1-key
```

Il `--source-region` parametro è obbligatorio quando si copia uno snapshot crittografato del cluster DB tra AWS GovCloud (Stati Uniti orientali) e AWS GovCloud Regioni (Stati Uniti occidentali). Per`--source-region`, specificare Regione AWS dell'istanza DB di origine. Il Regione AWS specificato in `source-db-cluster-snapshot-identifier` deve corrispondere al Regione AWS specificato per`--source-region`.

Se non si specifica `--source-region`, è necessario specificare un valore per `--pre-signed-url`. Un prefirmato URL è un file URL che contiene una richiesta firmata Signature Version 4 per il `copy-db-cluster-snapshot` comando richiamato nel codice sorgente Regione AWS. Per ulteriori informazioni sull'`pre-signed-url`opzione, copy-db-cluster-snapshotconsulta AWS CLI Riferimento ai comandi.

RDS API

Per copiare uno snapshot del cluster DB, utilizza l'operazione Amazon RDS API C opyDBCluster Snapshot. Se stai copiando lo snapshot su un altro Regione AWS, eseguire l'azione nella Regione AWS in cui verrà copiata l'istantanea.

I seguenti parametri vengono utilizzati per copiare una snapshot cluster database crittografata:

- `SourceDBClusterSnapshotIdentifier` – Identificatore per lo snapshot cluster database crittografato da copiare. Se stai copiando l'istantanea su un'altra Regione AWS, questo identificatore deve essere nel ARN formato dell'origine Regione AWS.
- `TargetDBClusterSnapshotIdentifier` – Identificatore per la nuova copia dello snapshot cluster database crittografato.
- `KmsKeyId`— L'identificatore della KMS chiave da utilizzare per crittografare la copia dello snapshot del cluster DB.

  Facoltativamente, è possibile utilizzare questo parametro se lo snapshot del cluster DB è crittografato, si copia l'istantanea nello stesso Regione AWS e si specifica una nuova KMS chiave da utilizzare per crittografare la copia. In caso contrario, la copia dello snapshot del cluster DB viene crittografata con la stessa KMS chiave dello snapshot del cluster DB di origine.

  È necessario utilizzare questo parametro se lo snapshot del cluster DB è crittografato e si sta copiando l'istantanea su un altro Regione AWS. In tal caso, è necessario specificare una KMS chiave per la destinazione Regione AWS.
- `PreSignedUrl`— Se state copiando l'istantanea su un'altra Regione AWS, è necessario specificare il parametro. `PreSignedUrl` Il `PreSignedUrl` valore deve essere un messaggio URL che contenga una richiesta firmata Signature Version 4 affinché l'`CopyDBClusterSnapshot`azione venga richiamata nell'origine Regione AWS da dove viene copiata l'istantanea del cluster DB. Per ulteriori informazioni sull'utilizzo di un predefinitoURL, consulta C Snapshot. opyDBCluster

Il seguente esempio di codice copia lo snapshot del cluster DB crittografato dalla regione Stati Uniti occidentali (Oregon) alla regione US East (N. Virginia). L'operazione viene chiamata nella regione US East (N. Virginia).

Example

```
https://rds.us-east-1.amazonaws.com/
```

```
    ?Action=CopyDBClusterSnapshot
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBClusterSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBClusterSnapshotIdentifier%253Darn%25253Aaws%25253Ards
%25253Aus-west-2%25253A123456789012%25253Acluster-snapshot%25253Aaurora-cluster1-
snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBClusterSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20161115
    &TargetDBClusterSnapshotIdentifier=myclustersnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

Il `PreSignedUrl` parametro è obbligatorio quando si copia un'istantanea crittografata del cluster DB tra AWS GovCloud (Stati Uniti orientali) e AWS GovCloud Regioni (Stati Uniti occidentali). Il `PreSignedUrl` valore deve essere un messaggio URL che contenga una richiesta firmata Signature Version 4 affinché l'`CopyDBClusterSnapshot`operazione venga richiamata nell'origine Regione AWS da dove viene copiata l'istantanea del cluster DB. Per ulteriori informazioni sull'utilizzo di un predefinitoURL, consulta C opyDBCluster Snapshot in Amazon RDS API Reference.

Per generare automaticamente anziché manualmente un predefinito, usa il URL AWS CLI copy-db-cluster-snapshotcomando con l'`--source-region`opzione invece.

Copia di una snapshot cluster database tra account

Puoi abilitare altri AWS account per copiare gli snapshot del cluster DB specificati utilizzando Amazon RDS API `ModifyDBClusterSnapshotAttribute` and `CopyDBClusterSnapshot` actions. Puoi copiare le istantanee dei cluster DB solo tra account dello stesso Regione AWS. Il processo di copia tra account funziona nel modo seguente, in cui l'Account A rende l'istantanea disponibile per la copia e l'Account B la copia.

1. Usando l'account A, chiama `ModifyDBClusterSnapshotAttribute`, specificando **`restore`** per il parametro `AttributeName` e l'ID per l'account B per il parametro `ValuesToAdd`.
2. (Se l'istantanea è crittografata) Utilizzando l'account A, aggiorna la politica della chiave per la KMS chiave, aggiungendo prima l'ARNaccount B come a `Principal` e quindi consenti l'azione. `kms:CreateGrant`
3. (Se l'istantanea è crittografata) Utilizzando l'account B, scegli o crea un utente e allega a quell'utente una IAM policy che gli consenta di copiare un'istantanea del cluster DB crittografato utilizzando la tua chiave. KMS
4. Utilizzando l'Account B, chiamate `CopyDBClusterSnapshot` e utilizzate il `SourceDBClusterSnapshotIdentifier` parametro per specificare lo snapshot ARN del cluster DB da copiare, che deve includere l'ID per l'Account A.

Per elencare tutte le AWS gli account autorizzati a ripristinare un'istantanea del cluster DB, utilizzano gli escribeDBSnapshotattributi D o escribeDBCluster SnapshotAttributes API l'operazione D.

Per rimuovere l'autorizzazione di condivisione per un AWS account, utilizza l'`ModifyDBClusterSnapshotAttribute`azione `ModifyDBSnapshotAttribute` o con `AttributeName` set to `restore` e l'ID dell'account da rimuovere nel `ValuesToRemove` parametro.

Copia di una snapshot cluster database non crittografata in un altro account

Utilizzare la seguente procedura per copiare un'istantanea del cluster DB non crittografata su un altro account nello stesso Regione AWS.

1. Nell'account di origine per lo snapshot cluster database chiama `ModifyDBClusterSnapshotAttribute`, specificando **`restore`** per il parametro `AttributeName` e l'ID per l'account di destinazione per il parametro `ValuesToAdd`.

L'esecuzione del seguente esempio utilizzando l'account 987654321 ne consente due AWS identificatori di account 123451234512 e123456789012, per ripristinare lo snapshot del cluster DB denominato. manual-snapshot1

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. Nell'account di destinazione, chiamate CopyDBClusterSnapshot e utilizzate il SourceDBClusterSnapshotIdentifier parametro per specificare lo snapshot ARN del cluster DB da copiare, che deve includere l'ID dell'account di origine.

   L'esecuzione dell'esempio seguente utilizzando l'account 123451234512 copia la snapshot cluster database aurora-cluster1-snapshot-20130805 dall'account 987654321 e crea una snapshot cluster database denominata dbclustersnapshot1.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
```

```
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copia di una snapshot cluster database crittografata in un altro account

Utilizzare la procedura seguente per copiare uno snapshot crittografato del cluster DB su un altro account nello stesso account Regione AWS.

1. Nell'account di origine per lo snapshot cluster database chiama `ModifyDBClusterSnapshotAttribute`, specificando **`restore`** per il parametro `AttributeName` e l'ID per l'account di destinazione per il parametro `ValuesToAdd`.

   L'esecuzione del seguente esempio utilizzando l'account ne `987654321` consente due AWS identificatori di account `123451234512` e`123456789012`, per ripristinare lo snapshot del cluster DB denominato. `manual-snapshot1`

   ```
   https://rds.us-west-2.amazonaws.com/
    ?Action=ModifyDBClusterSnapshotAttribute
    &AttributeName=restore
    &DBClusterSnapshotIdentifier=manual-snapshot1
    &SignatureMethod=HmacSHA256&SignatureVersion=4
    &ValuesToAdd.member.1=123451234512
    &ValuesToAdd.member.2=123456789012
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
    &X-Amz-Date=20150922T220515Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
   ```

2. Nell'account di origine per lo snapshot del cluster DB, crea una KMS chiave personalizzata nello stesso Regione AWS come istantanea crittografata del cluster DB. Durante la creazione della chiave gestita dal cliente, concedi l'accesso ad essa per la destinazione Account AWS. Per ulteriori informazioni, vedereCrea una chiave gestita dal cliente e consenti l'accesso ad essa.
3. Copia e condividi l'istantanea sulla destinazione Account AWS. Per ulteriori informazioni, vedereCopia e condividi l'istantanea dall'account di origine.
4. Nell'account di destinazione, chiamate `CopyDBClusterSnapshot` e utilizzate il `SourceDBClusterSnapshotIdentifier` parametro per specificare lo snapshot ARN del cluster DB da copiare, che deve includere l'ID dell'account di origine.

L'esecuzione dell'esempio seguente utilizzando l'account 123451234512 copia la snapshot cluster database aurora-cluster1-snapshot-20130805 dall'account 987654321 e crea una snapshot cluster database denominata dbclustersnapshot1.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Condivisione di uno snapshot cluster database

Utilizzando AmazonRDS, puoi condividere uno snapshot manuale del cluster DB nei seguenti modi:

- La condivisione di uno snapshot del cluster di database manuale, crittografato o non crittografato permette agli account AWS autorizzati di copiare lo snapshot.
- La condivisione di uno snapshot cluster database manuale, crittografato o non crittografato, permette agli account AWS autorizzati di ripristinare direttamente un cluster database dallo snapshot, anziché eseguirne una copia da cui effettuare il ripristino.

> **Note**
>
> Per condividere uno snapshot cluster database automatico, crea uno snapshot cluster database manuale copiando lo snapshot automatico e quindi condividi la copia. Questo processo si applica anche alle risorse generate da AWS Backup.

Per ulteriori informazioni sulla creazione di una copia di una snapshot, consulta Copia di snapshot del cluster DB. Per ulteriori informazioni sul ripristino di un'istanza database da uno snapshot del cluster di database, consulta Ripristino da uno snapshot cluster database.

Per ulteriori informazioni sul ripristino di un cluster DB da una snapshot cluster DB, consulta Panoramica di backup e ripristino di un cluster di database Aurora.

Puoi condividere uno snapshot manuale con un massimo di 20 altri. Account AWS

La seguente limitazione si applica quando si condividono istantanee manuali con altri: Account AWS

- Quando ripristini un cluster DB da uno snapshot condiviso utilizzando AWS Command Line Interface (AWS CLI) o Amazon RDSAPI, devi specificare l'Amazon Resource Name (ARN) dello snapshot condiviso come identificatore dello snapshot.

Scopri come condividere istantanee, istantanee pubbliche e istantanee crittografate nelle seguenti sezioni. Puoi anche scoprire come interrompere la condivisione di istantanee.

Argomenti

- Condivisione di uno snapshot

- Condivisione di snapshot pubblici
- Condivisione di snapshot crittografati
- Interruzione della condivisione delle istantanee

Condivisione di uno snapshot

Puoi condividere uno snapshot del cluster DB utilizzando il AWS Management Console, il AWS CLI, o il. RDS API

Console

Utilizzando la RDS console Amazon, puoi condividere uno snapshot manuale del cluster DB con un massimo di 20 Account AWS persone. Puoi anche utilizzare la console per interrompere la condivisione di una snapshot manuale con uno o più account.

Per condividere uno snapshot manuale del cluster DB utilizzando la console Amazon RDS

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Selezionare lo snapshot manuale da condividere.
4. Per Actions (Operazioni), seleziona Share snapshot (Condividi snapshot).
5. Scegliere una delle opzioni seguenti per DB snapshot visibility (Visibilità snapshot DB).

    - Se l'origine non è crittografata, scegli Pubblica per consentire Account AWS a tutti di ripristinare un cluster di DB dall'istantanea manuale del cluster di database, oppure scegli Privato per consentire solo a Account AWS ciò che hai specificato di ripristinare un cluster di DB dallo snapshot manuale del cluster DB.

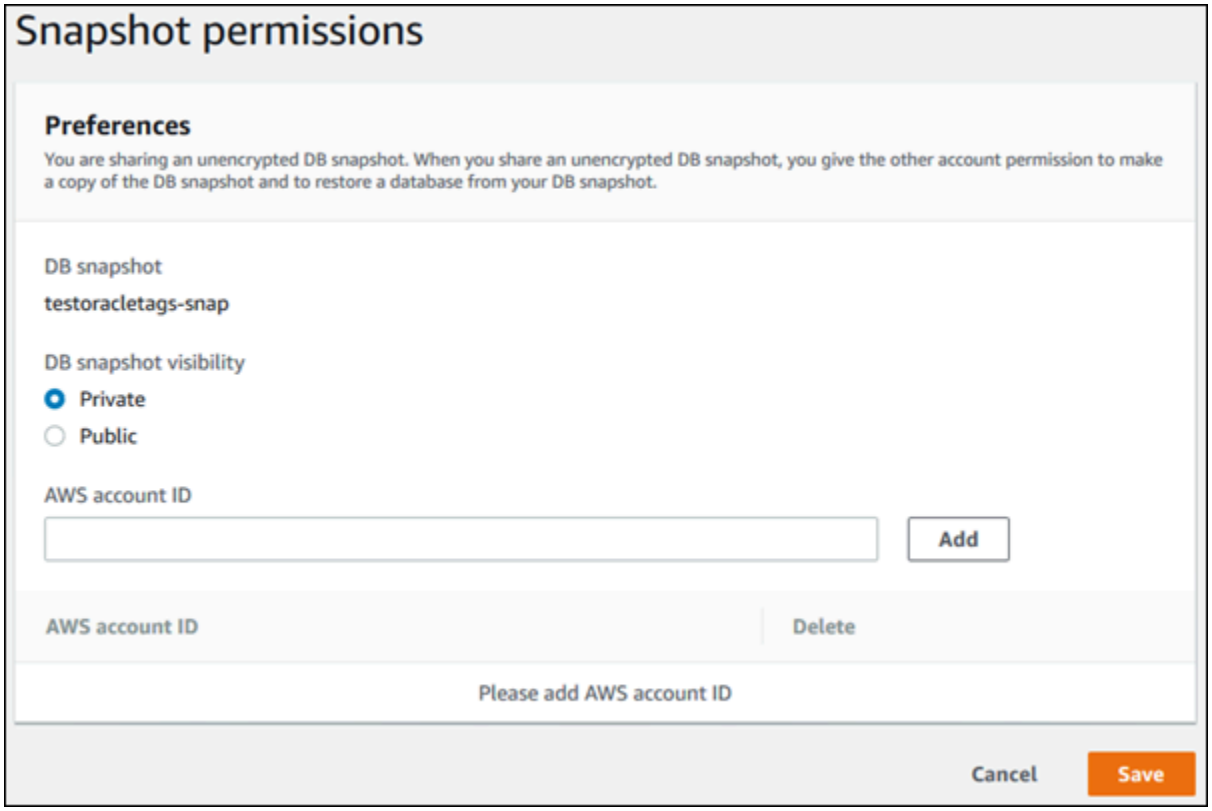
    > **Warning**
    >
    > Se imposti la visibilità dello snapshot DB su Pubblico, tutti Account AWS possono ripristinare un cluster DB dallo snapshot manuale del cluster DB e avere accesso ai tuoi dati. Non condividere snapshot di cluster di database manuali che contengono informazioni private impostandoli come Public (Pubblico).
    > Per ulteriori informazioni, consulta Condivisione di snapshot pubblici.

- Se l'origine è crittografata, l'opzione DB snapshot visibility (Visibilità snapshot DB) è impostata su Private (Privato), perché gli snapshot crittografati non possono essere condivisi come pubblici.

> **Note**
>
> Le istantanee che sono state crittografate con l'impostazione predefinita non AWS KMS key possono essere condivise. Per informazioni su come risolvere questo problema, consultaCondivisione di snapshot crittografati.

6. Per AWS Account ID, inserisci l' Account AWS identificatore di un account a cui desideri consentire il ripristino di un cluster DB dallo snapshot manuale, quindi scegli Aggiungi. Ripeti l'operazione per includere Account AWS identificatori aggiuntivi, fino a 20. Account AWS

   Se commetti un errore durante l'aggiunta di un Account AWS identificatore all'elenco degli account consentiti, puoi eliminarlo dall'elenco scegliendo Elimina a destra dell'identificatore errato Account AWS .



7. Dopo aver aggiunto gli identificatori per tutti gli elementi a Account AWS cui desideri consentire il ripristino dell'istantanea manuale, scegli Salva per salvare le modifiche.

AWS CLI

Per condividere uno snapshot del cluster di database, usa il comando `aws rds modify-db-cluster-snapshot-attribute`. Utilizzate il `--values-to-add` parametro per aggiungere un elenco dei IDs file autorizzati Account AWS a ripristinare l'istantanea manuale.

Example di condividere uno snapshot con un singolo account

L'esempio seguente abilita l' Account AWS identificatore `123456789012` per ripristinare lo snapshot del cluster DB denominato. `cluster-3-snapshot`

PerLinux, omacOS: Unix

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier cluster-3-snapshot \
--attribute-name restore \
--values-to-add 123456789012
```

Per Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier cluster-3-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example di condividere uno snapshot con più account

L'esempio seguente abilita due Account AWS identificatori `111122223333` e`444455556666`, per ripristinare lo snapshot del cluster DB denominato. `manual-cluster-snapshot1`

PerLinux, omacOS: Unix

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

Per Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
```

```
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Quando si utilizza il prompt dei comandi di Windows, è necessario evitare le virgolette doppie («) nel JSON codice anteponendole una barra rovesciata (\).

Per elencare gli utenti Account AWS abilitati al ripristino di un'istantanea, usa il comando. `describe-db-cluster-snapshot-attributes` AWS CLI

RDS API

Puoi anche condividere uno snapshot manuale del cluster DB con altri utenti Account AWS utilizzando Amazon RDSAPI. Per eseguire questa operazione, chiama l'operazione `ModifyDBClusterSnapshotAttribute`. `AttributeName`Specificate `restore` e utilizzate il `ValuesToAdd` parametro per aggiungere un elenco degli IDs utenti autorizzati a ripristinare lo snapshot manuale. Account AWS

Per rendere pubblica e ripristinabile da tutti un'istantanea manuale Account AWS, usa il valore. `all` Tuttavia, fai attenzione a non aggiungere `all` valore alle istantanee manuali che contengono informazioni private che non desideri siano disponibili per tutti. Account AWS Inoltre, non è necessario specificare `all` per le snapshot crittografate, perché l'operazione di rendere pubbliche queste snapshot non è supportata.

Per elencare tutte le istantanee Account AWS consentite per ripristinare un'istantanea, utilizzate l'`DescribeDBClusterSnapshotAttributes`APIoperazione.

Condivisione di snapshot pubblici

È possibile condividere un'istantanea manuale non crittografata come pubblica, in modo da renderla disponibile a tutti. Account AWS Asicurati, quando condividi uno snapshot come pubblico, che non siano incluse nessuna delle tue informazioni personali.

Quando un'istantanea viene condivisa pubblicamente, concede a tutti i Account AWS permessi sia per copiarla che per creare cluster DB a partire da essa.

Non ti viene addebitata l'archiviazione di backup degli snapshot pubblici di proprietà di altri account. Ti verranno addebitate solo gli snapshot di tua proprietà.

Se si copia uno snapshot pubblico, si è proprietari della copia. Viene addebitato l'archiviazione di backup dello snapshot istantaneo. Se si crea un cluster di database da uno snapshot pubblico, ti viene addebitato di tale cluster di database. Per informazioni sui prezzi di Amazon Aurora, consulta la pagina dei prezzi di Aurora.

È possibile eliminare solo gli snapshot pubblici di tua proprietà. Per eliminare un'istantanea condivisa o pubblica, assicurati di accedere alla persona proprietaria dell' Account AWS istantanea.

Visualizzazione di istantanee pubbliche di proprietà di altri Account AWS

Puoi visualizzare istantanee pubbliche di proprietà di altri account in un particolare Regione AWS nella scheda Pubblico della pagina Snapshot nella console AmazonRDS. Gli snapshot (quelli di proprietà del tuo account) non vengono visualizzati in questa scheda.

Per visualizzare gli snapshot pubblici

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere la scheda Public (Pubblico).

   Vengono visualizzati gli snapshot pubblici. È possibile vedere quale account possiede uno snapshot pubblico nella colonna Owner (Proprietario).

   > **Note**
   >
   > Potrebbe essere necessario modificare le preferenze della pagina, selezionando l'icona a forma di ingranaggio in alto a destra dell'elenco Public snapshots (Snapshot pubblici), per visualizzare questa colonna.

Visualizzazione degli snapshot pubblici

Puoi usare il seguente AWS CLI comando (solo Unix) per visualizzare le istantanee pubbliche di proprietà del tuo Account AWS computer. Regione AWS

```
aws rds describe-db-cluster-snapshots --snapshot-type public --include-public |
 grep account_number
```

Se si dispone di snapshot pubblici, l'output restituito è simile all'esempio seguente.

```
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot1",
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot2",
```

Condivisione di istantanee pubbliche da versioni obsolete del motore DB

Il ripristino o la copia di istantanee pubbliche da versioni obsolete del motore DB non è supportato. Per rendere disponibile lo snapshot pubblico esistente non supportato per il ripristino o la copia, procedi nel seguente modo:

1. Contrassegna l'istantanea come privata.
2. Ripristinare lo snapshot:
3. Aggiorna il cluster DB ripristinato a una versione del motore supportata.
4. Crea una nuova istantanea.
5. Condividi nuovamente l'istantanea pubblicamente.

Condivisione di snapshot crittografati

È possibile condividere istantanee del cluster DB che sono state crittografate «a riposo» utilizzando l'algoritmo di crittografia AES -256, come descritto in. Crittografia delle risorse Amazon Aurora

Per la condivisione di snapshot crittografati vigono le seguenti restrizioni:

- Non puoi condividere le snapshot crittografate come pubbliche.
- Non è possibile condividere un'istantanea che è stata crittografata utilizzando la chiave predefinita KMS di Account AWS che ha condiviso l'istantanea.

  Per ulteriori informazioni sull' AWS KMS gestione delle chiavi per AmazonRDS, vediAWS KMS key gestione.

Per risolvere il problema relativo alla KMS chiave predefinita, esegui le seguenti attività:

1. Crea una chiave gestita dal cliente e consenti l'accesso ad essa.
2. Copia e condividi l'istantanea dall'account di origine.
3. Copia l'istantanea condivisa nell'account di destinazione.

Crea una chiave gestita dal cliente e consenti l'accesso ad essa

Per prima cosa crei una KMS chiave personalizzata nella stessa Regione AWS come istantanea crittografata del cluster DB. Durante la creazione della chiave gestita dal cliente, concedi l'accesso ad essa per un'altra Account AWS.

Per creare una chiave gestita dal cliente e consentirne l'accesso

1. Accedi al AWS Management Console dalla fonte Account AWS.
2. Aprire il AWS KMS console in https://console.aws.amazon.com/kms.
3. Per modificare il Regione AWS, usa il selettore della regione nell'angolo superiore destro della pagina.
4. Nel riquadro di navigazione, scegli Chiavi gestite dal cliente.
5. Scegliere Create key (Crea chiave).
6. Nella pagina Configura chiave:
    a. Per Tipo di chiave, seleziona Symmetric.
    b. Per Utilizzo della chiave, seleziona Crittografa e decrittografa.
    c. Espandere Advanced options (Opzioni avanzate).
    d. Per Origine del materiale chiave, seleziona. KMS
    e. Per Regionalità, seleziona Chiave per regione singola.
    f. Scegli Next (Successivo).
7. Nella pagina Aggiungi etichette:
    a. Per Alias, inserisci un nome visualizzato per la tua KMS chiave, ad esempio**`share-snapshot`**.
    b. (Facoltativo) Inserisci una descrizione per la tua KMS chiave.
    c. (Facoltativo) Aggiungi tag alla tua KMS chiave.
    d. Scegli Next (Successivo).
8. Nella pagina Definisci le autorizzazioni per gestire la chiave scegli Avanti.
9. Nella pagina Definisci le autorizzazioni di utilizzo delle chiavi:
    a. Per Altro Account AWS, scegli Aggiungi un altro Account AWS.
    b. Inserisci l'ID del Account AWS a cui vuoi dare accesso.

Puoi dare accesso a più Account AWS.

c. Scegli Next (Successivo).

10. Controlla la KMS chiave, quindi scegli Fine.

Copia e condividi l'istantanea dall'account di origine

Successivamente si copia lo snapshot del cluster DB di origine in una nuova istantanea utilizzando la chiave gestita dal cliente. Quindi lo condividi con il destinatario Account AWS.

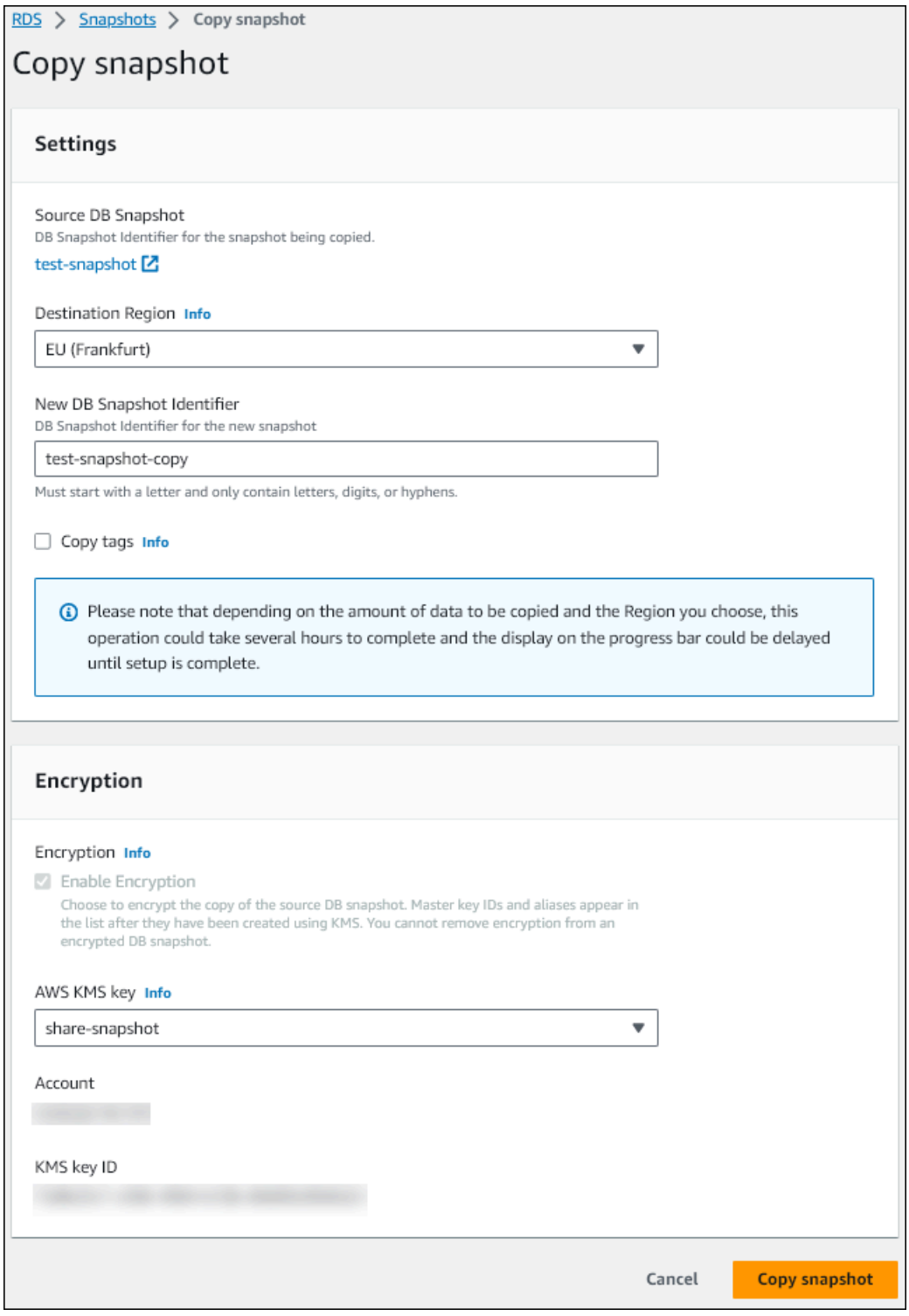
Per copiare e condividere l'istantanea

1. Accedere a AWS Management Console dalla fonte Account AWS.
2. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/
3. Nel pannello di navigazione, selezionare Snapshots (Snapshot).
4. Seleziona lo snapshot del cluster DB che desideri copiare.
5. In Operazioni, seleziona Copia snapshot.
6. Nella pagina Copia istantanea:

   a. Per Regione di destinazione, scegli Regione AWS dove hai creato la chiave gestita dal cliente nella procedura precedente.

   b. Digita il nome della copia di snapshot di cluster di database in New DB Snapshot Identifier (Nuovo identificatore snapshot DB).

   c. Per AWS KMS key, scegli la chiave gestita dal cliente che hai creato.



d. Selezionare Copy Snapshot (Copia snapshot).

7. Quando la copia dell'istantanea è disponibile, selezionala.
8. Per Actions (Operazioni), seleziona Share snapshot (Condividi snapshot).
9. Nella pagina delle autorizzazioni per lo snapshot:

a. Inserisci il Account AWS ID con cui condividi la copia dell'istantanea, quindi scegli Aggiungi.
b. Seleziona Salva.

L'istantanea è condivisa.

Copia l'istantanea condivisa nell'account di destinazione

Ora puoi copiare l'istantanea condivisa nella destinazione Account AWS.

Per copiare l'istantanea condivisa

1. Accedere a AWS Management Console dal bersaglio Account AWS.
2. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/
3. Nel pannello di navigazione, selezionare Snapshots (Snapshot).
4. Scegli la scheda Condivisi con me.
5. Seleziona l'istantanea condivisa.
6. In Operazioni, seleziona Copia snapshot.
7. Scegliete le impostazioni per copiare l'istantanea come nella procedura precedente, ma utilizzate un AWS KMS key che appartiene all'account di destinazione.

   Selezionare Copy Snapshot (Copia snapshot).

Interruzione della condivisione delle istantanee

Per interrompere la condivisione di uno snapshot del cluster DB, rimuovi l'autorizzazione dalla destinazione. Account AWS

Console

Per interrompere la condivisione manuale di uno snapshot del cluster DB con un Account AWS

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Selezionare lo snapshot manuale di cui interrompere la condivisione.

4. Scegli Actions (Operazioni) e quindi Share Snapshot (Condividi snapshot).
5. Per rimuovere l'autorizzazione per un AWS account Account AWS, scegli Elimina come identificatore dell'account dall'elenco degli account autorizzati.
6. Scegliere Salva per salvare le modifiche.

CLI

Per rimuovere un Account AWS identificatore dall'elenco, utilizza il `--values-to-remove` parametro.

Example di interrompere la condivisione degli snapshot

L'esempio seguente impedisce all' Account AWS ID 444455556666 di ripristinare l'istantanea.

PerLinux, o: macOS Unix

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

Per Windows:

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

RDS API

Per rimuovere l'autorizzazione di condivisione per un Account AWS, usa l'`ModifyDBClusterSnapshotAttribute`operazione con `AttributeName` set to `restore` e il `ValuesToRemove` parametro. Per contrassegnare una snapshot manuale come privata, rimuovi il valore `all` dall'elenco dei valori per l'attributo `restore`.

Esportazione dei dati del cluster database in Amazon S3

È possibile esportare i dati da un cluster database Amazon Aurora in tempo reale in un bucket Amazon S3. Il processo di esportazione viene eseguito in background e non ha ripercussioni sulle prestazioni dell'istanza del cluster di database attivo.

Per impostazione predefinita, vengono esportati tutti i dati nel cluster database. Tuttavia, è possibile scegliere di esportare set specifici di database, schemi o tabelle.

Amazon Aurora clona il cluster database, estrae i dati dal clone e li archivia in un bucket Amazon S3. I dati vengono archiviati in un formato Apache Parquet compresso e coerente. I singoli file Parquet hanno in genere una dimensione compresa tra 1 e 10 MB.

Le prestazioni più veloci che puoi ottenere con l'esportazione dei dati delle istantanee per Aurora My SQL versione 2 e versione 3 non si applicano all'esportazione dei dati del cluster DB. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Ti viene addebitato il costo dell'esportazione dell'intero cluster di database, indipendentemente dal fatto che esporti tutti o parte dei dati. Per ulteriori informazioni, consulta la pagina Prezzi di Amazon Aurora.

Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni sull'utilizzo di Athena per leggere i dati di Parquet, consulta Parquet SerDe nella Guida per l'utente di Amazon Athena. Per ulteriori informazioni sull'utilizzo di Redshift Spectrum per leggere i dati di Parquet, consulta i formati di dati COPY a colonne nella Amazon Redshift Database Developer Guide.

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni per l'esportazione dei dati del cluster database in S3, consulta Regioni supportate e motori Aurora DB per l'esportazione dei dati del cluster in Amazon S3.

Per esportare i dati del cluster database in un bucket Amazon S3 puoi utilizzare il processo riportato di seguito. Per ulteriori dettagli, consulta le seguenti sezioni:

Panoramica sull'esportazione dei dati di un cluster database

1. Identifica il cluster database di cui desideri esportare i dati.
2. Configurare l'accesso al bucket Simple Storage Service (Amazon S3).

Un bucket è un container per oggetti o file Simple Storage Service (Amazon S3). Per fornire le informazioni per accedere a un bucket, attenersi alla seguente procedura:

a. Identifica il bucket S3 in cui devono essere esportati i dati del cluster database. Il bucket S3 deve trovarsi nella stessa Regione AWS del cluster database. Per ulteriori informazioni, consulta Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare.

b. Crea un ruolo AWS Identity and Access Management (IAM) che conceda all'attività di esportazione del cluster DB l'accesso al bucket S3. Per ulteriori informazioni, consulta Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM.

3. Crea una crittografia simmetrica per la crittografia AWS KMS key lato server. La KMS chiave viene utilizzata dall'attività di esportazione del cluster per configurare la crittografia AWS KMS lato server durante la scrittura dei dati di esportazione su S3.

   La politica KMS chiave deve includere sia le autorizzazioni che. `kms:CreateGrant` `kms:DescribeKey` Per ulteriori informazioni sull'uso KMS delle chiavi in Amazon Aurora, consulta. AWS KMS key gestione

   Se nella tua policy KMS chiave è presente una dichiarazione di rifiuto, assicurati di escludere esplicitamente il responsabile del AWS servizio. `export.rds.amazonaws.com`

   Puoi utilizzare una KMS chiave all'interno del tuo AWS account oppure puoi utilizzare una chiave per più account. KMS Per ulteriori informazioni, consulta Utilizzo di un account multiplo AWS KMS key.

4. Esporta il cluster DB in Amazon S3 utilizzando la console o il `start-export-task` CLI comando. Per ulteriori informazioni, consulta Creazione di attività di esportazione di cluster DB.

5. Per accedere ai dati esportati nel bucket Simple Storage Service (Amazon S3), consulta Caricamento, download e gestione di oggetti nella Guida per l'utente di Amazon Simple Storage Service.

Scopri come configurare, esportare, monitorare, annullare e risolvere i problemi delle attività di esportazione dei cluster DB nelle seguenti sezioni.

Argomenti

- Considerazioni per le esportazioni di cluster DB
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)
- Creazione di attività di esportazione di cluster DB

- Monitoraggio delle attività di esportazione del cluster database
- Annullamento di un'attività di esportazione del cluster database
- Risoluzione dei problemi delle esportazioni di cluster DB

Considerazioni per le esportazioni di cluster DB

Utilizza le seguenti sezioni per conoscere le limitazioni, le convenzioni di denominazione dei file e la conversione e lo storage dei dati durante l'esportazione di dati di cluster DB in Amazon S3.

Argomenti

- Limitazioni
- Convenzione di denominazione file
- Formato di conversione e archiviazione dei dati

Limitazioni

L'esportazione dei dati del cluster database in Amazon S3 presenta le seguenti limitazioni:

- Non è possibile eseguire contemporaneamente più attività di esportazione per lo stesso cluster database. Ciò è valido sia per le esportazioni totali sia per le esportazioni parziali.
- Puoi avere in corso fino a cinque attività simultanee di esportazione di snapshot DB per Account AWS.
- Aurora Serverless v1 I cluster DB non supportano le esportazioni verso S3.
- Aurora My e SQL Aurora Postgre SQL supportano le esportazioni verso S3 solo per la modalità motore fornita.
- Le esportazioni verso S3 non supportano i prefissi S3 contenenti i due punti (:).
- I seguenti caratteri nel percorso del file S3 vengono convertiti in caratteri di sottolineatura (_) durante l'esportazione:

```
\ ` " (space)
```

- Se un database, uno schema o una tabella contiene caratteri diversi da quelli riportati di seguito, l'esportazione parziale non è supportata. Tuttavia, è possibile esportare l'intero cluster database.
  - Lettere latine (A–Z)

  - Numeri (0–9)
  - Simbolo del dollaro ($)
  - Carattere di sottolineatura (_)
- Gli spazi ( ) e alcuni caratteri non sono supportati nei nomi delle colonne delle tabelle del database. Le tabelle con i seguenti caratteri nei nomi delle colonne vengono ignorate durante l'esportazione:

```
, ; { } ( ) \n \t = (space)
```

- Le tabelle con barre (/) nei rispettivi nomi vengono ignorate durante l'esportazione.
- Le tabelle SQL temporanee e non registrate di Aurora Postgre vengono ignorate durante l'esportazione.
- Se i dati contengono un oggetto di grandi dimensioni, ad esempio un BLOB oCLOB, che è vicino o superiore a 500 MB, l'esportazione non riesce.
- Se una tabella contiene una riga di grandi dimensioni, vicine o superiori a 2 GB, la tabella viene ignorata durante l'esportazione.
- Per le esportazioni parziali, l'`ExportOnly`elenco ha una dimensione massima di 200 KB.
- Si consiglia vivamente di utilizzare un nome univoco per ogni attività di esportazione. Se non utilizzi un nome di attività univoco, potresti ricevere il seguente messaggio di errore:

  ExportTaskAlreadyExistsFault: Si è verificato un errore (ExportTaskAlreadyExists) durante la chiamata dell' StartExportTaskoperazione: L'operazione di esportazione con l'ID *`xxxxx`* esiste già.
- Poiché alcune tabelle potrebbero essere ignorate, ti consigliamo di verificare il numero di righe e tabelle nei dati dopo l'esportazione.

Convenzione di denominazione file

I dati esportati per tabelle specifiche vengono memorizzati nel formato *`base_prefix`*/*`files`*, dove il prefisso di base è il seguente:

```
export_identifier/database_name/schema_name.table_name/
```

Per esempio:

```
export-1234567890123-459/rdststcluster/mycluster.DataInsert_7ADB5D19965123A2/
```

I file di output utilizzano la seguente convenzione di denominazione, dove *partition_index* è alfanumerico:

```
partition_index/part-00000-random_uuid.format-based_extension
```

Per esempio:

```
1/part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
    a/part-00000-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
```

La convenzione di denominazione file è soggetta a modifiche. Pertanto, quando usi le tabelle di destinazione ti consigliamo di leggere tutto quanto riportato all'interno del prefisso di base della tabella.

Formato di conversione e archiviazione dei dati

Quando esporti un cluster DB in un bucket Amazon S3, Amazon Aurora converte, esporta e archivia i dati nel formato Parquet. Per ulteriori informazioni, consulta Conversione dei dati durante l'esportazione in un bucket Simple Storage Service (Amazon S3).

Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)

Individua il bucket Amazon S3 e fornisci all'attività di esportazione del cluster database l'autorizzazione per accedervi.

Argomenti

- Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare
- Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM
- Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount

Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare

Identifica il bucket Amazon S3 in cui esportare il cluster database. Utilizzare un bucket S3 esistente o crearne uno nuovo.

> **Note**
>
> Il bucket S3 deve trovarsi nella stessa AWS regione del cluster DB.

Per ulteriori informazioni sull'utilizzo dei bucket Simple Storage Service (Amazon S3), vedere quanto segue in Guida per l'utente di Amazon Simple Storage Service:

- Come visualizzare le proprietà di un bucket S3?
- In che modo si abilita la crittografia di default per un bucket Amazon S3?
- Come creare un bucket S3?

Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM

Prima di esportare i dati del cluster database in Amazon S3, fornisci l'autorizzazione di accesso in scrittura alle attività di esportazione al bucket Amazon S3.

Per concedere questa autorizzazione, crea una IAM policy che fornisca l'accesso al bucket, quindi crea un IAM ruolo e allega la policy al ruolo. Successivamente, puoi assegnare il IAM ruolo all'attività di esportazione del cluster DB.

> **Important**
>
> Se prevedi di utilizzare il AWS Management Console per esportare il tuo cluster DB, puoi scegliere di creare automaticamente la IAM policy e il ruolo quando esporti il cluster DB. Per istruzioni, consulta Creazione di attività di esportazione di cluster DB.

Per fornire alle attività l'accesso ad Amazon S3

1. Crea una IAM policy. Questa policy fornisce le autorizzazioni per bucket e oggetti che consentono all'attività di esportazione del cluster database l'accesso ad Amazon S3.

   Includi nella policy le seguenti operazioni obbligatorie per consentire il trasferimento dei file da Amazon Aurora a un bucket S3:

   - `s3:PutObject*`
   - `s3:GetObject*`

- `s3:ListBucket`
- `s3:DeleteObject*`
- `s3:GetBucketLocation`

Includi nella policy le seguenti risorse per identificare il bucket S3 e gli oggetti nel bucket. Il seguente elenco di risorse mostra il formato Amazon Resource Name (ARN) per accedere ad Amazon S3.

- `arn:aws:s3:::amzn-s3-demo-bucket`
- `arn:aws:s3:::amzn-s3-demo-bucket/*`

Per ulteriori informazioni sulla creazione di una IAM policy per Amazon Aurora, consulta. Creazione e utilizzo di una policy IAM per l'accesso al database IAM Vedi anche Tutorial: Crea e allega la tua prima politica gestita dai clienti nella Guida per l'IAMutente.

Il AWS CLI comando seguente crea una IAM politica denominata `ExportPolicy` con queste opzioni. Garantisce l'accesso a un bucket denominato *amzn-s3-demo-bucket*.

> **Note**
>
> Dopo aver creato la politica, annota ARN la politica. È necessario eseguire ARN un passaggio successivo quando si associa la politica a un IAM ruolo.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Crea un IAM ruolo, in modo che Aurora possa assumerlo IAM per tuo conto per accedere ai tuoi bucket Amazon S3. Per ulteriori informazioni, consulta Creare un ruolo per delegare le autorizzazioni a un IAM utente nella Guida per l'utente. IAM

   L'esempio seguente mostra l'utilizzo del AWS CLI comando per creare un ruolo denominato. `rds-s3-export-role`

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Allega la IAM politica che hai creato al IAM ruolo che hai creato.

   Il AWS CLI comando seguente collega la politica creata in precedenza al ruolo denominato`rds-s3-export-role`. Sostituiscilo *`your-policy-arn`* con la politica ARN che hai annotato in un passaggio precedente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount

Puoi utilizzare i bucket S3 su AWS più account. Per ulteriori informazioni, consulta Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount.

Creazione di attività di esportazione di cluster DB

Crea attività di esportazione per esportare i dati dal tuo cluster Aurora DB a un bucket Amazon S3. È possibile avere fino a cinque attività di esportazione di cluster database simultanee in esecuzione per ogni Account AWS.

> **Note**
>
> L'esportazione dei dati del cluster database può richiedere qualche minuto a seconda del tipo e delle dimensioni del database. L'attività di esportazione clona e ridimensiona innanzitutto l'intero database prima di estrarre i dati su Amazon S3. Lo stato di avanzamento dell'attività durante questa fase viene visualizzato come Avvio. Quando l'attività passa all'esportazione dei dati in S3, lo stato di avanzamento diventa In progress (In corso).
> Il tempo necessario per completare l'esportazione dipende dai dati memorizzati nel database. Ad esempio, le tabelle con chiave primaria numerica o colonne indice ben distribuite esporteranno più velocemente. Le tabelle che non contengono una colonna adatta al partizionamento e le tabelle con un solo indice su una colonna basata su stringhe richiedono più tempo perché l'esportazione utilizza un processo a thread singolo più lento.

Puoi esportare i dati del cluster DB in Amazon S3 utilizzando AWS Management Console, the AWS CLI, o. RDS API

Se usi una funzione Lambda per esportare i dati del cluster database, aggiungi l'operazione `kms:DescribeKey` alla policy della funzione Lambda. Per ulteriori informazioni, consulta Autorizzazioni di AWS Lambda.

Console

L'opzione Export to Amazon S3 (Esporta in Amazon S3) viene visualizzata solo per i cluster database che possono essere esportati in Amazon S3. Un cluster database potrebbe non essere disponibile per l'esportazione a causa dei seguenti motivi:

- Il motore del database non è supportato per l'esportazione S3.
- La versione del cluster database non è supportata per l'esportazione in S3.
- L'esportazione da S3 non è supportata nella AWS regione in cui è stato creato il cluster DB.

Per esportare i dati del cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli il cluster database di cui desideri esportare i dati.
4. Per Actions (Operazioni), scegli Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3).

   Viene visualizzata la finestra Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).
5. Per Export identifier (Identificatore di esportazione), immettere un nome per identificare l'attività di esportazione. Questo valore viene utilizzato anche per il nome del file creato nel bucket S3.
6. Scegli i dati da esportare:

   - Scegli All (Tutto) per esportare tutti i dati del cluster database.
   - Scegli Partial (Parziale) per esportare parti specifiche del cluster database. Per identificare le parti del cluster da esportare, immetti uno o più database, schemi o tabelle per Identifiers (Identificatori), separati da spazi.

     Utilizza il seguente formato:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.schemaN]
     [.tablen]
     ```

     Ad esempio:

     ```
     mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
      mydatabase2.myschema2.mytable2
     ```
7. Per S3 bucket (Bucket S3), scegliere il bucket in cui esportare.

   Per assegnare i dati esportati a un percorso di cartella nel bucket S3, immettere il percorso opzionale per S3 prefix (Prefisso S3).
8. Per IAMquanto riguarda il ruolo, scegli un ruolo che ti garantisca l'accesso in scrittura al bucket S3 che hai scelto oppure crea un nuovo ruolo.

- Se è stato creato un ruolo seguendo le fasi in Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM, scegliere tale ruolo.
- Se non è stato creato un ruolo che fornisce l'accesso in scrittura al bucket S3 scelto, scegli Create a new role (Crea un nuovo ruolo) per creare automaticamente il ruolo. Successivamente, inserisci un nome per il ruolo nel IAM nome del ruolo.

9. Per KMSchiave, inserisci la ARN chiave da utilizzare per crittografare i dati esportati.
10. Scegliere Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).

AWS CLI

Per esportare i dati del cluster DB in Amazon S3 utilizzando AWS CLI, usa il start-export-taskcomando con le seguenti opzioni obbligatorie:

- `--export-task-identifier`
- `--source-arn`— l'Amazon Resource Name (ARN) del cluster DB
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

Nei seguenti esempi, l'attività di esportazione è denominata *`my-cluster-export`*, che esporta i dati in un bucket S3 denominato *`amzn-s3-demo-destination-bucket`*.

Example

PerLinux, omacOS: Unix

```
aws rds start-export-task \
    --export-task-identifier my-cluster-export \
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Per Windows:

```
aws rds start-export-task ^
```

```
    --export-task-identifier my-DB-cluster-export ^
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Di seguito è riportato un output di esempio.

```
{
    "ExportTaskIdentifier": "my-cluster-export",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "IamRoleArn": "arn:aws:iam:123456789012:role/ExportTest",
    "KmsKeyId": "my-key",
    "Status": "STARTING",
    "PercentProgress": 0,
    "TotalExtractedDataInGB": 0,
}
```

Per fornire un percorso di cartella nel bucket S3 per l'esportazione del cluster DB, includi l'`--s3-prefix`opzione nel comando. start-export-task

RDS API

Per esportare i dati del cluster DB in Amazon S3 utilizzando Amazon RDSAPI, utilizza l'StartExportTaskoperazione con i seguenti parametri obbligatori:

- `ExportTaskIdentifier`
- `SourceArn`— il ARN del cluster DB
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Monitoraggio delle attività di esportazione del cluster database

È possibile monitorare le esportazioni di cluster DB utilizzando AWS Management Console AWS CLI, the o RDSAPI.

Console

Per monitorare le esportazioni del cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni del cluster database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Per visualizzare informazioni dettagliate su un'esportazione del cluster database specifica, scegli l'attività di esportazione.

AWS CLI

Per monitorare le attività di esportazione del cluster DB utilizzando il AWS CLI, usa il describe-export-taskscomando.

Nell'esempio seguente viene illustrato come visualizzare le informazioni correnti su tutte le esportazioni del cluster database.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2022-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:parameter-groups-
test"
        },
```

```
{
            "Status": "COMPLETE",
            "TaskStartTime": "2022-10-31T20:58:06.998Z",
            "TaskEndTime": "2022-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2022-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket amzn-s3-demo-bucket2 isn't located in the
 current AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-30-06-45"
        }
    ]
}
```

Per visualizzare informazioni su un'attività di esportazione specifica, includi l'opzione `--export-task-identifier` nel comando `describe-export-tasks`. Per filtrare l'output, includere l'opzione `--Filters`. Per ulteriori opzioni, vedete il describe-export-taskscomando.

RDS API

Per visualizzare informazioni sulle esportazioni di cluster DB utilizzando Amazon RDSAPI, utilizza l'DescribeExportTasksoperazione.

Per tenere traccia del completamento del flusso di lavoro di esportazione o per attivare un altro flusso di lavoro, è possibile sottoscrivere gli argomenti del Servizio di notifica semplice Amazon. Per ulteriori informazioni su AmazonSNS, consultaUtilizzo delle notifiche di RDS eventi di Amazon.

Annullamento di un'attività di esportazione del cluster database

Puoi annullare un'attività di esportazione di un cluster DB utilizzando il AWS Management Console AWS CLI, il o il RDSAPI.

> **Note**
>
> L'annullamento di un'attività di esportazione non rimuove i dati esportati in Amazon S3. Per informazioni su come eliminare i dati utilizzando la console, consultare Come eliminare oggetti da un bucket S3? Per eliminare i dati utilizzandoCLI, utilizzare il comando delete-object.

Console

Per annullare un'attività di esportazione del cluster database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni del cluster database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Scegli l'attività di esportazione che vuoi annullare.
4. Seleziona Cancel (Annulla).
5. Scegli Cancel export task (Annulla attività di esportazione) nella pagina di conferma.

AWS CLI

Per annullare un'operazione di esportazione utilizzando il AWS CLI, usa il cancel-export-taskcomando. Il comando richiede l'opzione `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my-export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my-export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:export-example-1"
}
```

RDS API

Per annullare un'operazione di esportazione utilizzando Amazon RDSAPI, utilizza l'CancelExportTaskoperazione con il `ExportTaskIdentifier` parametro.

Risoluzione dei problemi delle esportazioni di cluster DB

Utilizza le seguenti sezioni per risolvere i messaggi di errore e gli errori di SQL autorizzazione Postgre per le attività di esportazione di cluster DB su Amazon S3.

Messaggi di errore per le attività di esportazione di Amazon S3

Nella tabella seguente vengono descritti i messaggi restituiti quando le attività di esportazione di Amazon S3 non riescono.

Messaggio di errore	Descrizione
Impossibile trovare o accedere al cluster database di origine: [nome cluster]	Il cluster database di origine non può essere clonato.

Messaggio di errore	Descrizione
Si è verificato un errore interno sconosciuto.	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
Si è verificato un errore interno sconosciuto durante la scrittura dei metadati dell'attività di esportazione nel bucket S3 [nome bucket].	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
L'RDSesportazione non è riuscita a scrivere i metadati dell'attività di esportazione perché non può assumere il ruolo [ruolo]. IAM ARN	L'attività di esportazione assume il IAM ruolo dell'utente di verificare se è consentito scrivere metadati nel bucket S3. Se l'attività non può assumere il tuo IAM ruolo, ha esito negativo.
L'RDSesportazione non è riuscita a scrivere i metadati dell'attività di esportazione nel bucket S3 [nome del bucket] utilizzando il IAM ruolo [ruoloARN] con la KMS chiave [ID chiave]. Codice di errore: [codice di errore]	Mancano una o più autorizzazioni, quindi l'attività di esportazione non può accedere al bucket S3. Questo messaggio di errore viene generato quando si riceve uno dei seguenti codici di errore: • `AWSSecurityTokenServiceException` con il codice di errore `AccessDenied` • `AmazonS3Exception` con il codice di errore`NoSuchBucket` ,`AccessDenied` ,`KMS.KMSInvalidStateException` ,`403 Forbidden` , oppure `KMS.DisabledException` Questi codici di errore indicano che le impostazioni non sono configurate correttamente per il IAM ruolo, il bucket S3 o la chiave. KMS
Il IAM ruolo [ruoloARN] non è autorizzato a chiamare [azione S3] sul bucket S3 [nome del bucket]. Controlla le tue autorizzazioni e riprova l'esportazione.	La IAM policy non è configurata correttamente. L'autorizzazione per l'azione S3 specifica sul bucket S3 è mancante e questa condizione causa l'esito negativo dell'attività di esportazione.

Messaggio di errore	Descrizione
KMScontrollo delle chiavi non riuscito. Controlla le credenziali sulla tua KMS chiave e riprova.	Il controllo delle credenziali della KMS chiave non è riuscito.
Controllo delle credenziali S3 non riuscito. Controlla le autorizzazioni sul tuo bucket S3 e sulla tua policy. IAM	Il controllo delle credenziali S3 non è riuscito.
Il bucket S3 [nome bucket] non è valido. Non si trova nella corrente Regione AWS oppure non esiste. Esamina il nome del bucket S3 e riprova l'esporta zione.	Bucket S3 non è valido.
Il bucket S3 [nome bucket] non si trova nella corrente Regione AWS. Esamina il nome del bucket S3 e riprova l'esporta zione.	Il bucket S3 è sbagliato. Regione AWS

Risoluzione degli errori relativi alle autorizzazioni di Postgres SQL

Durante l'esportazione di SQL database Postgre su Amazon S3, potresti visualizzare un `PERMISSIONS_DO_NOT_EXIST` errore che indica che alcune tabelle sono state ignorate. Questo errore si verifica in genere quando l'utente con privilegi avanzati che hai specificato durante la creazione del cluster database, non dispone delle autorizzazioni per accedere alle tabelle.

Per risolvere questo errore, eseguire il comando seguente:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Per ulteriori informazioni sui privilegi utente con privilegi avanzati, vedere Privilegi dell'account utente master.

Esportazione dei dati dello snapshot del cluster di database in Amazon S3

È possibile esportare i dati dello snapshot del cluster di database in un bucket Amazon S3. Il processo di esportazione viene eseguito in background e non ha ripercussioni sulle prestazioni dell'istanza del cluster di database attivo.

Quando si esporta uno snapshot del cluster di database, Amazon Aurora estrae i dati dallo snapshot e li archivia in un bucket Amazon S3. È possibile esportare snapshot manuali e snapshot automatici di sistema. Per impostazione predefinita, vengono esportati tutti i dati nello snapshot. Tuttavia, è possibile scegliere di esportare set specifici di database, schemi o tabelle.

> **Note**
>
> L'esportazione di dati da uno snapshot del cluster DB richiede il ripristino dello snapshot. I tempi di ripristino sono influenzati da vari fattori, come la quantità di traffico di rete e Regione AWS riceve in relazione alla larghezza di banda disponibile. In caso di aumento improvviso del traffico, è possibile che si verifichino tempi di completamento più lunghi del previsto. Un'alternativa per ridurre i tempi di esportazione S3 per i database Aurora è l'esportazione live di cluster DB su S3. L'esportazione di cluster DB ha tempi di avvio più brevi rispetto all'esportazione di snapshot DB, perché non è necessario ripristinare un'istantanea. Per ulteriori informazioni, consulta Esportazione dei dati del cluster database in Amazon S3.

I dati vengono archiviati in un formato Apache Parquet compresso e coerente. I singoli file Parquet hanno in genere una dimensione compresa tra 1 e 10 MB.

Dopo l'esportazione dei dati, è possibile analizzare i dati esportati direttamente mediante strumenti quali Amazon Athena o Amazon Redshift Spectrum. Per ulteriori informazioni sull'utilizzo di Athena per leggere i dati di Parquet, consulta Parquet SerDe nella Guida per l'utente di Amazon Athena. Per ulteriori informazioni sull'utilizzo di Redshift Spectrum per leggere i dati di Parquet, consulta i formati di dati COPY a colonne nella Amazon Redshift Database Developer Guide.

La disponibilità e il supporto delle funzionalità variano a seconda delle versioni specifiche di ciascun motore di database e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità della versione e della regione dell'esportazione dei dati di snapshot del cluster DB in S3, consulta. Regioni supportate e motori Aurora DB per l'esportazione di dati di snapshot su Amazon S3

Per esportare i dati dello snapshot DB in un bucket Simple Storage Service (Amazon S3) puoi utilizzare il processo riportato di seguito. Per ulteriori dettagli, consulta le seguenti sezioni:

Panoramica sull'esportazione dei dati degli snapshot

1. Identificare lo snapshot da esportare.

   Utilizzare uno snapshot automatico o manuale esistente oppure creare uno snapshot manuale di un'istanza database.

2. Configurare l'accesso al bucket Simple Storage Service (Amazon S3).

   Un bucket è un container per oggetti o file Simple Storage Service (Amazon S3). Per fornire le informazioni per accedere a un bucket, attenersi alla seguente procedura:

   a. Identificare il bucket S3 in cui deve essere esportato lo snapshot. Il bucket S3 deve trovarsi nello stesso AWS Regione come istantanea. Per ulteriori informazioni, consulta Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare.

   b. Crea un AWS Identity and Access Management (IAM) ruolo che concede all'attività di esportazione delle istantanee l'accesso al bucket S3. Per ulteriori informazioni, consulta Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM.

3. Crea una crittografia simmetrica AWS KMS key per la crittografia lato server. La KMS chiave viene utilizzata dall'attività di esportazione delle istantanee per la configurazione AWS KMS crittografia lato server durante la scrittura dei dati di esportazione su S3.

   La politica KMS chiave deve includere sia le autorizzazioni che. `kms:CreateGrant` `kms:DescribeKey` Per ulteriori informazioni sull'uso KMS delle chiavi in Amazon Aurora, consulta. AWS KMS key gestione

   Se hai una dichiarazione di negazione nella tua policy KMS chiave, assicurati di escludere esplicitamente il AWS principale del servizio. `export.rds.amazonaws.com`

   Puoi usare una KMS chiave all'interno del tuo AWS account oppure puoi usare una KMS chiave multiaccount. Per ulteriori informazioni, consulta Utilizzo di un account multiplo AWS KMS key.

4. Esporta lo snapshot in Amazon S3 utilizzando la console o `start-export-task` CLI il comando. Per ulteriori informazioni, consulta Creazione di attività di esportazione di istantanee.

5. Per accedere ai dati esportati nel bucket Simple Storage Service (Amazon S3), consulta Caricamento, download e gestione di oggetti nella Guida per l'utente di Amazon Simple Storage Service.

Scopri come configurare, esportare, monitorare, annullare e risolvere i problemi relativi alle attività di esportazione degli snapshot del cluster DB nelle seguenti sezioni.

Argomenti

- Considerazioni per le esportazioni di snapshot del cluster DB
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)
- Creazione di attività di esportazione di istantanee
- Monitoraggio delle esportazioni di snapshot
- Annullamento di un'attività di esportazione di snapshot
- Andamento delle esportazioni in Aurora My SQL
- Risoluzione dei problemi relativi all'esportazione di istantanee

Considerazioni per le esportazioni di snapshot del cluster DB

Limitazioni

L'esportazione dei dati snapshot DB in Simple Storage Service (Amazon S3) presenta le seguenti limitazioni:

- Non è possibile eseguire contemporaneamente più attività di esportazione per lo stesso snapshot del cluster database. Ciò è valido sia per le esportazioni totali sia per le esportazioni parziali.
- È possibile avere in corso fino a cinque attività simultanee di esportazione di snapshot DB per volta. Account AWS
- Non è possibile esportare i dati delle istantanee da Aurora Serverless v1 Cluster DB su S3.
- Le esportazioni verso S3 non supportano i prefissi S3 contenenti i due punti (:).
- I seguenti caratteri nel percorso del file S3 vengono convertiti in caratteri di sottolineatura (_) durante l'esportazione:

```
\ ` " (space)
```

- Se un database, uno schema o una tabella contiene caratteri diversi da quelli riportati di seguito, l'esportazione parziale non è supportata. Tuttavia, è possibile esportare l'intero snapshot DB.
  - Lettere latine (A–Z)
  - Numeri (0–9)
  - Simbolo del dollaro ($)

  - Carattere di sottolineatura (_)
- Gli spazi ( ) e alcuni caratteri non sono supportati nei nomi delle colonne delle tabelle del database. Le tabelle con i seguenti caratteri nei nomi delle colonne vengono ignorate durante l'esportazione:

```
, ; { } ( ) \n \t = (space)
```

- Le tabelle con barre (/) nei rispettivi nomi vengono ignorate durante l'esportazione.
- Le tabelle temporanee e non registrate di Aurora PostgreSQL vengono ignorate durante l'esportazione.
- Se i dati contengono un oggetto di grandi dimensioni, ad esempio un BLOB o un CLOB, vicino o superiore a 500 MB, l'esportazione non riesce.
- Se una tabella contiene una riga di grandi dimensioni, vicine o superiori a 2 GB, la tabella viene ignorata durante l'esportazione.
- Per le esportazioni parziali, l'`ExportOnly`elenco ha una dimensione massima di 200 KB.
- Si consiglia vivamente di utilizzare un nome univoco per ogni attività di esportazione. Se non utilizzi un nome di attività univoco, potresti ricevere il seguente messaggio di errore:

  ExportTaskAlreadyExistsFault: Si è verificato un errore (ExportTaskAlreadyExists) durante la chiamata dell' StartExportTaskoperazione: l'attività di esportazione con l'ID esiste *xxxxx* già.
- È possibile eliminare uno snapshot durante l'esportazione dei suoi dati in S3, ma vengono comunque addebitati i costi di storage per tale snapshot fino al completamento dell'attività di esportazione.
- Non puoi ripristinare i dati degli snapshot esportati da S3 in un nuovo cluster di database.

Convenzione di denominazione file

I dati esportati per tabelle specifiche vengono memorizzati nel formato *base_prefix*/*files*, dove il prefisso di base è il seguente:

```
export_identifier/database_name/schema_name.table_name/
```

Ad esempio:

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

Esistono due convenzioni di denominazione per i file.

- Convenzione attuale:

```
batch_index/part-partition_index-random_uuid.format-based_extension
```

L'indice batch è un numero di sequenza che rappresenta un batch di dati letti dalla tabella. Se non riusciamo a partizionare la tabella in piccoli blocchi da esportare in parallelo, ci saranno più indici batch. La stessa cosa accade se la tabella è partizionata in più tabelle. Ci saranno più indici batch, uno per ciascuna delle partizioni di tabella della tabella principale.

Se riusciamo a partizionare la tabella in piccoli blocchi da leggere in parallelo, ci sarà solo la cartella batch index. 1

All'interno della cartella dell'indice batch, ci sono uno o più file Parquet che contengono i dati della tabella. Il prefisso del nome del file Parquet è. `part-`*`partition_index`* Se la tabella è partizionata, ci saranno più file che iniziano con l'indice delle partizioni. `00000`

Possono esserci delle lacune nella sequenza dell'indice delle partizioni. Ciò accade perché ogni partizione è ottenuta da una query a intervalli nella tabella. Se non ci sono dati nell'intervallo di quella partizione, quel numero di sequenza viene ignorato.

Ad esempio, supponiamo che la `id` colonna sia la chiave primaria della tabella e che i suoi valori minimo e massimo siano e. `100` `1000` Quando proviamo a esportare questa tabella con nove partizioni, la leggiamo con query parallele come le seguenti:

```
SELECT * FROM table WHERE id <= 100 AND id < 200
 SELECT * FROM table WHERE id <= 200 AND id < 300
```

Questo dovrebbe generare nove file, da a`part-00000-`*`random_uuid`*`.gz.parquet`. `part-00008-`*`random_uuid`*`.gz.parquet` Tuttavia, se non ci sono righe IDs comprese tra `200` e`350`, una delle partizioni completate è vuota e non viene creato alcun file per essa. Nell'esempio precedente, `part-00001-`*`random_uuid`*`.gz.parquet` non viene creato.

- Convenzione precedente:

```
part-partition_index-random_uuid.format-based_extension
```

È la stessa della convenzione attuale, ma senza il *`batch_index`* prefisso, ad esempio:

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
```

```
part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

La convenzione di denominazione file è soggetta a modifiche. Pertanto, quando usi le tabelle di destinazione ti consigliamo di leggere tutto quanto riportato all'interno del prefisso di base della tabella.

Conversione dei dati durante l'esportazione in un bucket Simple Storage Service (Amazon S3)

Quando si esporta uno snapshot di database in un bucket Amazon S3, Amazon Aurora converte, esporta e memorizza i dati nel formato Parquet. Per ulteriori informazioni su Parquet, consultare il sito Web Apache Parquet.

Parquet archivia tutti i dati in uno dei seguenti tipi primitivi:

- BOOLEAN
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY: un array di byte a lunghezza variabile, noto anche come binario
- FIXED_LEN_BYTE_ARRAY: un array di byte a lunghezza fissa utilizzato quando i valori hanno una dimensione costante

I tipi di dati Parquet sono pochi per ridurre la complessità di lettura e scrittura del formato. Parquet fornisce tipi logici per estendere i tipi primitivi. Un tipo logico viene implementato come annotazione con i dati in un campo di metadati `LogicalType`. L'annotazione di tipo logico spiega come interpretare il tipo primitivo.

Quando il tipo logico `STRING` annota un tipo `BYTE_ARRAY`, indica che l'array di byte deve essere interpretato come una stringa di caratteri con codifica UTF-8. Al termine di un'attività di esportazione, Amazon Aurora notifica all'utente se si è verificata una conversione di stringa. I dati sottostanti esportati sono sempre uguali ai dati provenienti dall'origine. Tuttavia, a causa della differenza di

codifica in UTF-8, alcuni caratteri potrebbero apparire diversi dall'origine quando vengono letti in strumenti come Athena.

Per ulteriori informazioni, consultare la sezione relativa alle definizioni dei tipi logici di Parquet nella documentazione di Parquet.

Argomenti

- Mappatura dei tipi di dati MySQL e MariaDB al formato Parquet
- Mappatura dei tipi di dati PostgreSQL su Parquet

Mappatura dei tipi di dati MySQL e MariaDB al formato Parquet

La tabella seguente mostra la mappature tra i tipi di dati MySQL e i tipi di dati Parquet quando i dati vengono convertiti ed esportati in Amazon S3.

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
Tipi di dati numerici			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN _BYTE_ARRAY(9)	DECIMAL(20,0)	Parquet supporta solo tipi firmati, quindi la mappatura richiede un byte aggiuntivo (8 più 1) per memorizza re il tipo BIGINT_UN SIGNED.
BIT	BYTE_ARRAY		
DECIMAL	INT32	DECIMAL(p,s)	Se il valore di origine è inferiore a 2^{31}, viene memorizzato come INT32.
	INT64	DECIMAL(p,s)	Se il valore di origine è 2^{31} o superiore, ma

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
			inferiore a 2^{63}, viene archiviato comeINT64 .
	FIXED_LEN _BYTE_ARRAY(N)	DECIMAL(p,s)	Se il valore di origine è 2^{63} o superiore , viene archiviato come FIXED_LEN _BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	Parquet non supporta la precisione decimale superiore a 38. Il valore decimale viene convertito in una stringa di tipo BYTE_ARRAY e codificato come. UTF8
DOUBLE	DOUBLE		
FLOAT	DOUBLE		
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
NUMERIC	INT32	DECIMAL(p,s)	Se il valore di origine è inferiore a 2^{31}, viene archiviato come. INT32
	INT64	DECIMAL(p,s)	Se il valore di origine è 2^{31} o superiore, ma inferiore a 2^{63}, viene archiviato comeINT64 .
	FIXED_LEN _ARRAY(N)	DECIMAL(p,s)	Se il valore di origine è 2^{63} o superiore , viene archiviato come FIXED_LEN _BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	Parquet non supporta la precisione numerica superiore a 38. Questo valore numerico viene convertito in una stringa di tipo BYTE_ARRAY e codificato come. UTF8
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
TINYINT UNSIGNED	INT32	INT (16, vero)	
Tipi di dati stringa			
BINARY	BYTE_ARRAY		
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
Tipi di dati data e ora			

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
DATE	BYTE_ARRAY	STRING	Una data viene convertita in una stringa di tipo BYTE_ARRAY e codificata come. UTF8
DATETIME	INT64	TIMESTAMP _MICROS	
TIME	BYTE_ARRAY	STRING	Un tipo TIME viene convertito in una stringa in un BYTE_ARRAY e codificato come. UTF8
TIMESTAMP	INT64	TIMESTAMP _MICROS	
YEAR	INT32		
Tipi di dati geometrici			
GEOMETRY	BYTE_ARRAY		
GEOMETRYC OLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		
POINT	BYTE_ARRAY		
POLYGON	BYTE_ARRAY		

Tipo di dati origine	Tipo Parquet primitivo	Annotazione del tipo logico	Note di conversione
Tipo di dati JSON			
JSON	BYTE_ARRAY	STRING	

Mappatura dei tipi di dati PostgreSQL su Parquet

Nella tabella seguente viene illustrata la mappatura dai tipi di dati PostgreSQL ai tipi di dati Parquet quando i dati vengono convertiti ed esportati in Simple Storage Service (Amazon S3).

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
Tipi di dati numerici			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	Un tipo DECIMAL viene convertito in una stringa di tipo BYTE_ARRAY e codificato come. UTF8 Questa conversio ne serve a evitare complicazioni dovute alla precisione dei dati e ai valori di dati che non sono un numero (NaN).
DOUBLE PRECISION	DOUBLE		

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
INTEGER	INT32		
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT (16, vero)	
SMALLSERIAL	INT32	INT (16, vero)	
Tipi di dati stringa e correlati			
ARRAY	BYTE_ARRAY	STRING	Un array viene convertito in una stringa e codificato come BINARY (UTF8). Questa conversione serve a evitare complicazioni dovute alla precisione dei dati, ai valori di dati che non sono un numero (NaN) e ai valori di dati temporali.
BIT	BYTE_ARRAY	STRING	
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	
XML	BYTE_ARRAY	STRING	
Tipi di dati data e ora			
DATE	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP WITH TIME ZONE	BYTE_ARRAY	STRING	
Tipi di dati geometrici			
BOX	BYTE_ARRAY	STRING	
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	

Tipo di dati PostgreSQL	Tipo Parquet primitivo	Annotazione del tipo logico	Note relative alla mappatura
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	
Tipi di dati JSON			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	
Altri tipi di dati			
BOOLEAN	BOOLEAN		
CIDR	BYTE_ARRAY	STRING	Tipo di dati di rete
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	Tipo di dati di rete
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	N/A		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)

Individua il bucket Amazon S3 e fornisci allo snapshot l'autorizzazione per accedervi.

Argomenti

- Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare
- Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM
- Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount
- Utilizzo di un account multiplo AWS KMS key

Identificazione del bucket Simple Storage Service (Amazon S3) in cui esportare

Identificare il bucket Simple Storage Service (Amazon S3) in cui esportare lo snapshot DB. Utilizzare un bucket S3 esistente o crearne uno nuovo.

> **Note**
>
> Il bucket S3 in cui esportare deve trovarsi nella stessa AWS regione dello snapshot.

Per ulteriori informazioni sull'utilizzo dei bucket Simple Storage Service (Amazon S3), vedere quanto segue in Guida per l'utente di Amazon Simple Storage Service:

- Come visualizzare le proprietà di un bucket S3?
- In che modo si abilita la crittografia di default per un bucket Amazon S3?
- Come creare un bucket S3?

Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM

Prima di esportare i dati dello snapshot DB in Simple Storage Service (Amazon S3), fornire l'autorizzazione di accesso in scrittura alle attività di esportazione dello snapshot al bucket Simple Storage Service (Amazon S3).

Per concedere questa autorizzazione, crea una IAM policy che fornisca l'accesso al bucket, quindi crea un IAM ruolo e allega la policy al ruolo. Successivamente, puoi assegnare il IAM ruolo all'attività di esportazione delle istantanee.

> **Important**
>
> Se si prevede di utilizzare il AWS Management Console per esportare l'istantanea, è possibile scegliere di creare automaticamente la IAM politica e il ruolo quando si esporta l'istantanea. Per istruzioni, consulta Creazione di attività di esportazione di istantanee.

Per fornire alle attività dello snapshot DB l'accesso a Amazon S3

1. Crea una politica. IAM Questa policy fornisce le autorizzazioni al bucket e all'oggetto che consentono all'attività di esportazione snapshot l'accesso a Amazon S3.

   Includi nella policy le seguenti operazioni obbligatorie per consentire il trasferimento dei file da Amazon Aurora a un bucket S3:

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

   Includi nella policy le seguenti risorse per identificare il bucket S3 e gli oggetti nel bucket. Il seguente elenco di risorse mostra il formato Amazon Resource Name (ARN) per accedere ad Amazon S3.

   - `arn:aws:s3:::amzn-s3-demo-bucket`
   - `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Per ulteriori informazioni sulla creazione di una IAM policy per Amazon Aurora, consulta. Creazione e utilizzo di una policy IAM per l'accesso al database IAM Vedi anche Tutorial: Crea e allega la tua prima politica gestita dai clienti nella Guida per l'IAMutente.

   Il AWS CLI comando seguente crea una IAM politica denominata `ExportPolicy` con queste opzioni. Garantisce l'accesso a un bucket denominato *`amzn-s3-demo-bucket`*.

> **Note**
>
> Dopo aver creato la politica, annota ARN la politica. È necessario eseguire ARN un passaggio successivo quando si associa la politica a un IAM ruolo.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Crea un IAM ruolo, in modo che Aurora possa assumerlo IAM per tuo conto per accedere ai tuoi bucket Amazon S3. Per ulteriori informazioni, consulta Creare un ruolo per delegare le autorizzazioni a un IAM utente nella Guida per l'utente. IAM

   L'esempio seguente mostra l'utilizzo del AWS CLI comando per creare un ruolo denominato. `rds-s3-export-role`

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
```

```
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Allega la IAM politica che hai creato al IAM ruolo che hai creato.

   Il AWS CLI comando seguente collega la politica creata in precedenza al ruolo denominato`rds-s3-export-role`. Sostituiscilo *`your-policy-arn`* con la politica ARN che hai annotato in un passaggio precedente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
export-role
```

Utilizzo di un bucket Simple Storage Service (Amazon S3) multiaccount

Puoi utilizzare i bucket Amazon S3 su più account. AWS Per utilizzare un bucket tra più account, aggiungi una policy sui bucket per consentire l'accesso al IAM ruolo che stai utilizzando per le esportazioni S3. Per informazioni, consulta Esempio 2: il proprietario del bucket concede autorizzazioni per il bucket multiaccount.

- Allega una policy di bucket al bucket, come mostrato nell'esempio riportato di seguito.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

Utilizzo di un account multiplo AWS KMS key

Puoi utilizzare un account multiplo AWS KMS key per crittografare le esportazioni Amazon S3. Innanzitutto, aggiungi una politica chiave all'account locale, quindi aggiungi IAM le politiche nell'account esterno. Per ulteriori informazioni, consulta Consentire agli utenti di altri account di utilizzare una KMS chiave.

Per utilizzare una chiave per più account KMS

1. Aggiungi una policy di chiave all'account locale.

   Il seguente esempio fornisce `ExampleRole` e `ExampleUser` nell'account esterno 444455556666 autorizzazioni nell'account locale 123456789012.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:role/ExampleRole",
            "arn:aws:iam::444455556666:user/ExampleUser"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "*"
```

```
}
```

2. Aggiungi IAM politiche all'account esterno.

   La seguente IAM politica di esempio consente al principale di utilizzare la KMS chiave nell'account 123456789012 per operazioni crittografiche. Per concedere questa autorizzazione a `ExampleRole` e `ExampleUser` nell'account 444455556666, collega la policy ad essi nell'account.

```
{
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Creazione di attività di esportazione di istantanee

Crea attività di esportazione di istantanee per esportare i dati dalla tua istantanea a un bucket Amazon S3. Puoi avere in corso fino a cinque attività simultanee di esportazione di snapshot DB per volta. Account AWS

> **Note**
>
> L'esportazione delle RDS istantanee può richiedere del tempo a seconda del tipo e delle dimensioni del database. L'attività di esportazione ripristina e ridimensiona innanzitutto l'intero database prima di estrarre i dati su Simple Storage Service (Amazon S3). Lo stato di avanzamento dell'attività durante questa fase viene visualizzato come Avvio. Quando l'attività passa all'esportazione dei dati in S3, lo stato di avanzamento diventa In progress (In corso). Il tempo necessario per completare l'esportazione dipende dai dati memorizzati nel database. Ad esempio, le tabelle con chiave primaria numerica o colonne indice ben distribuite

esporteranno più velocemente. Le tabelle che non contengono una colonna adatta al partizionamento e le tabelle con un solo indice su una colonna basata su stringhe richiedono più tempo. Questo tempo di esportazione più lungo si verifica perché l'esportazione utilizza un processo a thread singolo più lento.

Puoi esportare uno snapshot DB in Amazon S3 utilizzando, AWS Management Console AWS CLI the, o. RDS API

Se si utilizza una funzione Lambda per esportare uno snapshot, aggiungere l'operazione `kms:DescribeKey` alla policy della funzione Lambda. Per ulteriori informazioni, consulta Autorizzazioni di AWS Lambda.

Console

L'opzione Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)) viene visualizzata solo per gli snapshot che possono essere esportati in Simple Storage Service (Amazon S3). Uno snapshot potrebbe non essere disponibile per l'esportazione a causa dei seguenti motivi:

- Il motore del database non è supportato per l'esportazione S3.
- La versione dell'istanza database non è supportata per l'esportazione S3.
- L'esportazione da S3 non è supportata nella AWS regione in cui è stata creata la snapshot.

Per esportare uno snapshot DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, selezionare Snapshots (Snapshot).
3. Dalle schede, scegliere il tipo di snapshot che si desidera esportare.
4. Nell'elenco degli snapshot, scegliere lo snapshot che si desidera esportare.
5. Per Actions (Operazioni), scegli Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3).

   Viene visualizzata la finestra Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).
6. Per Export identifier (Identificatore di esportazione), immettere un nome per identificare l'attività di esportazione. Questo valore viene utilizzato anche per il nome del file creato nel bucket S3.

7. Scegli i dati da esportare:

   - Scegliere All (Tutti) per esportare tutti i dati nello snapshot.
   - Scegliere Partial (Parziali) per esportare parti specifiche dello snapshot. Per identificare le parti dello snapshot da esportare, immettere uno o più database, schemi o tabelle per Identifiers (Identificatori), separati da spazi.

     Utilizza il seguente formato:

     ```
     database[.schema][.table] database2[.schema2][.table2] ... databasen[.scheman]
     [.tablen]
     ```

     Ad esempio:

     ```
     mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
      mydatabase2.myschema2.mytable2
     ```

8. Per S3 bucket (Bucket S3), scegliere il bucket in cui esportare.

   Per assegnare i dati esportati a un percorso di cartella nel bucket S3, immettere il percorso opzionale per S3 prefix (Prefisso S3).

9. Per IAMquanto riguarda il ruolo, scegli un ruolo che ti garantisca l'accesso in scrittura al bucket S3 che hai scelto oppure crea un nuovo ruolo.

   - Se è stato creato un ruolo seguendo le fasi in Fornire l'accesso a un bucket Amazon S3 utilizzando un ruolo IAM, scegliere tale ruolo.
   - Se non è stato creato un ruolo che fornisce l'accesso in scrittura al bucket S3 scelto, scegli Create a new role (Crea un nuovo ruolo) per creare automaticamente il ruolo. Successivamente, inserisci un nome per il ruolo nel IAM nome del ruolo.

10. Per AWS KMS key, inserisci la ARN chiave da utilizzare per crittografare i dati esportati.
11. Scegliere Export to Amazon S3 (Esporta in Simple Storage Service (Amazon S3)).

AWS CLI

Per esportare uno snapshot DB in Amazon S3 utilizzando, usa AWS CLI il comando con start-export-taskle seguenti opzioni richieste:

- `--export-task-identifier`

- `--source-arn`
- `--s3-bucket-name`
- `--iam-role-arn`
- `--kms-key-id`

Negli esempi seguenti, l'attività di esportazione delle istantanee è denominata *my-snapshot-export*, che esporta un'istantanea in un bucket S3 denominato *amzn-s3-demo-destination-bucket*.

Example

PerLinux, o: macOS Unix

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Per Windows:

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Di seguito è riportato un output di esempio.

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
```

```
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

Per fornire un percorso di cartella nel bucket S3 per l'esportazione delle istantanee, includi l'`--s3-prefix`opzione nel comando. start-export-task

RDS API

Per esportare uno snapshot DB su Amazon S3 utilizzando RDS API Amazon, utilizza l'operazione con StartExportTaski seguenti parametri obbligatori:

- `ExportTaskIdentifier`
- `SourceArn`
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Monitoraggio delle esportazioni di snapshot

È possibile monitorare le esportazioni di istantanee DB utilizzando il AWS Management Console, il AWS CLI, o il. RDS API

Console

Per monitorare le esportazioni di snapshot DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni di snapshot di database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Per visualizzare informazioni dettagliate su un'esportazione di snapshot specifica, scegli l'attività di esportazione.

AWS CLI

Per monitorare le esportazioni di snapshot DB utilizzando il AWS CLI, usa il describe-export-taskscomando.

Nell'esempio seguente viene illustrato come visualizzare le informazioni correnti su tutte le esportazioni di snapshot.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
        },
{
            "Status": "COMPLETE",
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
```

```
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket my-exports isn't located in the current AWS
 Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

Per visualizzare informazioni su un'esportazione di snapshot specifica, includere l'opzione `--export-task-identifier` nel comando `describe-export-tasks`. Per filtrare l'output, includere l'opzione `--Filters`. Per ulteriori opzioni, vedete il describe-export-taskscomando.

RDS API

Per visualizzare informazioni sulle esportazioni di snapshot DB utilizzando Amazon RDSAPI, utilizza l'DescribeExportTasksoperazione.

Per tenere traccia del completamento del flusso di lavoro di esportazione o per attivare un altro flusso di lavoro, è possibile sottoscrivere gli argomenti del Servizio di notifica semplice Amazon. Per ulteriori informazioni su AmazonSNS, consultaUtilizzo delle notifiche di RDS eventi di Amazon.

Annullamento di un'attività di esportazione di snapshot

Puoi annullare un'attività di esportazione di snapshot DB utilizzando il AWS Management Console, il AWS CLI, o il. RDS API

> **Note**
>
> L'annullamento di un'attività di esportazione di snapshot non rimuove i dati esportati in Simple Storage Service (Amazon S3). Per informazioni su come eliminare i dati utilizzando la console, consultare Come eliminare oggetti da un bucket S3? Per eliminare i dati utilizzandoCLI, utilizzare il comando delete-object.

Console

Per annullare un'attività di esportazione di uno snapshot

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Exports in Amazon S3 (Esporta in Amazon S3).

   Le esportazioni di snapshot di database sono indicate nella colonna Source type (Tipo di origine). Lo stato dell'esportazione è visualizzato nella colonna Status (Stato).
3. Scegliere l'attività di esportazione di snapshot che si desidera annullare.
4. Seleziona Cancel (Annulla).
5. Scegli Cancel export task (Annulla attività di esportazione) nella pagina di conferma.

AWS CLI

Per annullare un'operazione di esportazione di istantanee utilizzando il AWS CLI, usa il cancel-export-taskcomando. Il comando richiede l'opzione `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
```

```
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
}
```

RDS API

Per annullare un'operazione di esportazione di istantanee utilizzando Amazon RDSAPI, utilizza l'CancelExportTaskoperazione con il `ExportTaskIdentifier` parametro.

Andamento delle esportazioni in Aurora My SQL

Le istantanee del cluster DB Aurora My SQL versione 2 e versione 3 utilizzano un meccanismo di esportazione avanzato per migliorare le prestazioni e ridurre i tempi di esportazione. Il meccanismo include ottimizzazioni come thread di esportazione multipli e Aurora My SQL parallel query per sfruttare l'architettura di archiviazione condivisa Aurora. Le ottimizzazioni vengono applicate in modo adattivo, a seconda delle dimensioni e della struttura del set di dati.

Non è necessario attivare la query parallela per utilizzare il processo di esportazione più rapido, ma il processo presenta le stesse limitazioni della query parallela. Inoltre, alcuni valori di dati non sono supportati, ad esempio le date in cui è il giorno del mese è `0` o l'anno è `0000`. Per ulteriori informazioni, consulta Interrogazione parallela per Amazon Aurora My SQL.

Quando vengono applicate le ottimizzazioni delle prestazioni, è possibile che vengano visualizzati anche file Parquet molto più grandi (~200 GB) per le esportazioni di Aurora My SQL versione 2 e 3.

Se non è possibile utilizzare il processo di esportazione più rapido, ad esempio a causa di tipi di dati o valori incompatibili, Aurora passa automaticamente a una modalità di esportazione a thread singolo senza query parallela. Le prestazioni di esportazione possono variare a seconda del processo utilizzato e della quantità di dati da esportare.

Risoluzione dei problemi relativi all'esportazione di istantanee

Utilizza le seguenti sezioni per risolvere i messaggi di errore e gli errori di SQL autorizzazione Postgre per le attività di esportazione di cluster DB su Amazon S3.

Messaggi di errore per le attività di esportazione di Amazon S3

Nella tabella seguente vengono descritti i messaggi restituiti quando le attività di esportazione di Amazon S3 non riescono.

Messaggio di errore	Descrizione
Si è verificato un errore interno sconosciuto.	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
Si è verificato un errore interno sconosciuto durante la scrittura dei metadati dell'attività di esportazione nel bucket S3 [nome bucket].	L'elaborazione della richiesta non è riuscita a causa di un errore, un'eccezione o un guasto interno sconosciuto.
L'RDSesportazione non è riuscita a scrivere i metadati dell'attività di esportazione perché non può assumere il ruolo [ruolo]. IAM ARN	L'attività di esportazione assume il IAM ruolo dell'utente di verificare se è consentito scrivere metadati nel bucket S3. Se l'attività non può assumere il tuo IAM ruolo, ha esito negativo.
L'RDSesportazione non è riuscita a scrivere i metadati dell'attività di esportazione nel bucket S3 [nome del bucket] utilizzando il IAM ruolo [ruoloARN] con la KMS chiave [ID chiave]. Codice di errore: [codice di errore]	Mancano una o più autorizzazioni, quindi l'attività di esportazione non può accedere al bucket S3. Questo messaggio di errore viene generato quando si riceve uno dei seguenti codici di errore: • `AWSSecurityTokenServiceException` con il codice di errore `AccessDenied` • `AmazonS3Exception` con il codice di erroreNoSuchBucket ,AccessDenied ,KMS.KMSInvalidStateException ,403 Forbidden , oppure KMS.DisabledException Questi codici di errore indicano che le impostazioni non sono configurate correttamente per il IAM ruolo, il bucket S3 o la chiave. KMS
Il IAM ruolo [ruoloARN] non è autorizzato a chiamare [azione S3] sul bucket S3 [nome del bucket]. Controlla le tue autorizzazioni e riprova l'esportazione.	La IAM policy non è configurata correttamente. L'autorizzazione per l'azione S3 specifica sul bucket S3 è mancante e questa condizione causa l'esito negativo dell'attività di esportazione.

Messaggio di errore	Descrizione
KMScontrollo delle chiavi non riuscito. Controlla le credenziali sulla tua KMS chiave e riprova.	Il controllo delle credenziali della KMS chiave non è riuscito.
Controllo delle credenziali S3 non riuscito. Controlla le autorizzazioni sul tuo bucket S3 e sulla tua policy. IAM	Il controllo delle credenziali S3 non è riuscito.
Il bucket S3 [nome bucket] non è valido. Non si trova nella corrente Regione AWS oppure non esiste. Esamina il nome del bucket S3 e riprova l'esporta zione.	Bucket S3 non è valido.
Il bucket S3 [nome bucket] non si trova nella corrente Regione AWS. Esamina il nome del bucket S3 e riprova l'esporta zione.	Il bucket S3 è sbagliato. Regione AWS

Risoluzione degli errori relativi alle autorizzazioni di Postgres SQL

Durante l'esportazione di SQL database Postgre su Amazon S3, potresti visualizzare un `PERMISSIONS_DO_NOT_EXIST` errore che indica che alcune tabelle sono state ignorate. Questo errore si verifica in genere quando l'utente con privilegi avanzati che hai specificato durante la creazione dell'istanza database, non dispone delle autorizzazioni per accedere alle tabelle.

Per risolvere questo errore, eseguire il comando seguente:

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Per ulteriori informazioni sui privilegi utente con privilegi avanzati, vedere Privilegi dell'account utente master.

Ripristino di un cluster di database a un determinato momento

Per ripristinare un cluster di database a un determinato momento, crea un nuovo cluster di database.

Quando ripristini un cluster di database a un determinato momento, puoi scegliere il gruppo di sicurezza VPC (Virtual Private Cloud) predefinito. In alternativa, puoi applicare un gruppo di sicurezza VPC personalizzato al tuo cluster di database.

Il cluster di database ripristinati vengono associati automaticamente al cluster e ai gruppi di parametri predefiniti. Tuttavia, puoi applicare un gruppo di parametri personalizzati specificandoli durante un ripristino.

Amazon Aurora carica i log per i cluster di database in Simple Storage Service (Amazon S3) in modo continuativo. Per visualizzare l'ora di ripristino più recente per un cluster DB, usa il AWS CLI describe-db-clusterscomando e guarda il valore restituito nel `LatestRestorableTime` campo per il cluster DB.

Puoi eseguire il ripristino point-in-time durante il tempo di conservazione del backup. Per visualizzare il primo orario di ripristino per un cluster di DB, usa il AWS CLI describe-db-clusterscomando e guarda il valore restituito nel `EarliestRestorableTime` campo per il cluster di DB.

Il periodo di conservazione dei backup del cluster di database ripristinato è uguale a quello del database di origine.

> **Note**
>
> Le informazioni contenute in questo argomento si applicano ad Amazon Aurora. Per informazioni sul ripristino di un'istanza database Amazon RDS, consulta Ripristino di un'istanza database a un determinato momento.
> Per ulteriori informazioni sul backup e sul ripristino di un cluster di database Aurora, consulta Panoramica di backup e ripristino di un cluster di database Aurora.
> Per Aurora MySQL, puoi ripristinare un cluster di database assegnato a un cluster di database Aurora Serverless. Per ulteriori informazioni, consulta Ripristino di un Aurora Serverless v1 cluster di database.
> Puoi anche utilizzarlo AWS Backup per gestire i backup dei cluster Amazon Aurora DB. Se il cluster DB è associato a un piano di backup in AWS Backup, tale piano di backup viene utilizzato per il ripristino. point-in-time Per informazioni, consultare Ripristino di un cluster DB a un orario specificato utilizzando AWS Backup.

Per informazioni sul ripristino di un cluster Aurora DB o di un cluster globale con una versione RDS Extended Support, vedere. Ripristino di Aurora DB o di un cluster globale con Amazon Extended Support RDS

Ripristina un cluster DB a un'ora specificata da un backup automatico, un backup automatizzato mantenuto o utilizzando. AWS Backup

Argomenti

- Ripristino di un cluster DB in un punto temporale
- Ripristino di un cluster database a un orario specificato da un backup automatico mantenuto
- Ripristino di un cluster DB a un orario specificato utilizzando AWS Backup

Ripristino di un cluster DB in un punto temporale

È possibile ripristinare un cluster DB in un punto temporale utilizzando l'API AWS Management Console RDS o AWS CLI l'API RDS.

Console

Per ripristinare un cluster di database a un determinato momento

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Automated backups (Backup automatici).

   I backup automatici vengono visualizzati nella scheda Current Region (Regione corrente).

3. Scegli il cluster di database che desideri ripristinare.
4. In Actions (Operazioni), scegli Restore to point in time (Ripristina a un istante temporale).

   Viene visualizzata la finestra Restore to point in time (Ripristina a un istante temporale).
5. Scegliere Latest restorable time (Ultimo orario di ripristino) per eseguire il ripristino in base al momento più recente oppure scegliere Custom (Personalizzato) per scegliere una data e un'ora.

Se scegli Custom (Personalizzato), inserisci la data e l'ora alla quale desideri ripristinare il cluster.

> **Note**
>
> Gli orari vengono visualizzati nel fuso orario locale, indicato come un offset dell'ora UTC (Coordinated Universal Time). Ad esempio, UTC-5 è l'orario standard degli Stati Uniti orientali/ora legale degli Stati Uniti centrali.

6. Per Identificatore cluster di database, inserisci il cluster database di destinazione ripristinato. Il nome deve essere univoco.
7. Scegliere le altre opzioni in base alle esigenze, ad esempio la classe dell'istanza database e la configurazione dell'archiviazione del cluster database.

   Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.
8. Scegli Restore to point in time (Ripristina per punto nel tempo).

AWS CLI

Per ripristinare un cluster DB a un orario specificato, utilizzare il AWS CLI comando restore-db-cluster-to- point-in-time to creare un nuovo cluster DB.

È possibile specificare altre impostazioni. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Il tagging di risorse è supportato per questa operazione. Quando usi l'opzione `--tags`, i tag del cluster di database di origine vengono ignorati e vengono utilizzati quelli forniti. In caso contrario, vengono utilizzati i tag più recenti del cluster di origine.

Example

Per LinuxmacOS, oUnix:

```
aws rds restore-db-cluster-to-point-in-time \
       --source-db-cluster-identifier mysourcedbcluster \
       --db-cluster-identifier mytargetdbcluster \
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
       --source-db-cluster-identifier mysourcedbcluster ^
       --db-cluster-identifier mytargetdbcluster ^
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Se utilizzi la console per ripristinare un cluster di database a un determinato momento, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database. Se si utilizza il AWS CLI per ripristinare un cluster DB a un orario specificato, è necessario creare in modo esplicito l'istanza principale per il cluster di DB. L'istanza primaria è la prima istanza creata in un cluster di database.
Per creare l'istanza principale per il tuo cluster DB, chiama il create-db-instance AWS CLI comando. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

API RDS

Per ripristinare un cluster di database a un punto temporale specifico, utilizza l'operazione API Amazon RDS `RestoreDBClusterToPointInTime` con i seguenti parametri:

- `SourceDBClusterIdentifier`
- `DBClusterIdentifier`
- `RestoreToTime`

Important

Se utilizzi la console per ripristinare un cluster di database a un determinato momento, Amazon RDS crea automaticamente l'istanza principale (writer) per il cluster di database.
Se utilizzi l'API RDS per ripristinare un cluster di database a un determinato momento, devi creare in modo esplicito l'istanza principale per il cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Per creare l'istanza principale per il tuo cluster DB, chiama l'operazione DBInstanceCreate dell'API RDS. Includi il nome del cluster di database come valore del parametro `DBClusterIdentifier`.

Ripristino di un cluster database a un orario specificato da un backup automatico mantenuto

È possibile ripristinare un cluster database da un backup automatico mantenuto dopo aver eliminato il cluster database di origine, se il backup rientra nel periodo di conservazione del cluster di origine. Il processo è simile al ripristino di un cluster database da un backup automatico.

Note

Non è possibile ripristinare un cluster Aurora Serverless v1 DB utilizzando questa procedura, poiché i backup automatici per Aurora Serverless v1 i cluster non vengono conservati.

Console

Per ripristinare un cluster di database a un determinato momento

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Automated backups (Backup automatici).
3. Scegli la scheda Conservato.

4. Scegli il cluster di database che desideri ripristinare.
5. In Actions (Operazioni), scegli Restore to point in time (Ripristina a un istante temporale).

   Viene visualizzata la finestra Restore to point in time (Ripristina a un istante temporale).

6. Scegliere Latest restorable time (Ultimo orario di ripristino) per eseguire il ripristino in base al momento più recente oppure scegliere Custom (Personalizzato) per scegliere una data e un'ora.

   Se scegli Custom (Personalizzato), inserisci la data e l'ora alla quale desideri ripristinare il cluster.

   > **Note**
   >
   > Gli orari vengono visualizzati nel fuso orario locale, indicato da una differenza rispetto al Coordinated Universal Time (UTC). Ad esempio, UTC -5 è l'ora solare orientale/ora legale centrale.

7. Per Identificatore cluster di database, inserisci il cluster database di destinazione ripristinato. Il nome deve essere univoco.
8. Scegli altre opzioni in base alle esigenze, ad esempio la classe di istanza database.

   Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.
9. Scegli Restore to point in time (Ripristina per punto nel tempo).

AWS CLI

Per ripristinare un cluster DB a un'ora specificata, utilizzare il AWS CLI comando restore-db-cluster-to- point-in-time per creare un nuovo cluster DB.

È possibile specificare altre impostazioni. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Il tagging di risorse è supportato per questa operazione. Quando usi l'opzione `--tags`, i tag del cluster di database di origine vengono ignorati e vengono utilizzati quelli forniti. In caso contrario, vengono utilizzati i tag più recenti del cluster di origine.

Example

Per LinuxmacOS, oUnix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE \
    --db-cluster-identifier mytargetdbcluster \
```

```
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Per Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE ^
    --db-cluster-identifier mytargetdbcluster ^
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Se utilizzi la console per ripristinare un cluster DB a un orario specificato, Amazon crea RDS automaticamente l'istanza principale (writer) per il tuo cluster DB. Se utilizzi il AWS CLI per ripristinare un cluster DB a un'ora specificata, devi creare in modo esplicito l'istanza principale per il tuo cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Per creare l'istanza principale per il tuo cluster DB, chiama il create-db-instance AWS CLI comando. Includi il nome del cluster di database come valore dell'opzione `--db-cluster-identifier`.

RDS API

Per ripristinare un cluster DB a un'ora specificata, chiama l'RDSAPI`RestoreDBClusterToPointInTime`operazione Amazon con i seguenti parametri:

- `SourceDbClusterResourceId`
- `DBClusterIdentifier`
- `RestoreToTime`

Important

Se utilizzi la console per ripristinare un cluster DB a un orario specificato, Amazon crea RDS automaticamente l'istanza principale (writer) per il tuo cluster DB. Se utilizzi il RDS API per ripristinare un cluster DB a un'ora specificata, assicurati di creare in modo esplicito l'istanza principale per il tuo cluster di database. L'istanza primaria è la prima istanza creata in un cluster di database.

Per creare l'istanza principale per il tuo cluster DB, chiama l'RDSAPIoperazione C. reateDBInstance Includi il nome del cluster di database come valore del parametro `DBClusterIdentifier`.

Ripristino di un cluster DB a un orario specificato utilizzando AWS Backup

È possibile AWS Backup utilizzarlo per gestire i backup automatici e quindi ripristinarli a un'ora specificata. A tale scopo, è necessario creare un piano di backup in AWS Backup e assegnare il cluster DB come risorsa. Quindi abiliti i backup continui PITR nella regola di backup. Per ulteriori informazioni sui piani di backup e sulle regole di backup, consulta laGuida per gli sviluppatori di AWS Backup.

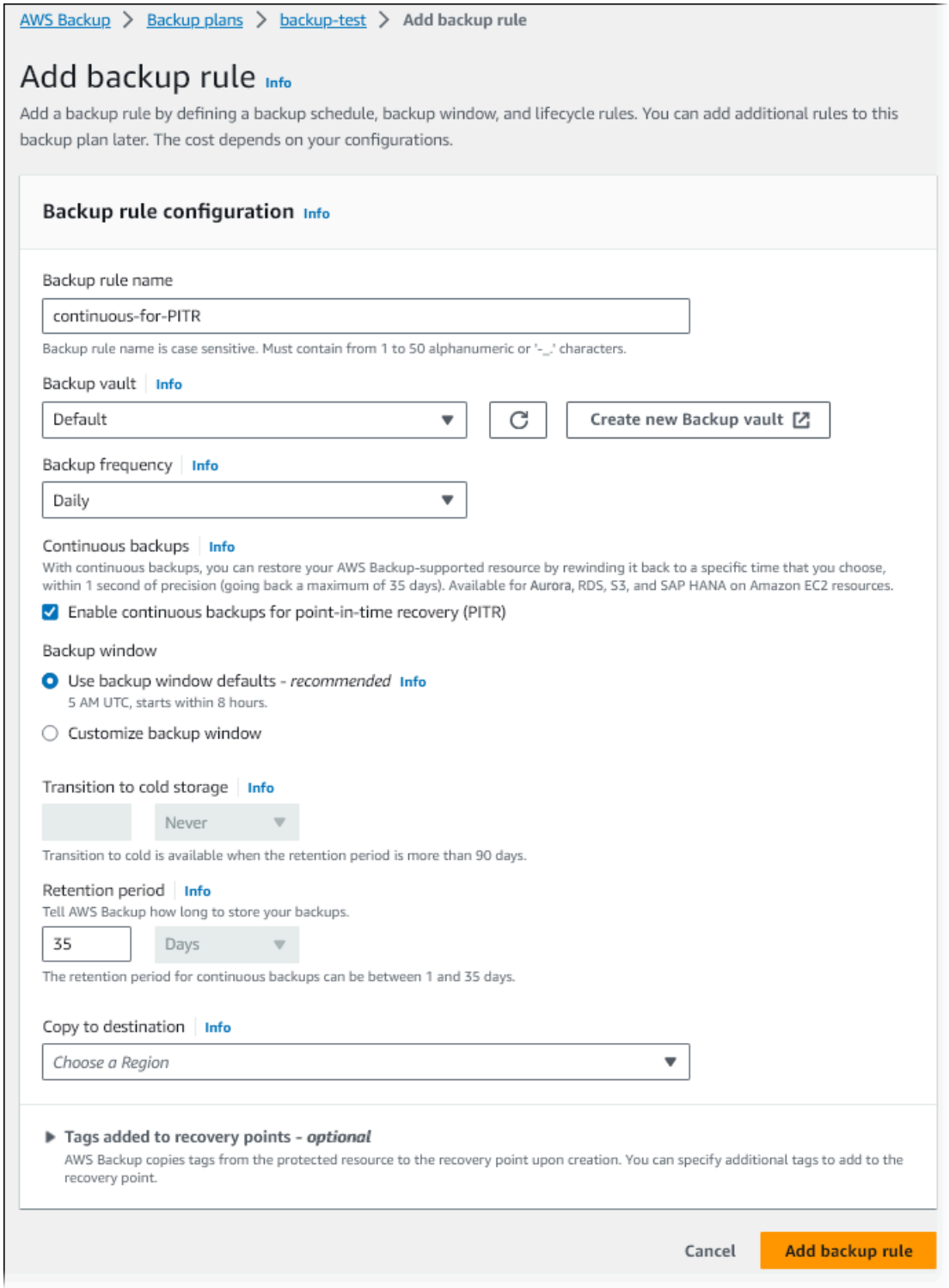
Abilitazione dei backup continui in AWS Backup

È possibile abilitare i backup continui nelle regole di backup.

Per abilitare i backup continui per PITR

1. Accedere a e aprire AWS Management Console la AWS Backup console all'indirizzo https://console.aws.amazon.com/backup.
2. Nel riquadro di navigazione scegliere Backup plans (Piani di backup).
3. In Nome del piano di backup, seleziona il piano di backup che utilizzi per eseguire il backup del cluster database.
4. Nella sezione Regole di backup, scegli Aggiungi regola di backup.

   Viene visualizzata la pagina Aggiungi regola di backup.
5. Seleziona la casella di controllo Abilita backup continui per point-in-time il ripristino (PITR).



6. Scegli le altre impostazioni, se necessario, quindi scegli Aggiungi regola di backup.

Ripristino da un backup continuo in AWS Backup

Puoi eseguire il ripristino a un'ora specificata da un vault di backup.

Console

È possibile utilizzare il AWS Management Console per ripristinare un cluster DB a un'ora specificata.

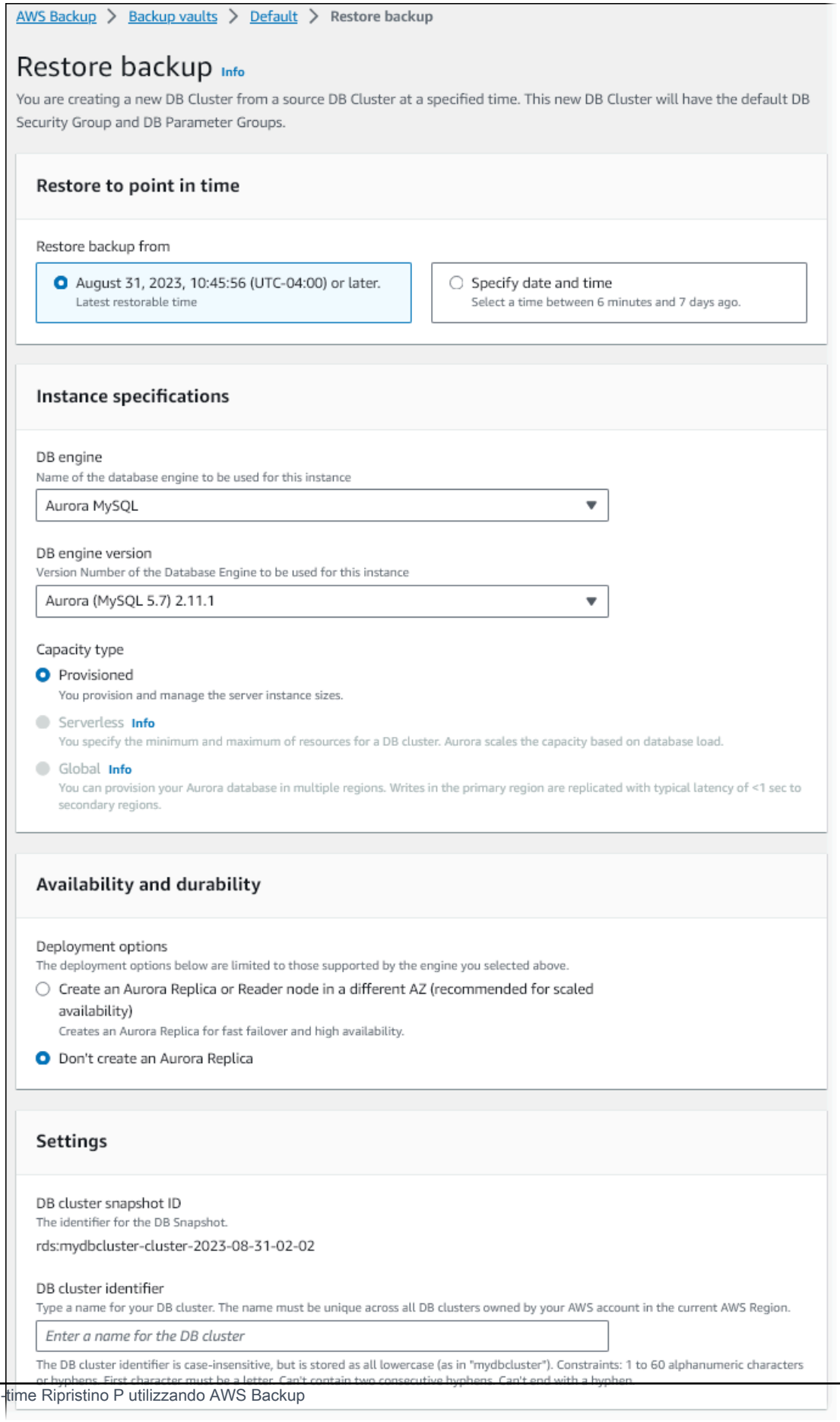
Per eseguire il ripristino da un backup continuo in AWS Backup

1. Accedere a e aprire la AWS Backup console all'indirizzo https://console.aws.amazon.com/backup. AWS Management Console
2. Nel riquadro di navigazione scegliere Backup vaults (Vault di backup).
3. Scegli il vault di backup contenente il backup continuo, ad esempio Predefinito.

   Viene visualizzata la pagina dei dettagli del vault di backup.
4. In Punti di ripristino, seleziona il punto di ripristino per il backup automatico.

   Il tipo di backup è Continuo e il nome e `continuous:cluster-`*`AWS-Backup-job-number`*.
5. In Operazioni, sceglie Riavvia.

   Viene visualizzata la pagina Ripristina backup.


AWS Backup > Backup vaults > Default > Restore backup
Restore backup Info
You are creating a new DB Cluster from a source DB Cluster at a specified time. This new DB Cluster will have the default DB Security Group and DB Parameter Groups.
Restore to point in time
Restore backup from
August 31, 2023, 10:45:56 (UTC-04:00) or later.
Latest restorable time
Specify date and time
Select a time between 6 minutes and 7 days ago.
Instance specifications
DB engine
Name of the database engine to be used for this instance
Aurora MySQL
DB engine version
Version Number of the Database Engine to be used for this instance
Aurora (MySQL 5.7) 2.11.1
Capacity type
Provisioned
You provision and manage the server instance sizes.
Serverless Info
You specify the minimum and maximum of resources for a DB cluster. Aurora scales the capacity based on database load.
Global Info
You can provision your Aurora database in multiple regions. Writes in the primary region are replicated with typical latency of <1 sec to secondary regions.
Availability and durability
Deployment options
The deployment options below are limited to those supported by the engine you selected above.
Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.
Don't create an Aurora Replica
Settings
DB cluster snapshot ID
The identifier for the DB Snapshot.
rds:mydbcluster-cluster-2023-08-31-02-02
DB cluster identifier
Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.
Enter a name for the DB cluster
The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

6. PerRipristina al point-in-time, seleziona Specifica data e ora per eseguire il ripristino a un'ora specifica.
7. Scegli le altre impostazioni necessarie per ripristinare il cluster database, quindi scegli Ripristina backup.

   Viene visualizzata la pagina Processi con il riquadro Processi di ripristino. Un messaggio nella parte superiore della pagina fornisce informazioni sul lavoro di ripristino.

Dopo il ripristino del cluster database, devi aggiungere l'istanza database (scrittura) primaria. Per creare l'istanza principale per il tuo cluster DB, chiama il create-db-instance AWS CLI comando. Includi il nome del cluster di database come valore del parametro `--db-cluster-identifier`.

CLI

Il start-restore-job AWS CLI comando viene utilizzato per ripristinare il cluster DB a un'ora specificata. I parametri seguenti sono obbligatori:

- `--recovery-point-arn`— Amazon Resource Name (ARN) per il punto di ripristino da cui eseguire il ripristino.
- `--resource-type`: utilizza `Aurora`.
- `--iam-role-arn`— Il ARN per il IAM ruolo che utilizzi per AWS Backup le operazioni.
- `--metadata`: i metadati utilizzati per ripristinare il cluster database. I parametri seguenti sono obbligatori:
  - `DBClusterIdentifier`
  - `Engine`
  - `RestoreToTime` o `UseLatestRestorableTime`

L'esempio seguente mostra come ripristinare un cluster database a un'ora specificata.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
--metadata '{"DBClusterIdentifier":"backup-pitr-test","Engine":"aurora-
mysql","RestoreToTime":"2023-09-01T17:00:00.000Z"}'
```

L'esempio seguente mostra come ripristinare un cluster database all'ultima ora ripristinabile.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
--metadata '{"DBClusterIdentifier":"backup-pitr-latest","Engine":"aurora-
mysql","UseLatestRestorableTime":"true"}'
```

Dopo il ripristino del cluster database, devi aggiungere l'istanza database (scrittura) primaria. Per creare l'istanza principale per il tuo cluster DB, chiama il create-db-instance AWS CLI comando. Includi il nome del cluster di database come valore del parametro `--db-cluster-identifier`.

Eliminazione di una snapshot del cluster di database

Puoi eliminare gli snapshot dei cluster DB gestiti da Amazon RDS quando non ti servono più.

> **Note**
>
> Per eliminare i backup gestiti da AWS Backup, usa la AWS Backup console. Per ulteriori informazioni su AWS Backup, consulta la Guida per sviluppatori di AWS Backup .

Eliminazione di una snapshot del cluster di database

È possibile eliminare un'istantanea del cluster DB utilizzando la console, il AWS CLI, o il. RDS API

Per eliminare uno snapshot condiviso o pubblico, è necessario accedere all' AWS account proprietario dello snapshot.

Console

Per eliminare una snapshot del cluster di database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere la snapshot del cluster di database da eliminare.
4. Per Actions (Operazioni), scegliere Delete Snapshot (Elimina snapshot).
5. Nella pagina di conferma, scegliere Delete (Elimina).

AWS CLI

È possibile eliminare uno snapshot del cluster DB utilizzando il AWS CLI comando delete-db-cluster-snapshot.

Le seguenti opzioni vengono utilizzate per eliminare una snapshot del cluster di database.

- `--db-cluster-snapshot-identifier` – L'identificatore per la snapshot del cluster di database.

Example

Il seguente codice elimina la snapshot del cluster di database `mydbclustersnapshot`.

Per LinuxmacOS, oUnix:

```
aws rds delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Per Windows:

```
aws rds delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

RDS API

Puoi eliminare uno snapshot del cluster DB utilizzando lo snapshot D eleteDBCluster di Amazon RDS API Operation.

I seguenti parametri vengono utilizzati per eliminare uno snapshot del cluster di database.

- `DBClusterSnapshotIdentifier` – L'identificatore per la snapshot del cluster di database.

Tutorial: Ripristino di un cluster database Amazon Autora da uno snapshot cluster DB

Quando si utilizza Amazon Aurora è normale avere un'istanza database utilizzata occasionalmente ma che non serve a tempo pieno. Ad esempio, potrebbe essere necessario utilizzare un cluster database per conservare i dati di un report che viene eseguito solo trimestralmente. Per risparmiare denaro su tale scenario, è possibile acquisire uno snapshot cluster DB del cluster database al termine del report. Eliminare quindi il cluster database e ripristinarlo quando è necessario caricare nuovi dati ed eseguire il report durante il trimestre successivo.

Quando si ripristina un cluster database, si fornisce il nome dello snapshot cluster DB da cui eseguire il ripristino. Quindi si fornisce un nome per il nuovo cluster database creato dall'operazione di ripristino. Per ulteriori informazioni sul ripristino di cluster database da snapshot, consulta Ripristino da uno snapshot cluster database.

In questo tutorial, aggiorniamo anche il cluster DB ripristinato da Aurora My SQL versione 2 (compatibile con My SQL 5.7) ad Aurora My SQL versione 3 (compatibile con My 8.0). SQL

Ripristina un cluster DB a un'ora specificata da uno snapshot del cluster DB utilizzando la RDS console Amazon o il AWS CLI.

Per informazioni su AWS KMS gestione delle chiavi per AmazonRDS, vediAWS KMS key gestione.

Argomenti

- Tutorial: ripristino di un cluster DB da uno snapshot del cluster DB utilizzando la console Amazon RDS
- Tutorial: ripristino di un cluster DB da uno snapshot del cluster DB utilizzando AWS CLI

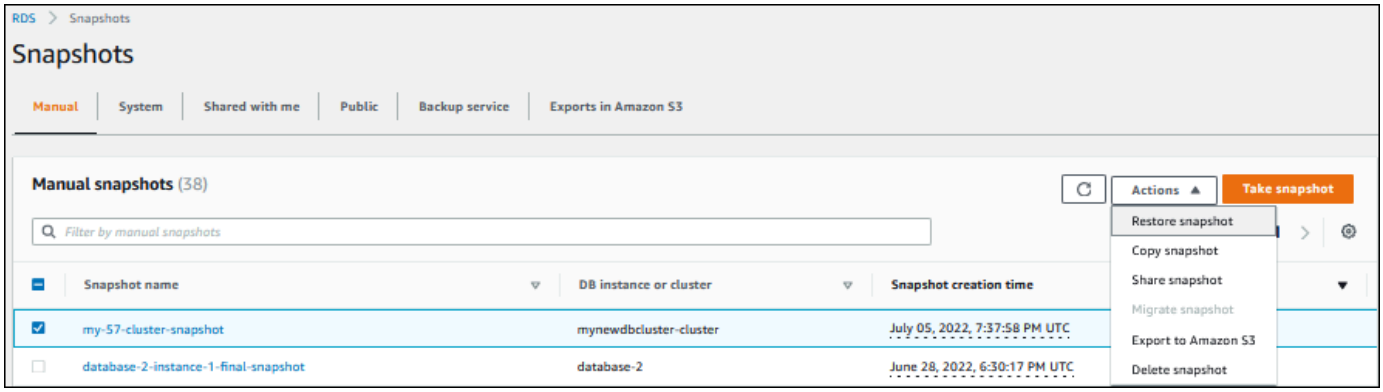
Tutorial: ripristino di un cluster DB da uno snapshot del cluster DB utilizzando la console Amazon RDS

In questo tutorial, ripristini un cluster DB da uno snapshot del cluster DB utilizzando la console Amazon RDS. Quando ripristini un cluster DB da uno snapshot utilizzando AWS Management Console, viene creata anche l'istanza DB primaria (writer).

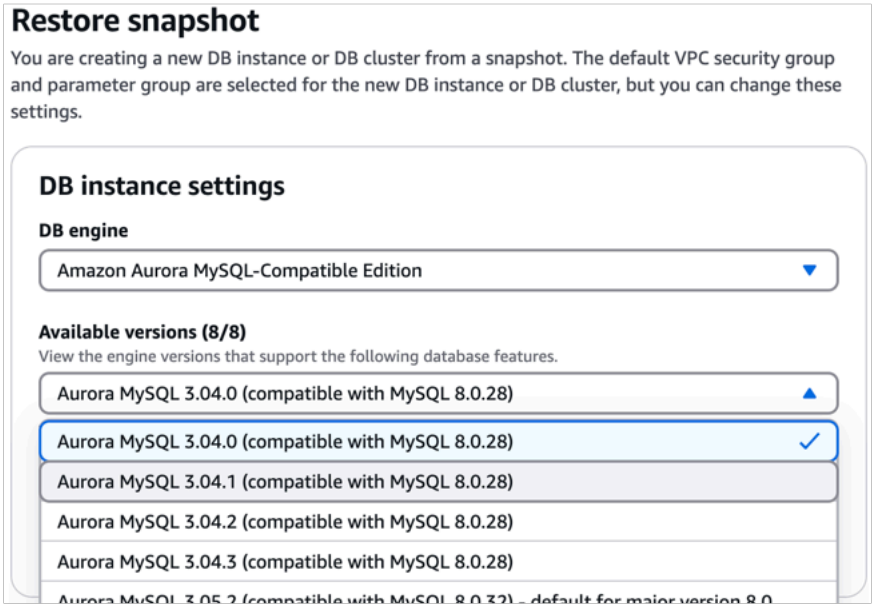
> **Note**
>
> Mentre l'istanza database primaria è in corso di creazione, viene visualizzata come istanza di lettura, ma al termine è un'istanza di scrittura.

Per ripristinare un cluster database da una snapshot cluster DB

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, selezionare Snapshots (Snapshot).
3. Scegliere lo snapshot cluster database da cui eseguire il ripristino.
4. Per Actions (Operazioni), selezionare Restore Snapshot (Ripristina snapshot).



   Viene visualizzata la pagina Restore snapshot (Ripristina snapshot).
5. In DB instance settings (Impostazioni dell'istanza database), effettua le seguenti operazioni:
   a. Usa l'impostazione predefinita per DB engine (Motore database).
   b. Per le versioni disponibili, scegli una versione compatibile con MySQL 8.0, come Aurora MySQL 3.04.0 (compatibile con MySQL 8.0.28).



6. Sotto Settings (Impostazioni), per DB instance identifier (Identificatore istanze DB) inserisci il nome univoco da usare per l'istanza database ripristinata, ad esempio **`my-80`**.

   > **Note**
   >
   > Per creare l'identificatore del cluster DB, Amazon RDS lo aggiunge `-cluster` all'identificatore dell'istanza DB specificato.

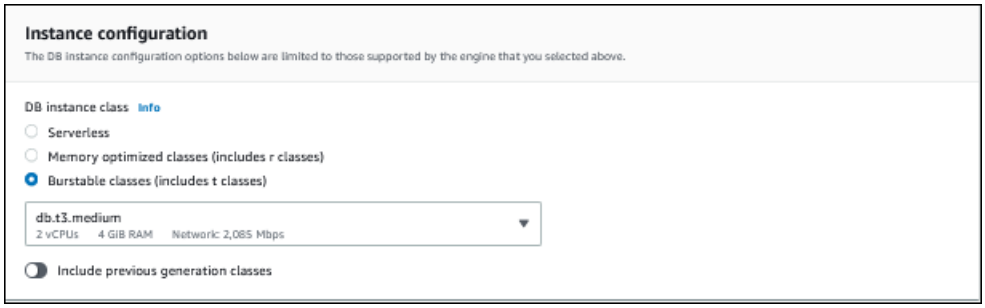
7. Sotto Connectivity (Connettività), utilizza le impostazioni di default per quanto segue:

   - Virtual Private Cloud (VPC) (Cloud privato virtuale (VPC)
   - DB subnet group (Gruppo di sottoreti DB)
   - Accesso pubblico
   - VPC security group (firewall) (Gruppo di sicurezza VPC (firewall)

8. Scegli la DB instance class (Classe di istanza database).

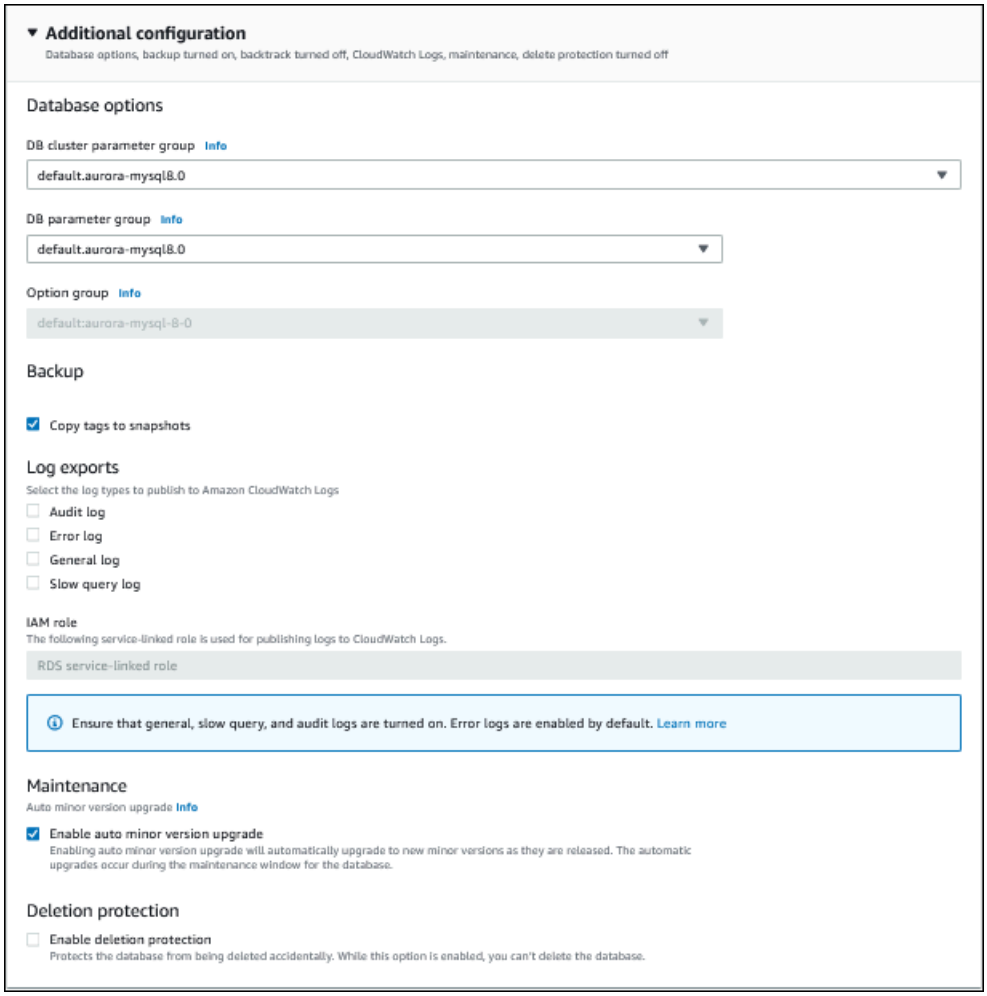
   Per questo tutorial, scegli Burstable classes (includes t classes) (Classi espandibili (include le classi t) e quindi seleziona db.t3.medium.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.



9. Per Database authentication (Autenticazione database), utilizza l'impostazione predefinita.
10. Per Encryption (Crittografia), utilizza le impostazioni di default.

    Se il cluster database di origine per lo snapshot è stato crittografato, anche il cluster database ripristinato viene crittografato. Non è possibile renderla non crittografato.
11. Espandi Additional configuration (Configurazione aggiuntiva) nella parte inferiore della pagina.



12. Effettua le scelte riportate di seguito:

    a. Per questo tutorial, utilizza il valore predefinito per DB cluster parameter group (Gruppo di parametri del cluster DB).

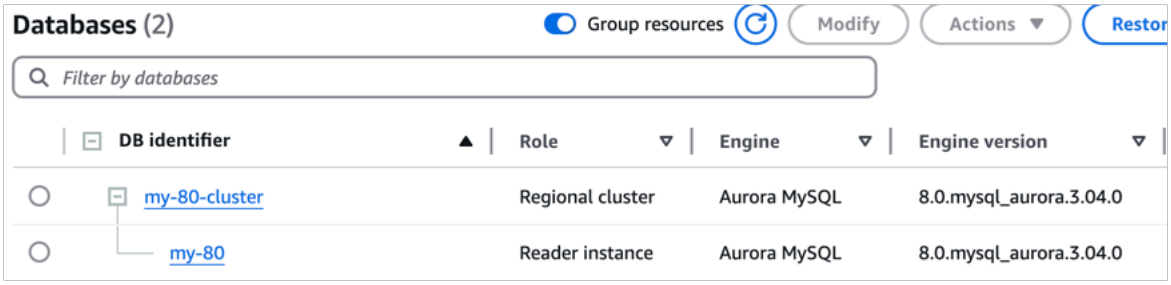
    b. Per questo tutorial, utilizza il valore predefinito per DB parameter group (Gruppo di parametri DB).

    c. Per Log exports (Esportazioni log), seleziona tutte le caselle di controllo.

    d. Per Deletion protection (Protezione da eliminazione), seleziona la casella di controllo Enable deletion protection (Abilita protezione da eliminazione).

13. Selezionare Ripristina istanza database.

La pagina Databases (Database) visualizza il cluster database ripristinato, con uno stato `Creating`.



Mentre l'istanza database primaria è in corso di creazione, viene visualizzata come istanza di lettura, ma al termine è un'istanza di scrittura.

Tutorial: ripristino di un cluster DB da uno snapshot del cluster DB utilizzando AWS CLI

In questo tutorial, ripristini un cluster DB da uno snapshot del cluster DB utilizzando il. AWS CLI Il ripristino di un cluster DB da un'istantanea utilizzando il AWS CLI prevede due passaggi:

1. Ripristino del cluster databaseutilizzando il comando restore-db-cluster-from -snapshot
2. Creazione dell'istanza database (scrittura) primariautilizzando il comando create-db-instance

Ripristino del cluster database

Si utilizza il comando `restore-db-cluster-from-snapshot`. Sono richieste le seguenti opzioni:

- `--db-cluster-identifier`: il nome del cluster database ripristinato.
- `--snapshot-identifier`: il nome dello snapshot DB da cui eseguire il ripristino.
- `--engine`: il motore di database del cluster database ripristinato. Deve essere compatibile con il motore di database del cluster database di origine.

  Le scelte sono le seguenti:

  - `aurora-mysql`— Compatibile con Aurora My SQL 5.7 e 8.0.
  - `aurora-postgresql`— Compatibile con Aurora SQL Postgre.

  In questo esempio viene utilizzato `aurora-mysql`.
- `--engine-version`: la versione del cluster database ripristinato. In questo esempio, utilizziamo una versione compatibile con My SQL -8.0.

L'esempio seguente ripristina un cluster DB compatibile con Aurora SQL My 8.0 `my-new-80-cluster` denominato da uno snapshot del cluster DB denominato. `my-57-cluster-snapshot`

Per ripristinare il cluster database

- Utilizzare uno dei seguenti comandi.

  macOSPer, oLinux: Unix

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-80-cluster \
    --snapshot-identifier my-57-cluster-snapshot \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.02.0
```

  Per Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier my-new-80-cluster ^
    --snapshot-identifier my-57-cluster-snapshot ^
    --engine aurora-mysql ^
    --engine-version 8.0.mysql_aurora.3.02.0
```

L'output è simile a quello riportato di seguito.

```
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "eu-central-1b",
            "eu-central-1c",
            "eu-central-1a"
        ],
        "BackupRetentionPeriod": 14,
        "DatabaseName": "",
        "DBClusterIdentifier": "my-new-80-cluster",
        "DBClusterParameterGroup": "default.aurora-mysql8.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "my-new-80-cluster.cluster-############.eu-
central-1.rds.amazonaws.com",
```

```
        "ReaderEndpoint": "my-new-80-cluster.cluster-ro-############.eu-
central-1.rds.amazonaws.com",
        "MultiAZ": false,
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "Port": 3306,
        "MasterUsername": "admin",
        "PreferredBackupWindow": "01:55-02:25",
        "PreferredMaintenanceWindow": "thu:21:14-thu:21:44",
        "ReadReplicaIdentifiers": [],
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "HostedZoneId": "Z1RLNU0EXAMPLE",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:123456789012:key/#######-5ccc-49cc-8aaa-
############",
        "DbClusterResourceId": "cluster-ZZ12345678ITSJUSTANEXAMPLE",
        "DBClusterArn": "arn:aws:rds:eu-central-1:123456789012:cluster:my-new-80-
cluster",
        "AssociatedRoles": [],
        "IAMDatabaseAuthenticationEnabled": false,
        "ClusterCreateTime": "2022-07-05T20:45:42.171000+00:00",
        "EngineMode": "provisioned",
        "DeletionProtection": false,
        "HttpEndpointEnabled": false,
        "CopyTagsToSnapshot": false,
        "CrossAccountClone": false,
        "DomainMemberships": [],
        "TagList": []
    }
}
```

Creazione dell'istanza database (scrittura) primaria

Per creare l'istanza database (scrittura) primaria, si utilizza il comando `create-db-instance`. Sono richieste le seguenti opzioni:

- `--db-cluster-identifier`: il nome del cluster database ripristinato.

- `--db-instance-identifier`: il nome dell'istanza database primaria.
- `--db-instance-class`: la classe istanza dell'istanza database primaria. In questo esempio viene utilizzato `db.t3.medium`.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.

- `--engine`- il motore di database dell'istanza database primaria. Deve essere identico a quello utilizzato dal cluster database ripristinato.

  Le scelte sono le seguenti:

  - `aurora-mysql`— Compatibile con Aurora My SQL 5.7 e 8.0.
  - `aurora-postgresql`— Compatibile con Aurora SQL Postgre.

  In questo esempio viene utilizzato `aurora-mysql`.

L'esempio seguente crea un'istanza DB primaria (writer) denominata `my-new-80-cluster-instance` nel cluster DB ripristinato compatibile con Aurora My SQL 8.0 denominato. `my-new-80-cluster`

Per creare l'istanza database primaria

- Utilizzare uno dei seguenti comandi.

  PerLinux, o: macOS Unix

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-80-cluster \
    --db-instance-identifier my-new-80-cluster-instance \
    --db-instance-class db.t3.medium \
    --engine aurora-mysql
```

  Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-80-cluster ^
```

```
    --db-instance-identifier my-new-80-cluster-instance ^
    --db-instance-class db.t3.medium ^
    --engine aurora-mysql
```

L'output è simile a quello riportato di seguito.

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-new-80-cluster-instance",
        "DBInstanceClass": "db.t3.medium",
        "Engine": "aurora-mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 1,
        "PreferredBackupWindow": "01:55-02:25",
        "BackupRetentionPeriod": 14,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.aurora-mysql8.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2305ca49",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1a"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
```

```
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1b"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1c"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sat:02:41-sat:03:11",
        "PendingModifiedValues": {},
        "MultiAZ": false,
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "general-public-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:aurora-mysql-8-0",
                "Status": "in-sync"
            }
        ],
        "PubliclyAccessible": false,
        "StorageType": "aurora",
        "DbInstancePort": 0,
        "DBClusterIdentifier": "my-new-80-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:534026745191:key/#######-5ccc-49cc-8aaa-
############",
        "DbiResourceId": "db-5C6UT5PU0YETANOTHEREXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
```

```
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:eu-central-1:123456789012:db:my-new-80-cluster-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": [],
        "TagList": []
    }
}
```

Monitoraggio dei parametri in un cluster di database Amazon Aurora

Amazon Aurora utilizza un cluster di server di database replicati. Il monitoraggio di un cluster Aurora in genere richiede il controllo dell'integrità di più istanze database. Le istanze potrebbero avere ruoli specializzati, gestendo principalmente operazioni di scrittura, solo operazioni di lettura o una combinazione di entrambe. È inoltre possibile monitorare lo stato generale del cluster misurando il ritardo di replica. Questa è la quantità di tempo per le modifiche apportate da un'istanza database per essere disponibili per le altre istanze.

Argomenti

- Piano di monitoraggio
- Baseline delle prestazioni
- Linee guida per le prestazioni
- Strumenti di monitoraggio per Aurora
- Visualizzazione dello stato dell' del cluster
-
- Visualizzazione delle metriche nella console Amazon RDS
- Visualizzazione delle metriche combinate con la dashboard Performance Insights
- Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch
- Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch
- Monitoraggio del carico DB con Performance Insights su Amazon Aurora
- Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS
- Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato
- Riferimento alle metriche per

Piano di monitoraggio

Prima di iniziare a monitorare Amazon Aurora, crea un piano di monitoraggio. Questo piano deve rispondere alle domande seguenti:

- Quali sono gli obiettivi del monitoraggio?

- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno usati?
- Chi eseguirà le attività di monitoraggio?
- Chi deve ricevere la notifica quando si verifica un problema?

Baseline delle prestazioni

Per raggiungere gli obiettivi di monitoraggio è necessario stabilire una baseline. A tale scopo, misura le prestazioni in condizioni di carico diverse in momenti diversi nel tuo Amazon Aurora. Puoi monitorare parametri come i seguenti:

- Throughput di rete
- Connessioni client
- I/O per operazioni di lettura, scrittura o metadati
- Saldi credito burst per le istanze database

Ti consigliamo di archiviare i dati storici sulle prestazioni per Amazon Aurora. Utilizzando i dati archiviati puoi confrontare le prestazioni correnti con le tendenze passate. Puoi distinguere i normali modelli di prestazioni dalle anomalie e definire i metodi per risolvere i problemi.

Linee guida per le prestazioni

In generale, i valori accettabili per i parametri delle prestazioni dipendono dalle attività dell'applicazione in relazione alla tua baseline. Indagare le variazioni della baseline coerenti o che rappresentano dei trend. I seguenti parametri sono spesso fonte di problemi di prestazioni:

- Alto CPU o RAM consumo: valori CPU o RAM consumi elevati potrebbero essere appropriati, se sono in linea con gli obiettivi dell'applicazione (ad esempio produttività o concorrenza) e sono previsti.
- Consumo dello spazio su disco: esamina il consumo dello spazio su disco se lo spazio usato supera costantemente l'85% dello spazio su disco totale. Verifica se è possibile eliminare dati dall'istanza o archiviare dati su un sistema diverso per liberare spazio.

- Traffico di rete – Per il traffico di rete, rivolgiti al tuo amministratore di sistema per identificare il throughput previsto per la rete del dominio e la connessione Internet. Indaga il traffico di rete se il throughput è costantemente al di sotto del valore previsto.
- Connessioni al database – Se noti un numero elevato di connessioni utente insieme a un peggioramento delle prestazioni e del tempo di risposta dell'istanza, valuta se limitare le connessioni al database . Il numero ideale di connessioni utente per l'istanza database dipende dalla classe di istanza e dalla complessità delle operazioni eseguite. Per determinare il numero di connessioni di database, associa l'istanza database a un gruppo di parametri dove il parametro `User Connections` è impostato su un valore diverso da 0 (illimitato). Puoi utilizzare un gruppo di parametri esistente o crearne uno nuovo. Per ulteriori informazioni, consulta .
- IOPSmetriche: i valori previsti per le IOPS metriche dipendono dalle specifiche del disco e dalla configurazione del server, quindi utilizza la linea di base per conoscere le caratteristiche tipiche. Verifica se i valori sono costantemente diversi dalla baseline. Per IOPS prestazioni ottimali, assicurati che il tuo set di lavoro tipico si adatti alla memoria per ridurre al minimo le operazioni di lettura e scrittura.

Quando le prestazioni non rientrano nella baseline stabilita, potrebbe essere necessario apportare modifiche per ottimizzare la disponibilità del database per il carico di lavoro. Ad esempio, potrebbe essere necessario modificare la classe di istanza dell'istanza database. In alternativa, potrebbe essere necessario modificare il numero di istanze database e leggere le repliche disponibili per i client.

Strumenti di monitoraggio per Aurora

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon Aurora e di altri AWS soluzioni. AWS fornisce diversi strumenti di monitoraggio per guardare Amazon Aurora, segnalare quando qualcosa non va e intraprendere azioni automatiche quando necessario.

Argomenti

- Strumenti di monitoraggio automatici
- Strumenti di monitoraggio manuali

Strumenti di monitoraggio automatici

Si consiglia di automatizzare il più possibile i processi di monitoraggio.

Argomenti

- : stato e consigli del cluster Amazon Aurora
- CloudWatch Parametri Amazon per
- Amazon RDS Performance Insights e monitoraggio del sistema operativo
- Servizi integrati

: stato e consigli del cluster Amazon Aurora

Puoi utilizzare i seguenti strumenti automatici per guardare Aurora e segnalare quando qualcosa non va:

- Stato del cluster di Aurora: visualizza i dettagli sullo stato attuale del cluster di utilizzando la RDS console Amazon, AWS CLI, o il. RDS API
- Raccomandazioni di RDS Amazon Aurora: rispondi ai consigli automatici per le risorse di database, come istanze DB, cluster DB, e gruppi di parametri del cluster DB. Per ulteriori informazioni, consulta .

CloudWatch Parametri Amazon per

Aurora si integra con Amazon CloudWatch per funzionalità di monitoraggio aggiuntive.

- Amazon CloudWatch: questo servizio monitora il tuo AWS le risorse e le applicazioni su cui esegui AWS in tempo reale. Puoi utilizzare le seguenti CloudWatch funzionalità di Amazon con Aurora:
  - CloudWatch Parametri Amazon Aurora invia automaticamente i parametri ogni minuto CloudWatch per ogni database attivo. Non sono previsti costi aggiuntivi per i RDS parametri di Amazon in CloudWatch. Per ulteriori informazioni, consulta CloudWatch Parametri Amazon per Amazon Aurora
  - CloudWatch Allarmi Amazon: puoi controllare una singola metrica di Amazon Aurora per un periodo di tempo specifico. È quindi possibile eseguire una o più operazioni in base al valore del parametro rispetto a una soglia impostata.

Amazon RDS Performance Insights e monitoraggio del sistema operativo

Puoi utilizzare i seguenti strumenti automatizzati per monitorare le prestazioni di Amazon Aurora:

- Amazon RDS Performance Insights: valuta il carico sul tuo database e determina quando e dove agire. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- Amazon RDS Enhanced Monitoring: esamina i parametri in tempo reale per il sistema operativo. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Servizi integrati

I seguenti AWS i servizi sono integrati con Amazon Aurora:

- Amazon EventBridge è un servizio di bus eventi senza server che semplifica la connessione delle applicazioni con dati provenienti da una varietà di fonti. Per ulteriori informazioni, consulta .
- Amazon CloudWatch Logs ti consente di monitorare, archiviare e accedere ai tuoi file di log da istanze Aurora e altre CloudTrail fonti. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Aurora.
- AWS CloudTrailregistra le API chiamate e gli eventi correlati effettuati da o per conto del Account AWS e consegna i file di log a un bucket Amazon S3 da te specificato. Per ulteriori informazioni, consulta Monitoraggio delle chiamate API Aurora in AWS CloudTrail.
- Database Activity Streams Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

- DevOpsGuru for RDS è una funzionalità di Amazon DevOps Guru che applica l'apprendimento automatico ai parametri di Performance Insights per i database Amazon Aurora. Per ulteriori informazioni, consulta Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS.

Strumenti di monitoraggio manuali

È necessario monitorare manualmente gli elementi non coperti dagli CloudWatch allarmi. L'Amazzonia RDS CloudWatch, AWS Trusted Advisor e altro AWS le dashboard della console forniscono una at-a-glance visione dello stato del AWS ambiente. Consigliamo anche di controllare i file di log nell'istanza database.

- Dalla RDS console Amazon, puoi monitorare i seguenti elementi per le tue risorse:
  - Il numero di connessioni a un'istanza database
  - Il numero di operazioni di lettura e scrittura a un'istanza database
  - Quantità di storage utilizzato al momento dall'istanza database
  - La quantità di memoria e CPU l'utilizzo per un'istanza DB
  - La quantità di traffico di rete verso e da un'istanza database
- Dal Trusted Advisor dalla dashboard, puoi esaminare i seguenti controlli di ottimizzazione dei costi, sicurezza, tolleranza agli errori e miglioramento delle prestazioni:
  - Istanze database Amazon RDS Idle
  - Rischio di accesso ad Amazon RDS Security Group
  - RDSBackup Amazon
  - Amazon RDS Multi-AZ
  - Accessibilità dell'istanza database Aurora

  Per ulteriori informazioni su questi controlli, consulta best practice Trusted Advisor (Controlli).
- CloudWatch la home page mostra:
  - Stato e allarmi attuali
  - Grafici degli allarmi e delle risorse
  - Stato di integrità dei servizi

  Inoltre, è possibile utilizzare CloudWatch per effettuare le seguenti operazioni:
  - Crea pannelli di controllo personalizzati per monitorare i servizi rilevanti.

- Crea grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
- Cerca e sfoglia tutti i tuoi AWS metriche delle risorse.
- Crea e modifica gli allarmi per ricevere le notifiche dei problemi.

Visualizzazione dello stato dell' del cluster

Utilizzando la console Amazon RDS, puoi accedere rapidamente allo stato della tua del cluster DB.

Argomenti

- Visualizzazione di un cluster di database Amazon Aurora
- Visualizzazione dello stato del cluster del DB
- Visualizzazione dello stato dell'istanza database di in un cluster Aurora

Visualizzazione di un cluster di database Amazon Aurora

Hai diverse opzioni per visualizzare le informazioni sui tuoi cluster di database Amazon Aurora e sulle istanze database nei tuoi cluster di database.

- Puoi visualizzare i cluster di database e le istanze database nella console Amazon RDS scegliendo Databases (Database) dal riquadro di navigazione.
- È possibile ottenere informazioni su cluster e istanze DB utilizzando (). AWS Command Line Interface AWS CLI
- Puoi ottenere informazioni sul cluster di database e le istanze database utilizzando l'API Amazon RDS.

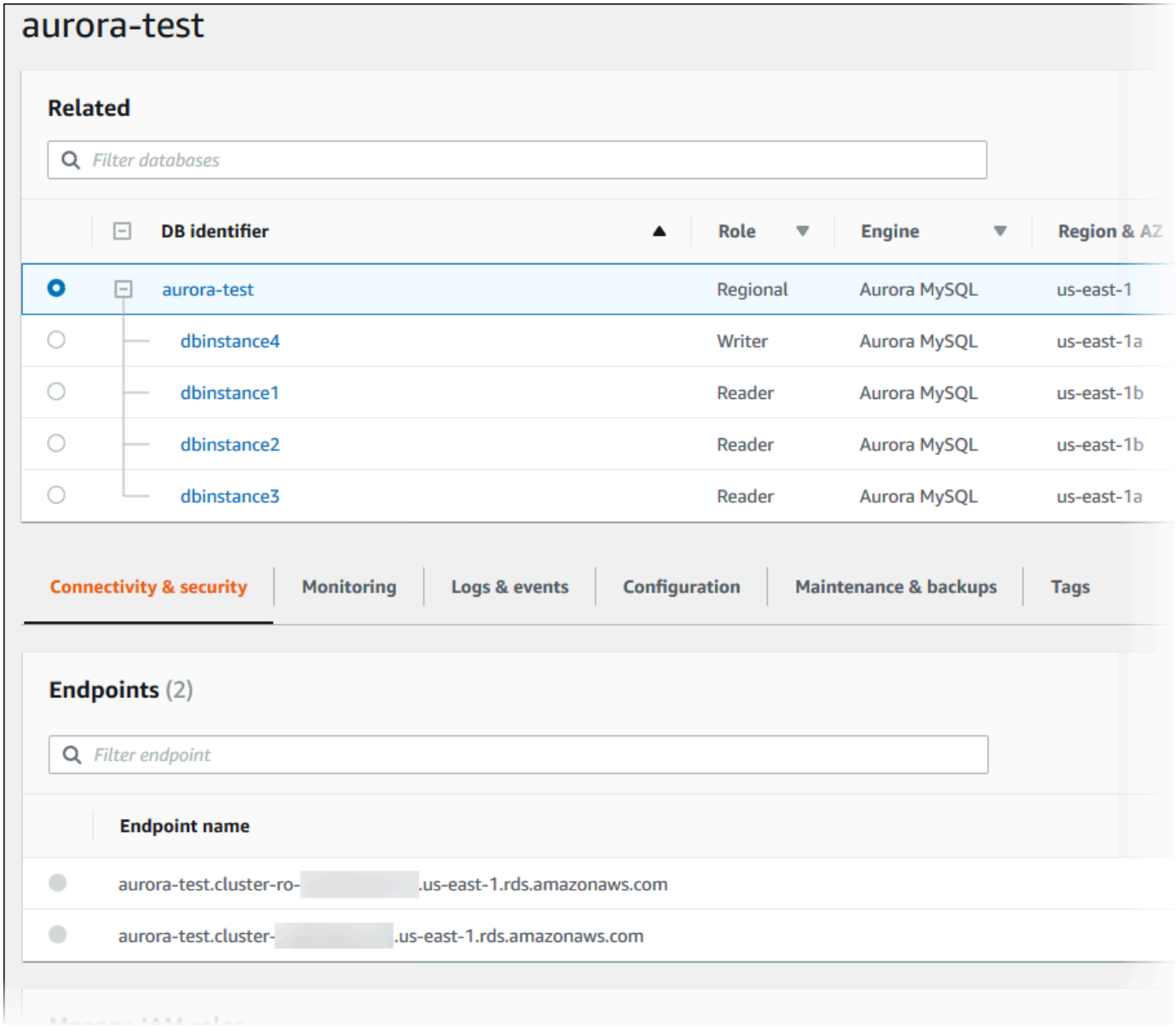
Console

Nella Amazon RDS console, puoi vedere i dettagli di un cluster di dB scegliendo Database dal riquadro di navigazione della console. Puoi anche vedere i dettagli delle istanze database che fanno parte di un cluster di database Amazon Aurora.

Per visualizzare o modificare i cluster DB nella console di Amazon RDS

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegli il nome del cluster di database Aurora che desideri visualizzare dall'elenco.

   Ad esempio, di seguito è riportata l'immagine della pagina dei dettagli per il cluster di database denominato `aurora-test`. Il cluster di database ha quattro istanze DB mostrate nell'elenco DB indentifier (Identificatore database). L'istanza DB writer, `dbinstance4`, è l'istanza DB principale per il cluster di database.



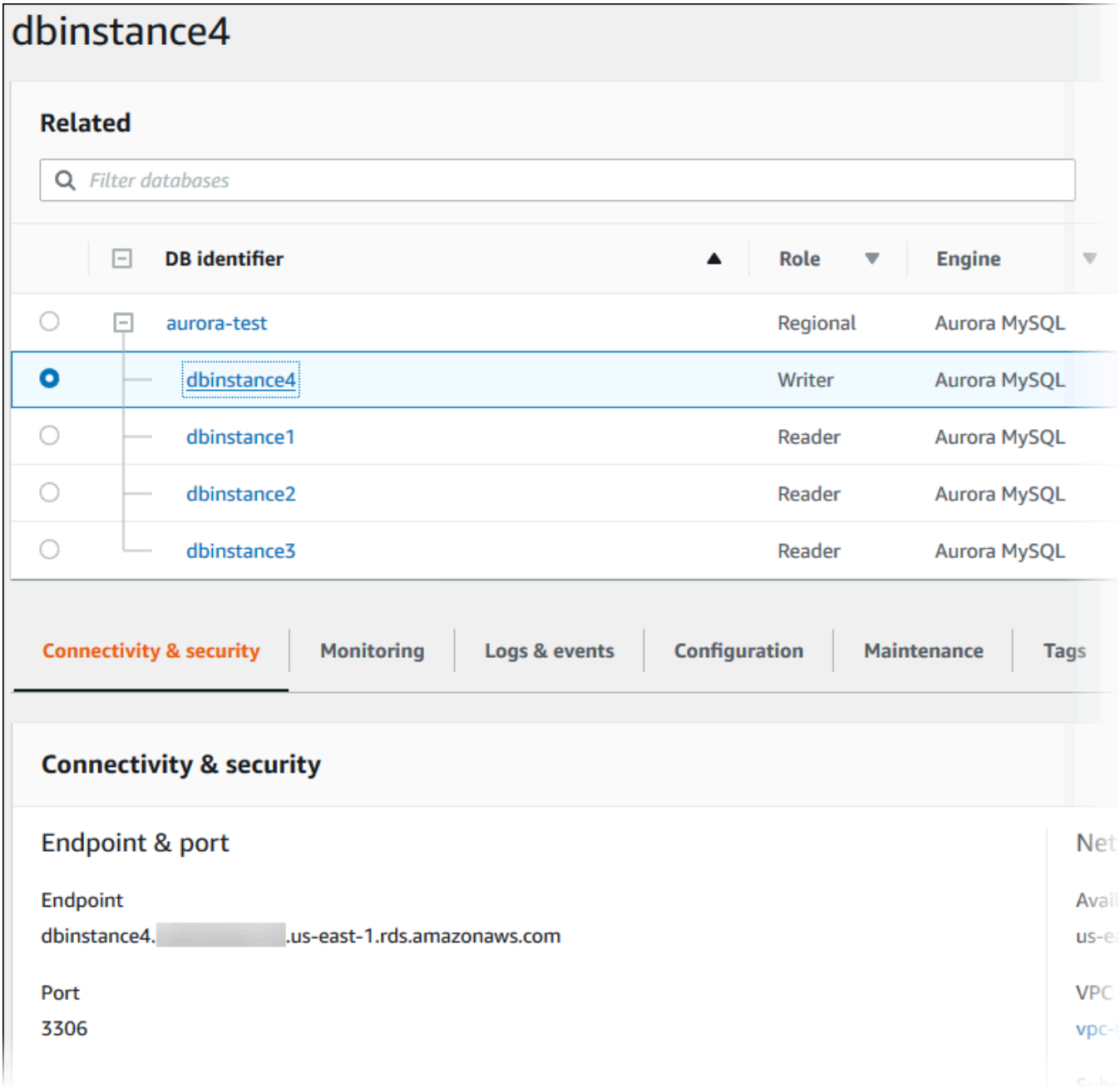
4. Per modificare un cluster di database, selezionalo nell'elenco e scegli Modify (Modifica).

Per visualizzare o modificare istanze database di un cluster di database nella console di Amazon RDS

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Esegui una di queste operazioni:

- Per visualizzare un'istanza database, scegline una dall'elenco che sia un membro del cluster di database Aurora.

  Ad esempio, se scegli l'identificatore istanze database `dbinstance4`, la console mostra la pagina dei dettagli per l'istanza database `dbinstance4`, come mostrato nell'immagine seguente.



- Per modificare un'istanza database, sceglila dall'elenco e seleziona Modify (Modifica). Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

AWS CLI

Per visualizzare le informazioni sul cluster DB utilizzando il AWS CLI, usa il describe-db-clusterscomando. Ad esempio, il AWS CLI comando seguente elenca le informazioni sul cluster DB per tutti i cluster DB nell'`us-east-1`area di modifica per l' AWS account configurato.

```
aws rds describe-db-clusters --region us-east-1
```

Il comando restituisce il seguente output se si AWS CLI è configurati per l'output JSON.

```
{
    "DBClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "sample-cluster1.cluster-123456789012.us-
east-1.rds.amazonaws.com"
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "sample-cluster1",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T03:35:42.563Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-replica"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-primary"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "03:34-04:04",
            "VpcSecurityGroups": [
                {
```

```
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-ddb65fec"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:06:08.903Z",
            "PreferredMaintenanceWindow": "wed:08:15-wed:08:45"
        },
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "aurora-sample.cluster-123456789012.us-
east-1.rds.amazonaws.com",
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "aurora-sample-cluster",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T10:21:34.826Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-replica-sample"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-sample"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "10:20-10:50",
            "VpcSecurityGroups": [
                {
                    "Status": "active",
```

```
                    "VpcSecurityGroupId": "sg-55da224b"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:00:11.491Z",
            "PreferredMaintenanceWindow": "sun:03:53-sun:04:23"
        }
    ]
}
```

API RDS

Per visualizzare le informazioni sui cluster DB utilizzando l'API Amazon RDS, utilizza l'DBClustersoperazione Descrivi.

Visualizzazione dello stato del cluster del DB

Lo stato di un cluster DB indica la sua integrità. Puoi visualizzare lo stato di un cluster DB e delle istanze del cluster utilizzando la console Amazon RDS AWS CLI, o l'API.

> **Note**
>
> Aurora usa anche un altro stato denominato stato di manutenzione, mostrato nella colonna Maintenance (Manutenzione) della console Amazon RDS. Questo valore indica lo stato delle patch di manutenzione da applicare a un cluster DB. Lo stato della manutenzione è indipendente dallo stato del cluster DB. Per ulteriori informazioni sullo stato della manutenzione, consulta Applicazione degli aggiornamenti a un cluster di DB.

Puoi trovare i valori di stato possibili per i cluster DB nella tabella seguente.

Stato cluster DB	Fatturata	Descrizione
Disponibilità	Fatturata	Il cluster DB è integro e disponibile. Quando un cluster Aurora Serverless è disponibile e sospeso, viene addebitato solo lo spazio di archiviazione.
Backup	Fatturata	Il cluster DB è attualmente sottoposto a backup.
Backtracking	Fatturata	Il cluster DB è attualmente sottoposto a backtrack. Questo stato si applica solo ad Aurora MySQL.
Cloning-failed (Clonazione non riuscita)	Non fatturata	Errore nella clonazione di un cluster DB.
Creating (Creazione in corso)	Non fatturata	Il cluster DB è in fase di creazione. Non è possibile accedere al cluster DB mentre è in fase di creazione.

Stato cluster DB	Fatturata	Descrizione
Deleting (Eliminazione in corso)	Non fatturata	Il cluster DB è in fase di eliminazione.
Failing-over (Failover)	Fatturata	È in corso di esecuzione un failover da parte dell'istanza primaria in una replica di Aurora.
Inaccessible-encryption-credentials	Non fatturata	Non è possibile accedere o recuperare le informazioni AWS KMS key utilizzate per crittografare o decrittografare il cluster DB.
Inaccessible-encryption-credentials-recoverable	Fatturato per storage	Non è possibile accedere alla chiave KMS utilizzata per crittografare o decrittografare il cluster di database. Tuttavia, se la chiave KMS è attiva, il riavvio del cluster di database può ripristinarla. Per ulteriori informazioni, consulta Creazione di un cluster di database Amazon Aurora.
Maintenance (Manutenzione)	Fatturata	È in corso l'applicazione da parte di Amazon RDS di un aggiornamento di manutenzione al cluster DB. Questo stato viene utilizzato per la manutenzione a livello di cluster DB pianificata in anticipo da RDS.
Migrating (Migrazione in corso)	Fatturata	Uno snapshot cluster DB è stato ripristinato in un cluster DB.
Migration-failed (Migrazione non riuscita)	Non fatturata	Una migrazione non è riuscita.
Modifying (Modifica in corso)	Fatturata	È in corso la modifica del cluster DB in seguito alla richiesta da parte di un cliente.
Promoting (Promozione in corso)	Fatturata	Una replica di lettura è in corso di promozione su un cluster di database standalone.

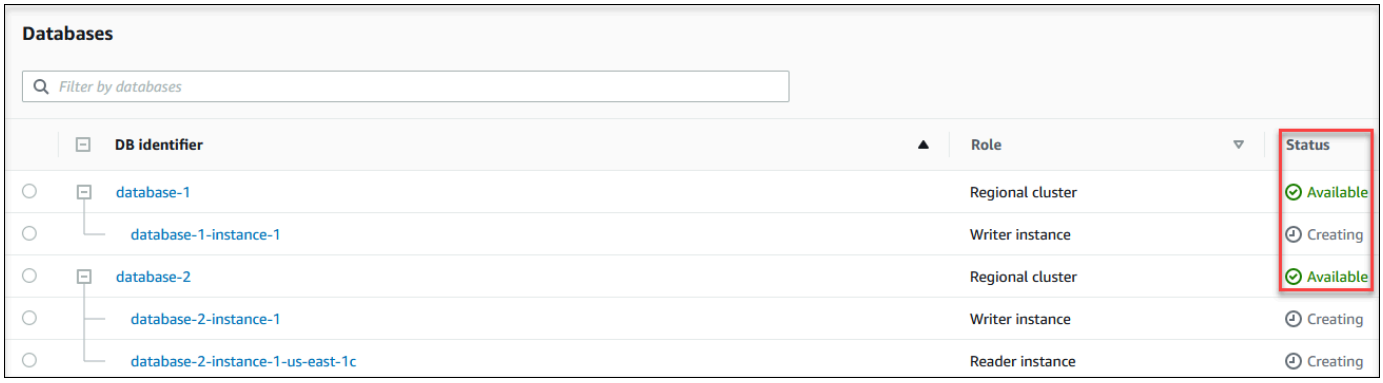
Stato cluster DB	Fatturata	Descrizione
Preparing-data-migration	Fatturato	Amazon RDS si sta preparando a migrare i dati su Aurora.
Ridenominazione	Fatturata	È in corso la ridenominazione del cluster DB in seguito alla richiesta da parte di un cliente.
Resetting-master-credentials	Fatturato	È in corso il ripristino delle credenziali master del cluster DB in seguito alla richiesta da parte di un cliente.
Avvio di	Fatturato per storage	Il cluster di database è in fase di avvio.
Arrestate	Fatturato per storage	Il cluster di database è stato arrestato.
Stopping (In arresto)	Fatturato per storage	Il cluster di database è in fase di arresto.
Storage-optimization (Ottimizzazione archiviazione)	Fatturata	È in corso la modifica delle dimensioni o del tipo di storage dell'istanza database. L'istanza database è completamente operativa. Tuttavia, quando per l'istanza database è attivo lo stato storage-optimization, non è possibile richiedere modifiche dello storage dell'istanza database. Il processo di ottimizzazione dello storage è solitamente breve, ma a volte può richiedere oltre 24 ore.
Update-iam-db-auth	Fatturato	Autorizzazione IAM per il cluster DB che è attualmente sottoposto ad aggiornamento.
Aggiornamento	Fatturato	Il motore del cluster DB o la versione del sistema operativo sono in fase di aggiornamento.

Console

Per visualizzare o stato di un cluster di database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).

   La pagina Databases (Database) viene visualizzata con l'elenco dei cluster di database. Per ogni cluster di database, viene visualizzato un valore dello stato.



CLI

Per visualizzare solo lo stato dei cluster DB, usa la seguente query in. AWS CLI

```
aws rds describe-db-clusters --query 'DBClusters[*].[DBClusterIdentifier,Status]' --
output table
```

Visualizzazione dello stato dell'istanza database di in un cluster Aurora

Lo stato di un'istanza database in un Aurora cluster indica l'integrità dell'istanza db. Puoi utilizzare le seguenti procedure per visualizzare lo stato dell'istanza DB di un cluster nella console Amazon RDS, nel AWS CLI comando o nell'operazione API.

> **Note**
>
> Amazon RDS usa anche un altro stato denominato stato di manutenzione, mostrato nella colonna Maintenance (Manutenzione) della console Amazon RDS. Questo valore indica lo stato delle patch di manutenzione da applicare a un'istanza database. Lo stato della manutenzione è indipendente dallo stato dell'istanza database. Per ulteriori informazioni sullo stato della manutenzione, consulta Applicazione degli aggiornamenti a un cluster di DB.

I valori di stato possibili per le Istanze DB nella tabella seguente. Nella tabella viene inoltre indicato se è prevista la fatturazione per l'istanza db e lo storage, o solo per lo storage oppure se la fatturazione non è prevista. Per tutti gli stati delle istanze database, l'utilizzo del backup viene inserito in fattura.

Stato istanza database	Fatturata	Descrizione
available (disponibile)	Fatturata	L'istanza database è integra e disponibile.
backing-up (backup in corso)	Fatturata	L'istanza database è attualmente sottoposta a backup.
backtracking (backtrack in corso)	Fatturata	L'istanza database è attualmente sottoposta a backtrack. Questo stato si applica solo ad Aurora MySQL.
configuring-enhanced-monitoring	Fatturato	Il monitoraggio avanzato per questa istanza database è in fase di abilitazione/disabilitazione.
configuring-iam-database-auth	Fatturato	AWS Identity and Access Management L'autenticazione del database (IAM) è abilitata o disabilitata per questa istanza DB.

Stato istanza database	Fatturata	Descrizione
configuring-log-exports	Fatturato	La pubblicazione dei file di log su Amazon CloudWatch Logs è abilitata o disabilitata per questa istanza DB.
converting-to-vpc	Fatturato	È in corso la conversione dell'istanza database da un'istanza database che non si trova in un Amazon Virtual Private Cloud (Amazon VPC) a un'istanza database che si trova in un Amazon VPC.
creating (creazione in corso)	Non fatturato (non PITR) Fatturato (solo PITR)	È in corso la creazione dell'istanza database. Non è possibile accedere all'istanza database mentre è in fase di creazione. Se ripristini un database durante il point-in-time ripristino (PITR), ti verrà addebitato il costo mentre il database è in fase di creazione. Questo è l'unico scenario in cui lo Stato di creazione incorre in addebiti.
delete-precheck	Non fatturata	Amazon RDS sta verificando se le repliche di lettura sono integre e sicure per l'eliminazione.
deleting (eliminazione in corso)	Non fatturata	È in corso l'eliminazione dell'istanza database.
Non riuscito	Non fatturata	L'istanza database ha restituito un errore e Amazon RDS non è stato in grado di recuperarla. Esegui un point-in-time ripristino all'ora di ripristino più recente dell'istanza DB per ripristinare i dati.
inaccessible-encryption-credentials	Non fatturata	Non è possibile accedere o recuperare l'istanza DB AWS KMS key utilizzata per crittografare o decrittografare l'istanza DB.

Stato istanza database	Fatturata	Descrizione
inaccessible-encryption-credentials-recoverable	Fatturato per storage	Non è possibile accedere alla chiave KMS utilizzata per crittografare o decrittografare l'istanza database. Tuttavia, se la chiave KMS è attiva, il riavvio dell'istanza database può ripristinarla. Per ulteriori informazioni, consulta Creazione di un cluster di database Amazon Aurora.
creazione incompatibile	Non fatturata	Amazon RDS sta tentando di creare un'istanza DB ma non può farlo perché le risorse sono incompatibili con l'istanza DB. Questo stato può verificarsi se, ad esempio, il profilo dell'istanza per l'istanza DB non dispone delle autorizzazioni corrette.
incompatible-network	Non fatturata	Il tentativo di Amazon RDS di eseguire un'operazione di ripristino su un'istanza database ha esito negativo perché il VPC si trova in uno stato che impedisce il completamento dell'operazione. Questo stato si verifica ad esempio se tutti gli indirizzi IP di una sottorete sono in uso e Amazon RDS non è in grado di ottenere un indirizzo IP per l'istanza database.
incompatible-option-group	Fatturato	Amazon RDS ha tentato di applicare una modifica a un gruppo di opzioni, ma non può farlo e non può eseguire il rollback allo stato del gruppo di opzioni precedente. Per ulteriori informazioni, consulta l'elenco Recent Events (Eventi recenti) per l'istanza database. Questo stato si verifica ad esempio se il gruppo di opzioni contiene un'opzione come TDE e l'istanza database non contiene informazioni crittografate.

Stato istanza database	Fatturata	Descrizione
parametri incompatibili	Fatturata	Amazon RDS non è in grado di avviare l'istanza database perché i parametri specificati nel gruppo di parametri database non sono compatibili con l'istanza database. È necessario annullare le modifiche apportate ai parametri o rendere tali parametri compatibili con l'istanza database per ottenere di nuovo l'accesso all'istanza database. Per ulteriori informazioni sui parametri incompatibili, consulta l'elenco Recent Events (Eventi recenti) per l'istanza database.
ripristino incompatibile	Non fatturata	Amazon RDS non può eseguire un point-in-time ripristino. Tra le cause più comuni di questo stato è incluso l'uso di tabelle temporanee o , i tabelle MyISAM con MySQL .
capacità insufficiente	Non fatturata	Amazon RDS non può creare l'istanza perché al momento non è disponibile una capacità sufficiente. Per creare l'istanza database nella stessa AZ con lo stesso tipo di istanza, elimina l'istanza database, attendi qualche ora e prova a crearla di nuovo. In alternativa, crea una nuova istanza utilizzando una classe di istanza o una AZ diversa.
manutenzione	Fatturata	È in corso l'applicazione da parte di Amazon RDS di un aggiornamento di manutenzione all'istan za database. Questo stato viene utilizzato per la manutenzione a livello di istanza pianificata in anticipo da RDS.
modifying (modifica in corso)	Fatturata	È in corso la modifica dell'istanza database in seguito alla richiesta da parte di un cliente.
moving-to-vpc	Fatturato	È in corso lo spostamento dell'istanza database in un nuovo Amazon Virtual Private Cloud (Amazon VPC).

Stato istanza database	Fatturata	Descrizione
riavvio in corso	Fatturata	È in corso il riavvio dell'istanza database a causa della richiesta di un cliente o perché è necessario per un processo Amazon RDS.
resetting-master-credentials	Fatturato	È in corso il ripristino delle credenziali master dell'istanza database in seguito alla richiesta da parte di un cliente.
renaming (ridenominazione)	Fatturata	È in corso la ridenominazione dell'istanza database in seguito alla richiesta da parte di un cliente.
restore-error (errore-ripristino)	Fatturato	L'istanza DB ha riscontrato un errore nel tentativo di ripristinare uno snapshot point-in-time o da uno snapshot.
avvio in corso	Fatturato per storage	L'istanza database è in fase di avvio.
stopped (arrestato)	Fatturato per storage	L'istanza database è stata arrestata.
stopping (in arresto)	Fatturato per storage	L'istanza database è in fase di arresto.
storage-config-upgrade	Fatturato	La configurazione del file system di archiviazione dell'istanza database è in fase di aggiornamento. Questo stato si applica solo ai database verdi all'interno di un'implementazione blu/verde o alle repliche di lettura delle istanze database.

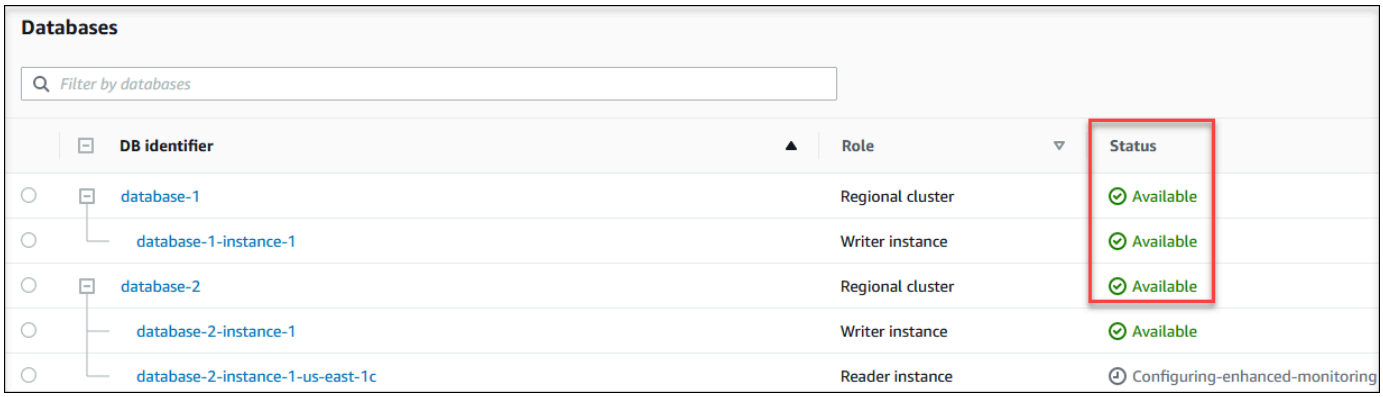
Stato istanza database	Fatturata	Descrizione
storage pieno	Fatturata	L'istanza database ha raggiunto la capacità di storage allocata. Si tratta di uno stato critico che richiede l'immediata risoluzione del problema. A tale scopo, è necessario aumentare la capacità di storage modificando l'istanza database. Per evitare questa situazione, imposta gli CloudWatch allarmi di Amazon per avvisarti quando lo spazio di archiviaz ione si sta esaurendo.
inizializzazione dello storage	Fatturato	L'istanza DB sta caricando blocchi di dati da Amazon S3 per ottimizzare le prestazioni del volume dopo il ripristino da uno snapshot. Rimane disponibile per le operazioni, ma le prestazioni potrebbero non essere ottimali fino al completamento dell'inizializzazione.
storage-optimizati on (ottimizzazione-st orage)	Fatturato	Amazon RDS sta ottimizzando lo storage dell'istanza database. Il processo di ottimizzazione dello storage è solitamente breve, ma a volte può richiedere oltre 24 ore. Durante l'ottimizzazione dello storage, l'istanza DB rimane disponibile. L'ottimizzazione dello storage è un processo in background che non influisce sulla disponibilità dell'istanza.
aggiornamento in corso	Fatturato	È in corso l'aggiornamento del motore di database o della versione del sistema operativo.

Console

Per visualizzare lo stato di un'istanza database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).

La pagina Databases (Database) viene visualizzata con l'elenco delle istanze database. Per ogni istanza database in un cluster, viene visualizzato il valore dello stato.



CLI

Per visualizzare l'istanza DB e le relative informazioni sullo stato utilizzando il AWS CLI, usa il describe-db-instancescomando. Ad esempio, il AWS CLI comando seguente elenca tutte le informazioni sulle istanze DB.

```
aws rds describe-db-instances
```

Per visualizzare un'istanza DB specifica e il relativo stato, chiamate il describe-db-instancescomando con l'opzione seguente:

- `DBInstanceIdentifier` – Il nome dell'istanza database.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

Per visualizzare solo lo stato di tutte le istanze DB, usa la seguente query in AWS CLI.

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

API

Per visualizzare lo stato dell'istanza DB utilizzando l'API Amazon RDS, chiama l'DBInstancesoperazione Descrivi.

Amazon Aurora fornisce consigli automatici per le risorse di database, come istanze DB, cluster DB, e gruppi di parametri DB. Queste raccomandazioni forniscono consigli sulle best practice analizzando la configurazione dei cluster di database, la configurazione delle istanze database, l'utilizzo e i dati sulle prestazioni.

Amazon RDS Performance Insights monitora parametri specifici e crea automaticamente soglie analizzando quali livelli sono considerati potenzialmente problematici per una risorsa specifica. Quando i nuovi valori delle metriche superano una soglia predefinita in un determinato periodo di tempo, Performance Insights genera una raccomandazione proattiva. Questa raccomandazione aiuta a prevenire futuri impatti sulle prestazioni del database. Ad esempio, la raccomandazione «Idle In Transaction» viene generata Aurora SQL Postgre quando le sessioni connesse al database non svolgono un lavoro attivo, ma possono mantenere bloccate le risorse del database. Per ricevere consigli proattivi, devi attivare Performance Insights con un periodo di conservazione a pagamento. Per informazioni sull'attivazione di Performance Insights, consultaAttivazione e disattivazione di Performance Insights per Aurora. Per informazioni sui prezzi e sulla conservazione dei dati per Performance Insights, vederePrezzi e conservazione dei dati per Performance Insights.

DevOpsGuru for RDS monitora determinate metriche per rilevare quando il comportamento della metrica diventa molto insolito o anomalo. Queste anomalie vengono segnalate come approfondimenti reattivi con raccomandazioni. Ad esempio, DevOps Guru for RDS potrebbe consigliarvi di prendere in considerazione l'aumento CPU della capacità o di analizzare gli eventi di attesa che contribuiscono al carico del DB. DevOpsGuru for fornisce RDS anche consigli proattivi basati su soglie. Per questi consigli, devi attivare DevOps Guru per. RDS Per informazioni su come attivare DevOps Guru perRDS, vedi. Attivare DevOps Guru e specificare la copertura delle risorse

I consigli avranno uno dei seguenti stati: attivi, ignorati, in sospeso o risolti. I consigli risolti sono disponibili per 365 giorni.

È possibile visualizzare o ignorare i consigli. È possibile applicare immediatamente un consiglio attivo basato sulla configurazione, programmarlo nella finestra di manutenzione successiva o ignorarlo. Per i consigli proattivi basati sulla soglia e quelli reattivi basati sull'apprendimento automatico, è necessario esaminare la causa suggerita del problema e quindi eseguire le azioni consigliate per risolverlo.

I consigli sono supportati nei seguenti casi: Regioni AWS

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)

- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Asia Pacifico (Mumbai)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Europa (Francoforte)
- Europa (Irlanda)
- Europe (London)
- Europe (Paris)
- Europa (Stoccolma)
- Sud America (San Paolo)

Scopri come visualizzare, applicare, ignorare e modificare i consigli di Aurora nelle seguenti sezioni.

Argomenti

- Visualizzazione dei di RDS Amazon Aurora
- Applicazione dei consigli di Aurora
- Ignorare i di RDS Amazon Aurora
- Modificare i consigli ignorati di Aurora in consigli attivi
- Raccomandazioni tratte dal RDS riferimento Aurora

Visualizzazione dei di RDS Amazon Aurora

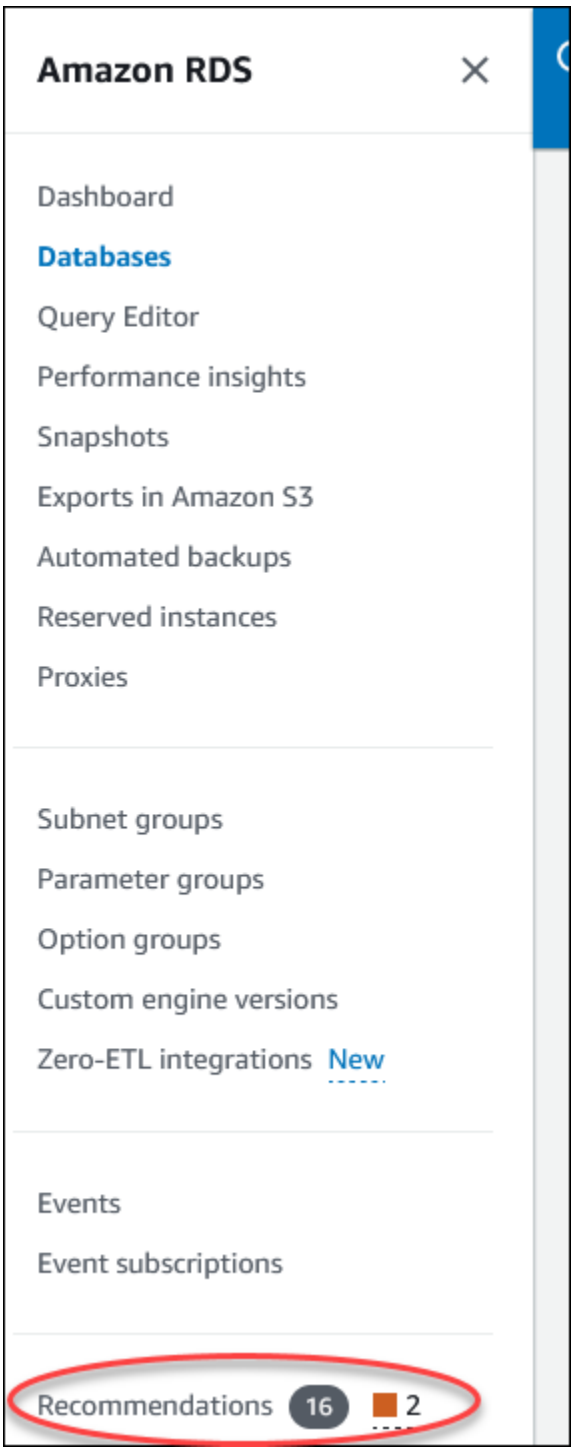
Utilizzando la RDS console Amazon, puoi visualizzare i consigli di Amazon Aurora per le tue risorse di database. Per un cluster DB, vengono visualizzati i consigli per il cluster DB e le relative istanze.

Console

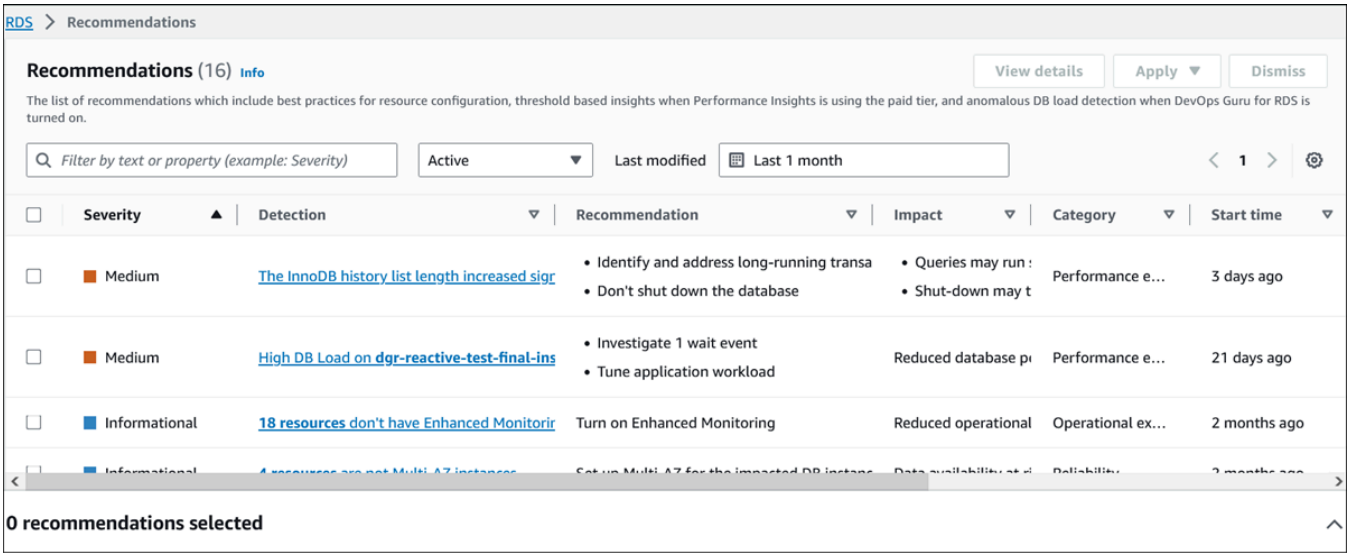
Per visualizzare i consigli di Amazon Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, esegui una delle seguenti operazioni:

   - Scegliete Consigli. Il numero di consigli attivi per le tue risorse e il numero di consigli con la massima severità generati nell'ultimo mese sono disponibili accanto a Consigli. Per trovare il numero di consigli attivi per ogni gravità, scegli il numero che mostra la gravità più alta.

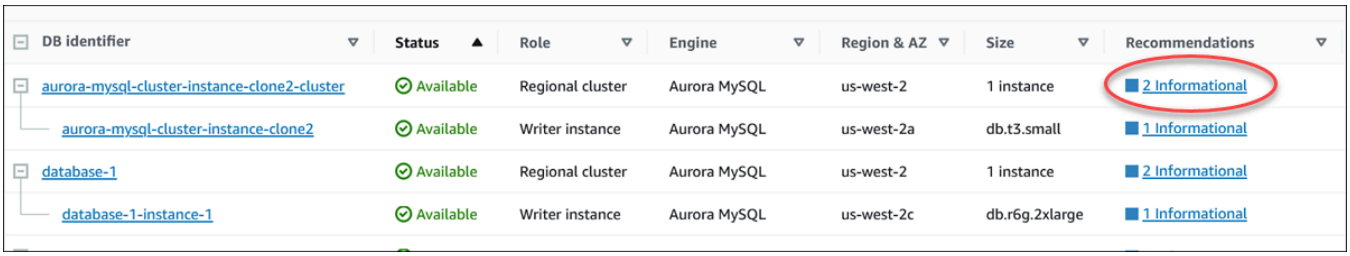


Per impostazione predefinita, la pagina Consigli mostra un elenco di nuovi consigli nell'ultimo mese. Amazon Aurora fornisce consigli per tutte le risorse del tuo account e li ordina in base alla loro gravità.

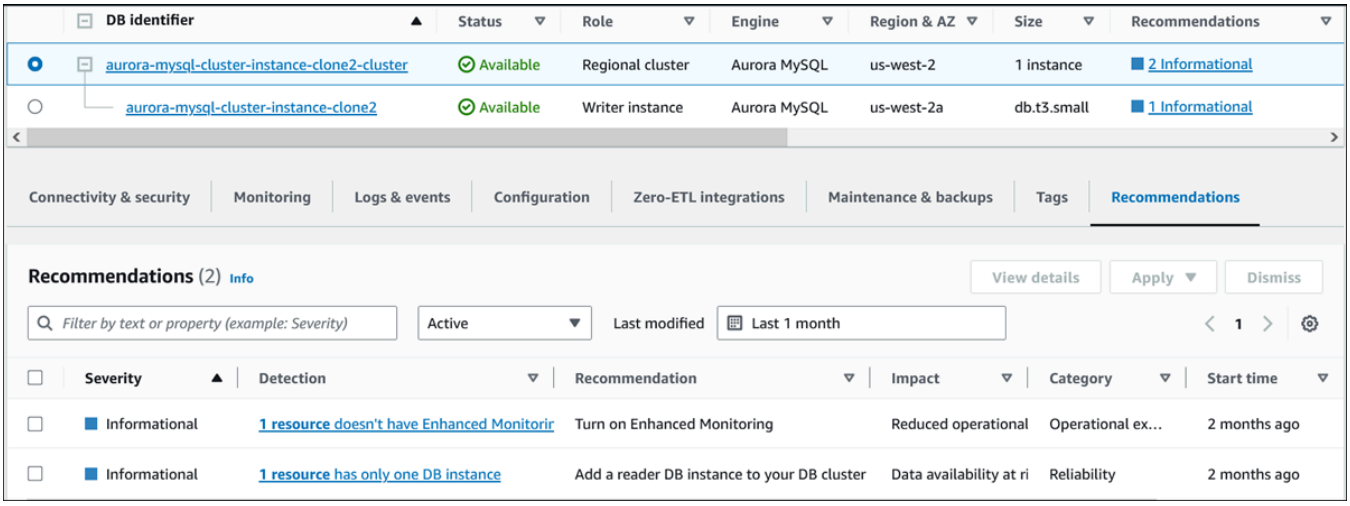


Puoi scegliere un consiglio per visualizzare una sezione nella parte inferiore della pagina che contiene le risorse interessate e i dettagli su come verrà applicata la raccomandazione.

- Nella pagina Database, scegli Consigli per una risorsa.



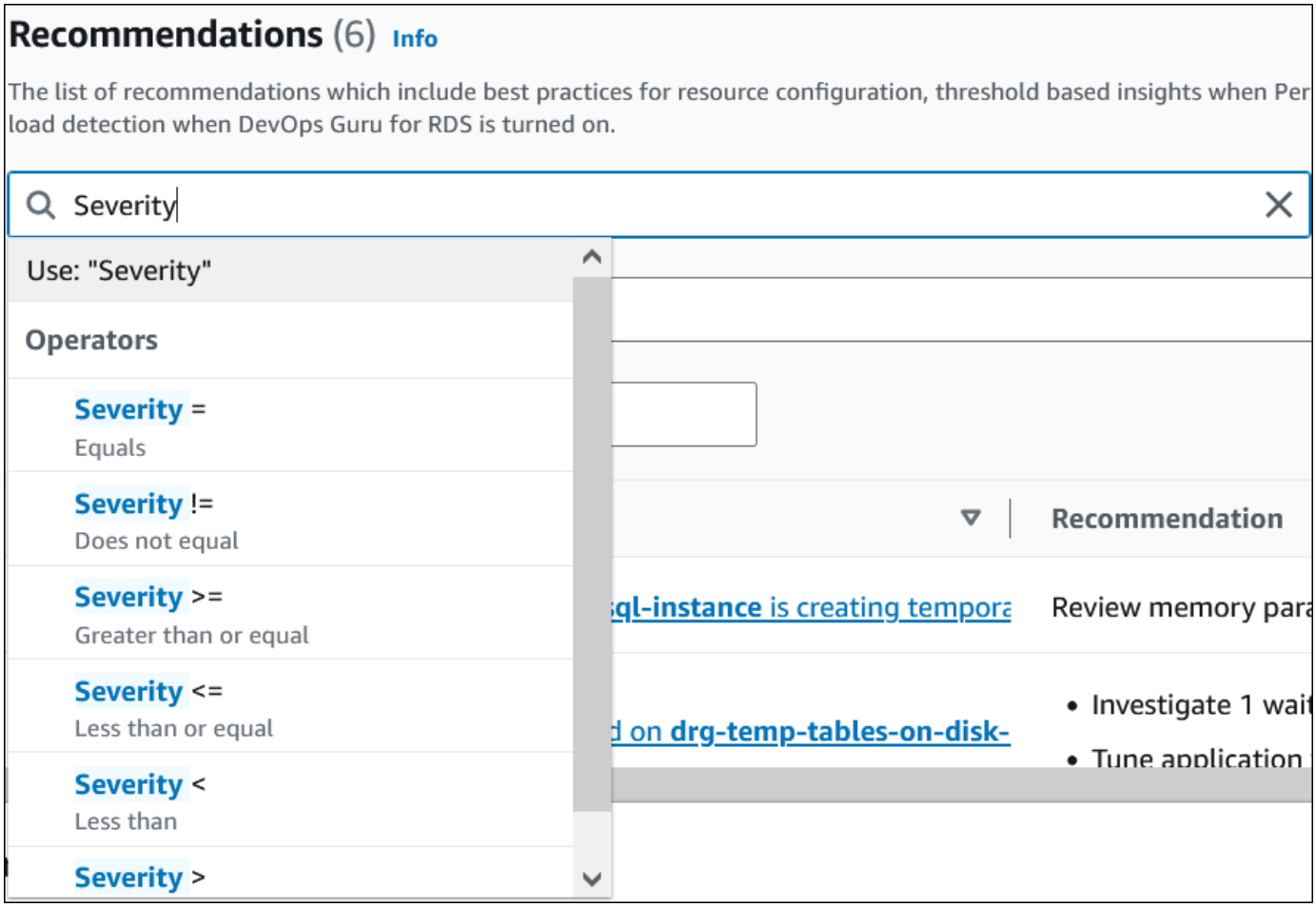
La scheda Consigli mostra i consigli e i relativi dettagli per la risorsa selezionata.



I seguenti dettagli sono disponibili per i consigli:

- Gravità: il livello di implicazione del problema. I livelli di gravità sono Alto, Medio, Basso e Informativo.
- Rilevamento: il numero di risorse interessate e una breve descrizione del problema. Scegli questo link per visualizzare la raccomandazione e i dettagli dell'analisi.
- Raccomandazione: una breve descrizione dell'azione consigliata da applicare.
- Impatto: una breve descrizione del possibile impatto quando la raccomandazione non viene applicata.
- Categoria: il tipo di raccomandazione. Le categorie sono Efficienza delle prestazioni, Sicurezza, Affidabilità, Ottimizzazione dei costi, Eccellenza operativa e Sostenibilità.
- Stato: lo stato attuale della raccomandazione. Gli stati possibili sono Tutti, Attivo, Ignorato, Risolto e In sospeso.
- Ora di inizio: l'ora in cui è iniziato il problema. Ad esempio, 18 ore fa.
- Ultima modifica: l'ora in cui il consiglio è stato aggiornato l'ultima volta dal sistema a causa di una modifica della severità o l'ora in cui hai risposto al consiglio. Ad esempio, 10 ore fa.
- Ora di fine: l'ora in cui il problema è terminato. L'ora non verrà visualizzata per eventuali problemi persistenti.
- Identificatore di risorsa: il nome di una o più risorse.

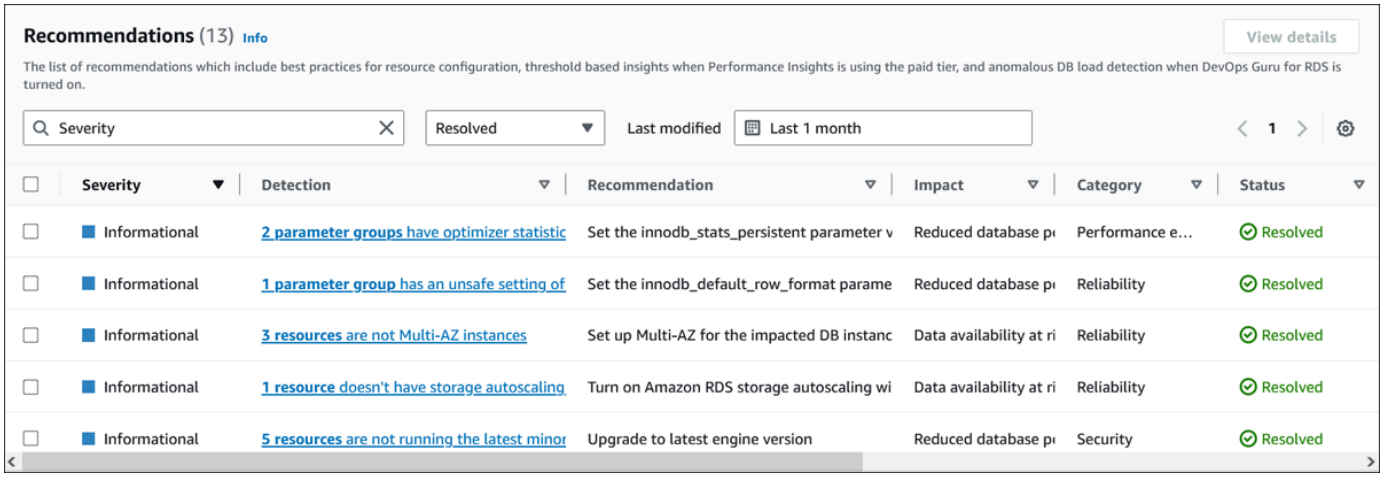
3. (Facoltativo) Scegliete gli operatori di severità o categoria nel campo per filtrare l'elenco dei consigli.



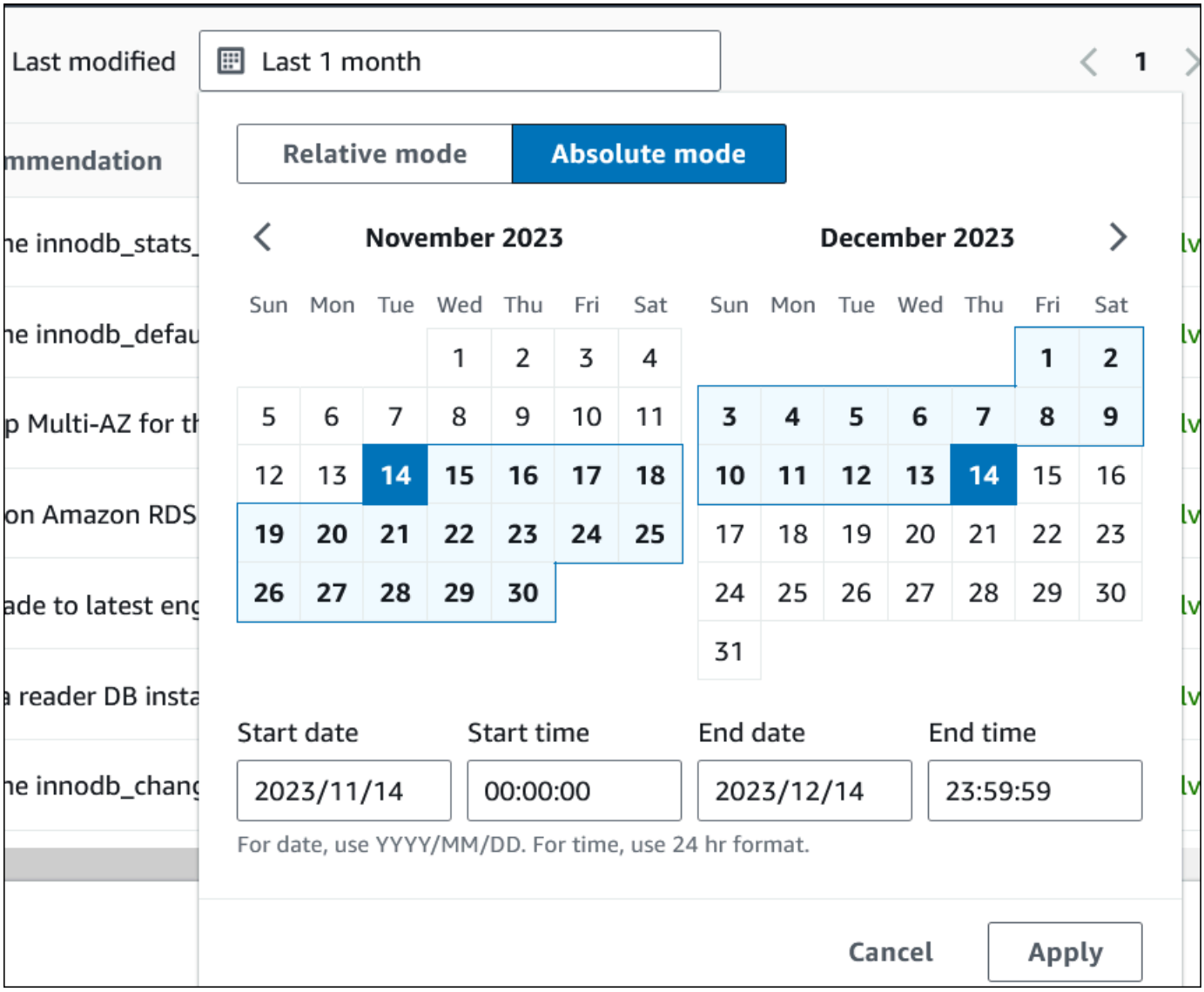
Vengono visualizzati i consigli per l'operazione selezionata.

4. (Facoltativo) Scegliete uno dei seguenti stati di raccomandazione:

   - Attivo (impostazione predefinita): mostra i consigli correnti che è possibile applicare, programmarli per la finestra di manutenzione successiva o ignorarli.
   - Tutti: mostra tutti i consigli con lo stato corrente.
   - Ignorato: mostra i consigli ignorati.
   - Risolto: mostra i consigli che sono stati risolti.
   - In sospeso: mostra i consigli le cui azioni consigliate sono in corso o pianificate per la finestra di manutenzione successiva.



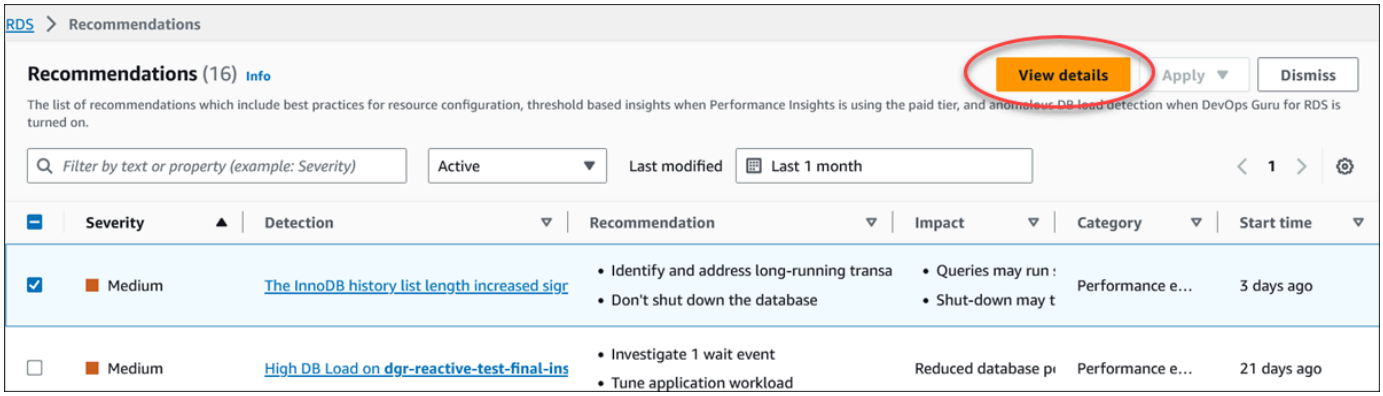
5. (Facoltativo) Scegliete la modalità relativa o la modalità Assoluta in Ultima modifica per modificare il periodo di tempo. La pagina Consigli mostra i consigli generati nel periodo di tempo. Il periodo di tempo predefinito è l'ultimo mese. In modalità Assoluta, puoi scegliere il periodo di tempo o inserire l'ora nei campi Data di inizio e Data di fine.



Vengono visualizzati i consigli per il periodo di tempo impostato.

Tieni presente che puoi visualizzare tutti i consigli relativi alle risorse del tuo account impostando l'intervallo su Tutti.

6. (Facoltativo) Scegli Preferenze a destra per personalizzare i dettagli da visualizzare. Puoi scegliere una dimensione di pagina, disporre le righe del testo e consentire o nascondere le colonne.
7. (Facoltativo) Scegli un consiglio, quindi scegli Visualizza dettagli.



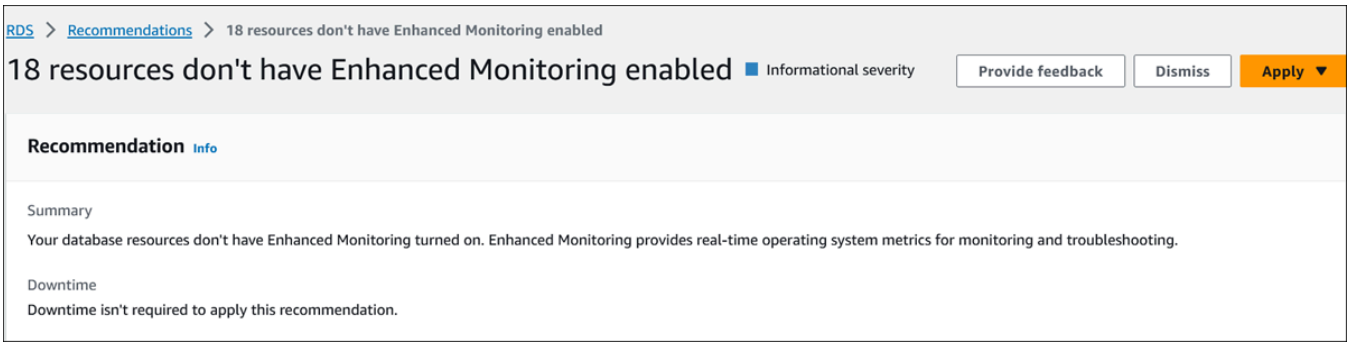
Viene visualizzata la pagina dei dettagli dei consigli. Il titolo fornisce il conteggio totale delle risorse con il problema rilevato e la gravità.

Per informazioni sui componenti nella pagina dei dettagli per una raccomandazione reattiva basata sulle anomalie, consulta la sezione Visualizzazione delle anomalie reattive nella Amazon DevOps Guru User Guide.

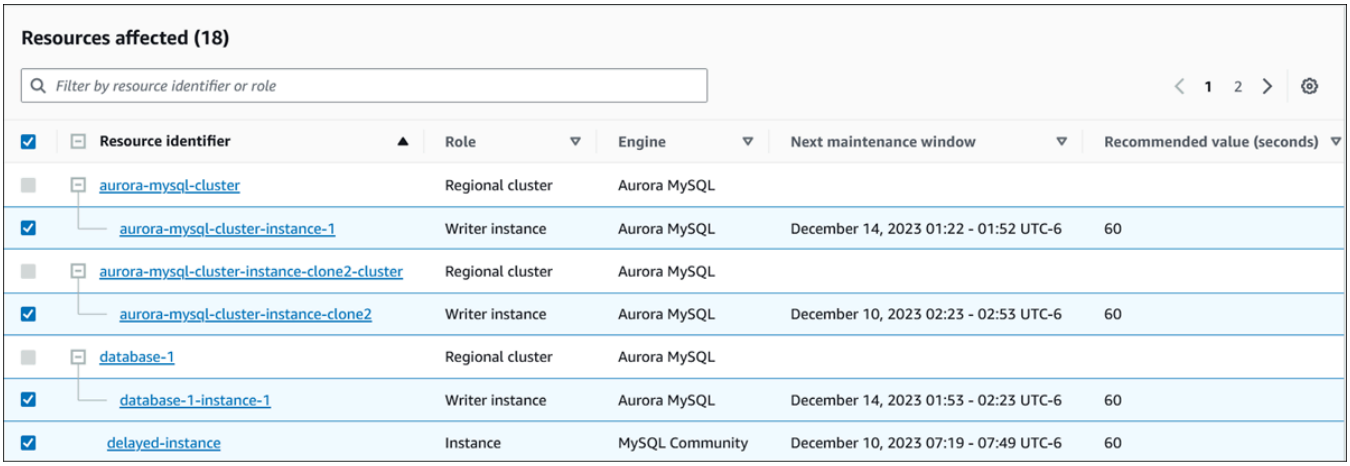
Per informazioni sui componenti nella pagina dei dettagli per una raccomandazione proattiva basata su una soglia, consulta. Visualizzazione dei consigli proattivi di Performance Insights

Gli altri consigli automatici visualizzano i seguenti componenti nella pagina dei dettagli dei consigli:

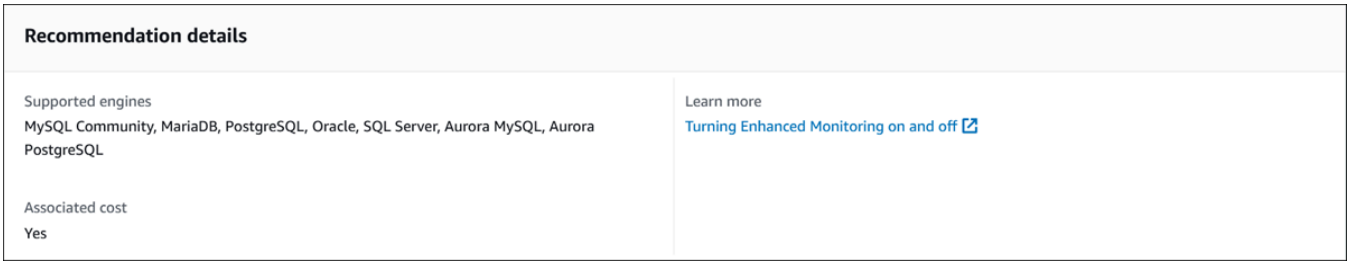
- Raccomandazione: un riepilogo della raccomandazione e indica se sono necessari tempi di inattività per applicarla.



- Risorse interessate: dettagli delle risorse interessate.



- Dettagli sui consigli: informazioni sul motore supportato, eventuali costi associati necessari per applicare il consiglio e link alla documentazione per saperne di più.



CLI

Per visualizzare i RDS consigli di Amazon sulle istanze DB o sui cluster DB, usa il seguente comando in. AWS CLI

```
aws rds describe-db-recommendations
```

RDS API

Per visualizzare i RDS consigli di Amazon sull'utilizzo di Amazon RDSAPI, utilizza l'escribeDBRecommendationsoperazione D.

Applicazione dei consigli di Aurora

Per applicare i consigli di Aurora utilizzando la RDS console Amazon, seleziona un consiglio basato sulla configurazione o una risorsa interessata nella pagina dei dettagli. Quindi, scegli di applicare immediatamente il consiglio o programmalo per la finestra di manutenzione successiva. Potrebbe essere necessario riavviare la risorsa per rendere effettiva la modifica. Per alcuni consigli sui gruppi di parametri DB, potrebbe essere necessario riavviare le risorse.

I consigli proattivi basati sulla soglia o reattivi basati sulle anomalie non avranno l'opzione Applica e potrebbero richiedere un'ulteriore revisione.

Console

Per applicare un consiglio basato sulla configurazione

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, esegui una delle seguenti operazioni:

   - Scegliete Consigli.

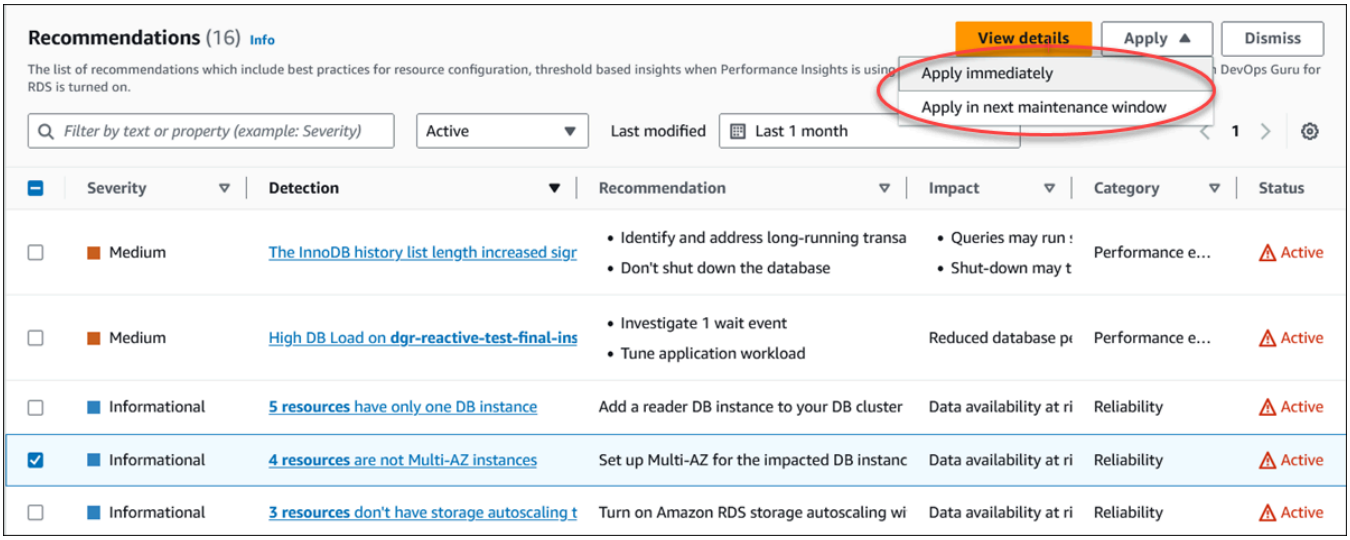
     Viene visualizzata la pagina Consigli con l'elenco di tutti i consigli.
   - Scegli Database, quindi scegli Consigli per una risorsa nella pagina dei database.

     I dettagli vengono visualizzati nella scheda Consigli per il consiglio selezionato.
   - Scegli Rilevamento per un consiglio attivo nella pagina Consigli o nella scheda Consigli nella pagina Database.

     Viene visualizzata la pagina dei dettagli dei consigli.
3. Scegli un consiglio o una o più risorse interessate nella pagina dei dettagli del consiglio ed esegui una delle seguenti operazioni:

   - Scegli Applica, quindi scegli Applica immediatamente per applicare immediatamente il consiglio.
   - Scegli Applica, quindi scegli Applica nella finestra di manutenzione successiva per programmarla nella finestra di manutenzione successiva.

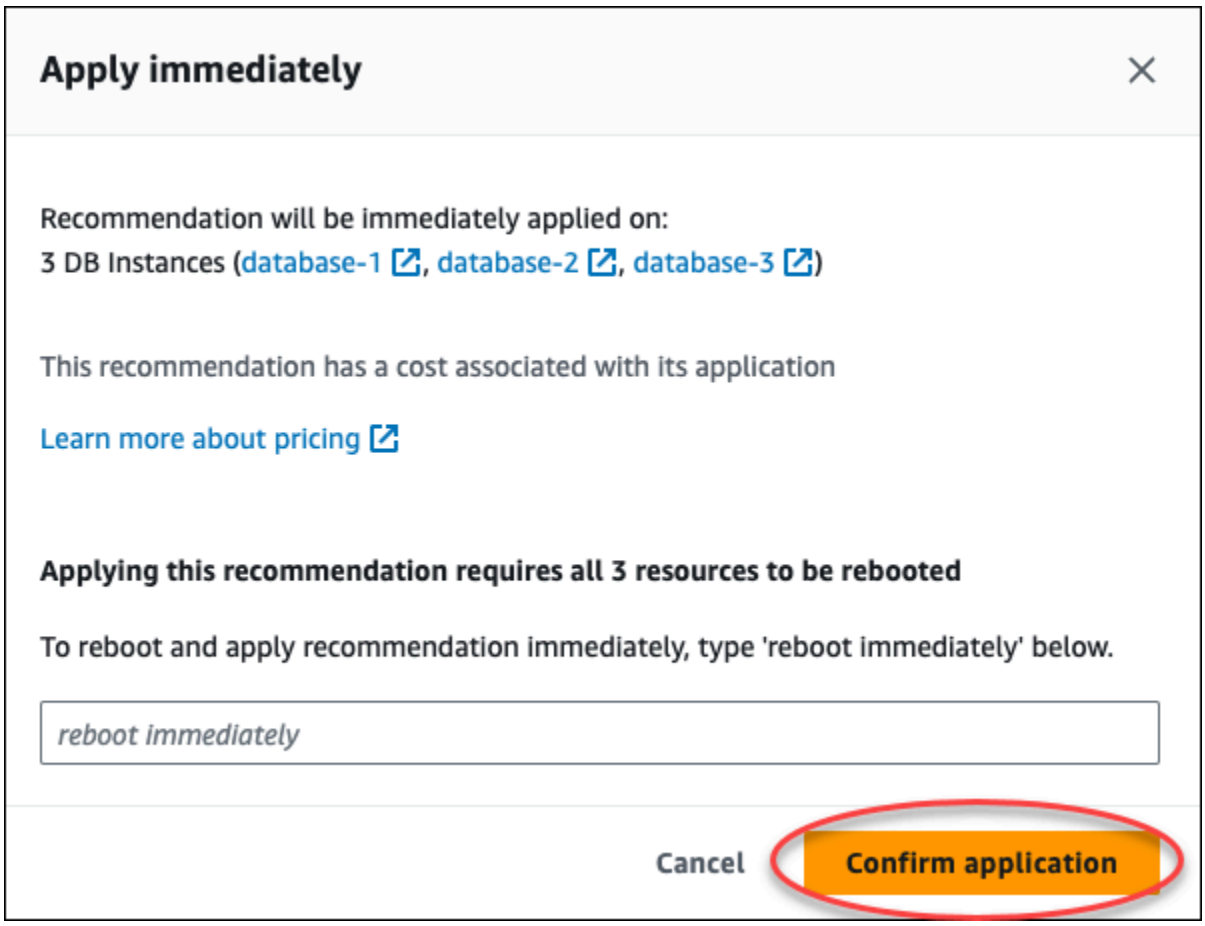
     Lo stato del consiglio selezionato viene aggiornato in sospeso fino alla finestra di manutenzione successiva.



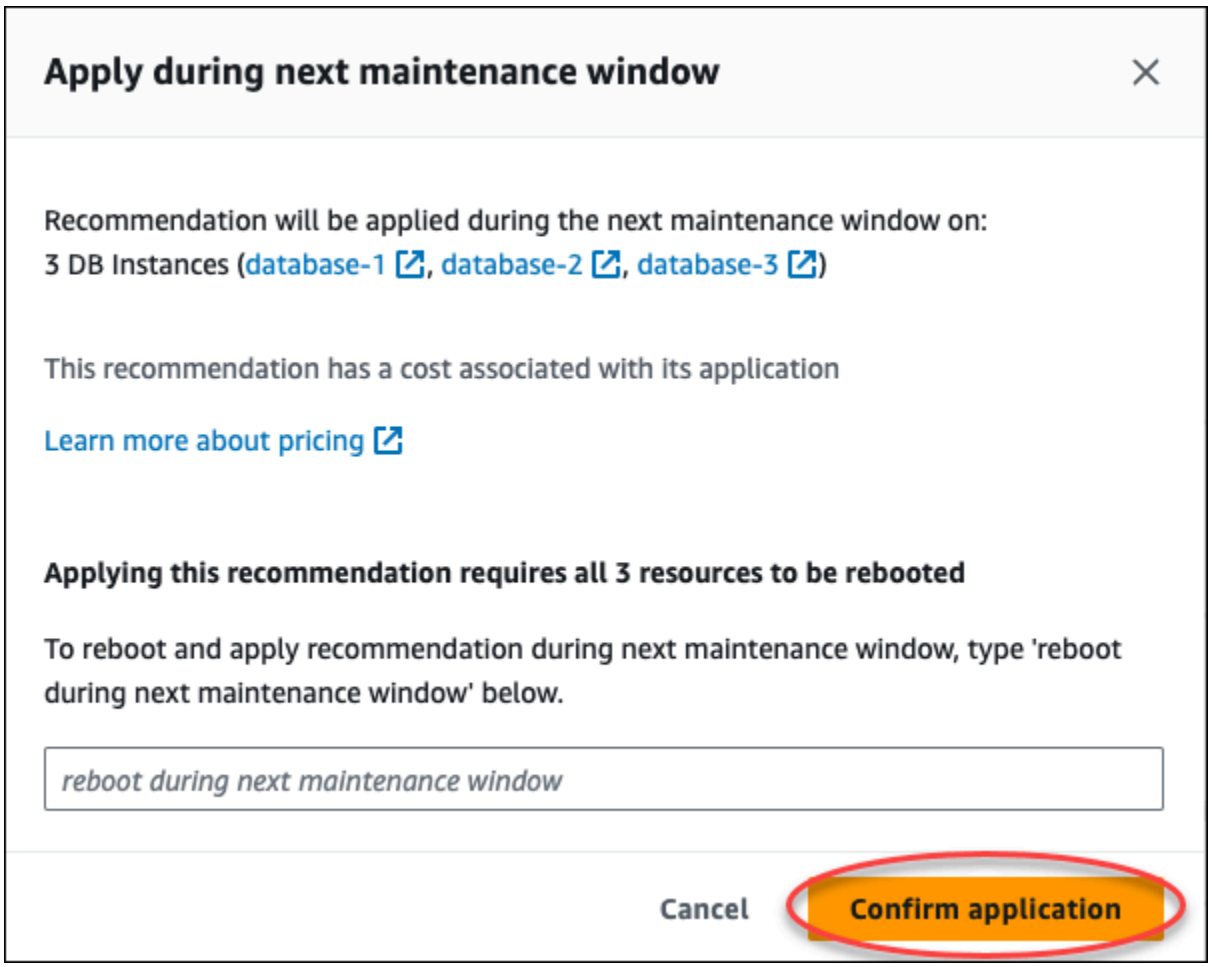
Viene visualizzata una finestra di conferma.

4. Scegli Conferma applicazione per applicare il consiglio. Questa finestra conferma se le risorse necessitano di un riavvio automatico o manuale per rendere effettive le modifiche.

   L'esempio seguente mostra la finestra di conferma per applicare immediatamente il consiglio.

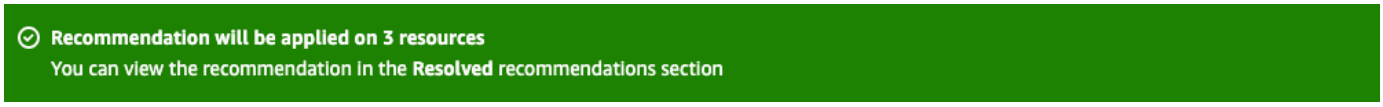


L'esempio seguente mostra la finestra di conferma per pianificare l'applicazione del consiglio nella finestra di manutenzione successiva.



Un banner mostra un messaggio quando il consiglio applicato ha esito positivo o negativo.

L'esempio seguente mostra il banner con il messaggio di successo.



L'esempio seguente mostra il banner con il messaggio di errore.

RDS API

Per applicare una configurazione basata su Aurora, consigliamo di utilizzare Amazon RDS API

1. Usa l'escribeDBRecommendationsoperazione D. `RecommendedActions`Nell'output possono essere presenti una o più azioni consigliate.
2. Utilizzate l'RecommendedActionoggetto per ogni azione consigliata dal passaggio 1. L'output contiene `Operation` e`Parameters`.

   L'esempio seguente mostra l'output con un'azione consigliata.

```
   "RecommendedActions": [
       {
           "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
           "Title": "Turn on auto backup", // localized
           "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
           "Operation": "ModifyDbInstance",
           "Parameters": [
               {
                   "Key": "DbInstanceIdentifier",
                   "Value": "my-mysql-instance-1"
               },
               {
                   "Key": "BackupRetentionPeriod",
                   "Value": "7"
               }
           ],
           "ApplyModes": ["immediately", "next-maintenance-window"],
           "Status": "applied"
       },
       ... // several others
   ],
```

3. Usa il `operation` per ogni azione consigliata dall'output nel passaggio 2 e inserisci i `Parameters` valori.
4. Una volta completata l'operazione nel passaggio 2, utilizzate l'odifyDBRecommendationoperazione M per modificare lo stato della raccomandazione.

Ignorare i di RDS Amazon Aurora

Puoi ignorare uno o più consigli di Aurora utilizzando la console RDS AWS CLI Amazon o Amazon. RDS API

Console

Per ignorare uno o più consigli

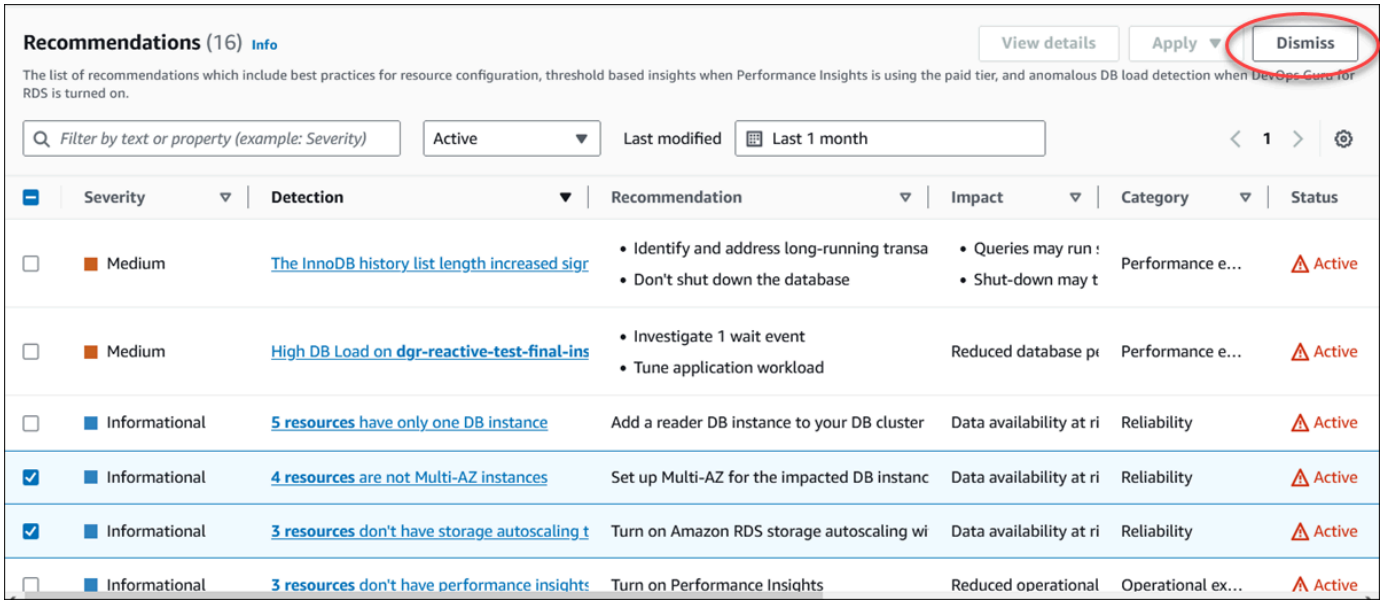
1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, esegui una delle seguenti operazioni:
   - Scegliete Consigli.

     Viene visualizzata la pagina Consigli con l'elenco di tutti i consigli.
   - Scegli Database, quindi scegli Consigli per una risorsa nella pagina dei database.

     I dettagli vengono visualizzati nella scheda Consigli per il consiglio selezionato.
   - Scegli Rilevamento per un consiglio attivo nella pagina Consigli o nella scheda Consigli nella pagina Database.

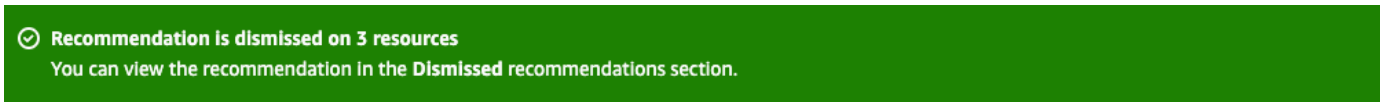
     La pagina dei dettagli dei consigli mostra l'elenco delle risorse interessate.
3. Scegli uno o più consigli o una o più risorse interessate nella pagina dei dettagli del consiglio, quindi scegli Ignora.

   L'esempio seguente mostra la pagina Consigli con più consigli attivi selezionati per essere ignorati.



Un banner mostra un messaggio quando uno o più consigli selezionati vengono ignorati.

L'esempio seguente mostra il banner con il messaggio di successo.



L'esempio seguente mostra il banner con il messaggio di errore.

CLI

Per ignorare Aurora utilizzando il AWS CLI

1. Esegui il comando `aws rds describe-db-recommendations --filters "Name=status,Values=active"`.

   L'output fornisce un elenco di raccomandazioni in `active` corso.

2. `recommendationId`Individua il consiglio che desideri eliminare dal passaggio 1.
3. Esegui il comando `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` `recommendationId` dal passaggio 2 per ignorare il consiglio.

RDS API

Per ignorare raccomandazione Aurora utilizzando RDS API Amazon, usa l'operazione M. odifyDBRecommendation

Modificare i consigli ignorati di Aurora in consigli attivi

Puoi spostare uno o più consigli Aurora ignorati in consigli attivi utilizzando la console RDS AWS CLI Amazon o Amazon. RDS API

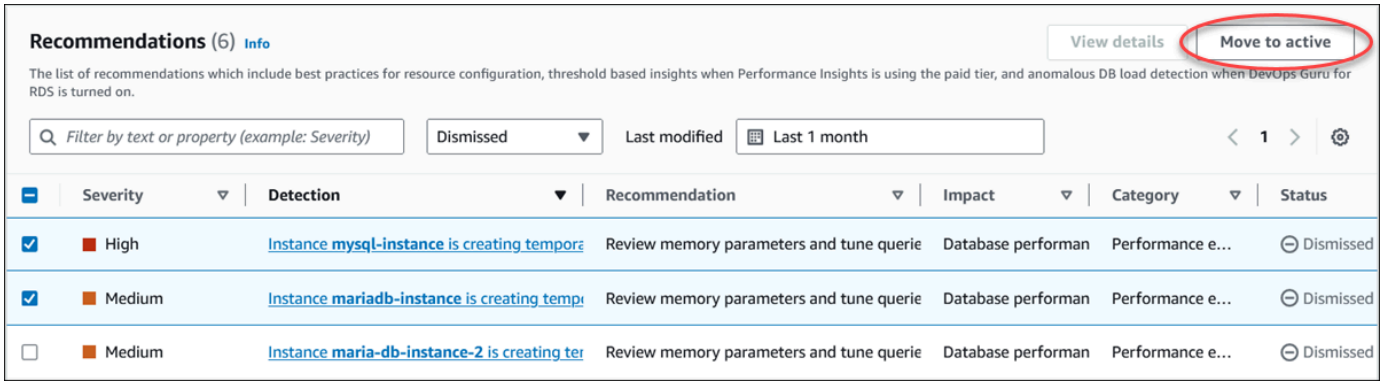
Console

Per spostare uno o più consigli ignorati in consigli attivi

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, esegui una delle seguenti operazioni:
   - Scegliete Consigli.

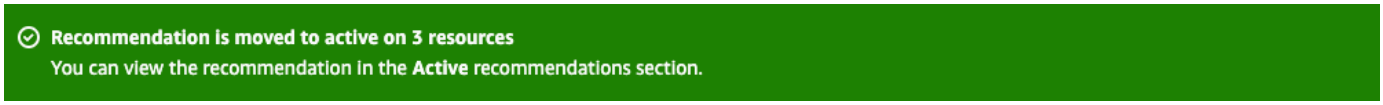
     La pagina Consigli mostra un elenco di consigli ordinati in base alla gravità per tutte le risorse del tuo account.
   - Scegli Database, quindi scegli Consigli per una risorsa nella pagina dei database.

     La scheda Consigli mostra i consigli e i relativi dettagli per la risorsa selezionata.
3. Scegli uno o più consigli ignorati dall'elenco, quindi scegli Sposta su attivo.



Un banner mostra un messaggio di successo o di fallimento quando si spostano i consigli selezionati dallo stato ignorato a quello attivo.

L'esempio seguente mostra il banner con il messaggio di successo.



L'esempio seguente mostra il banner con il messaggio di errore.

CLI

Per modificare una raccomandazione Aurora ignorata in una raccomandazione attiva utilizzando il AWS CLI

1. Esegui il comando `aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"`.

   L'output fornisce un elenco di raccomandazioni in `dismissed` corso.
2. Trova `recommendationId` il suggerimento di cui desideri modificare lo stato dal passaggio 1.
3. Esegui `>aws rds modify-db-recommendation --status active --recommendationId <ID>` il comando `recommendationId` dal passaggio 2 per passare alla raccomandazione attiva.

RDS API

Per modificare una raccomandazione Aurora ignorata in una raccomandazione attiva utilizzando RDS API Amazon, utilizza l'operazione M. odifyDBRecommendation

Raccomandazioni tratte dal RDS riferimento Aurora

Amazon Aurora genera consigli per una risorsa quando la risorsa viene creata o modificata. Puoi trovare esempi di consigli di Amazon Aurora nella tabella seguente.

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I backup di Resource Automated sono disattivati	I backup automatici non sono attivati per le istanze DB. I backup automatici sono consigliati perché consentono il point-in-time ripristino delle istanze DB.	Attiva i backup automatici con un periodo di conservazione fino a 14 giorni.	Sì	Panoramica di backup e ripristino di un cluster di database Aurora Demistificazione dei costi dello storage di backup di RDS Amazon sul AWS Database Blog
È richiesto l'aggiornamento della versione secondaria del motore	Le risorse del database non eseguono l'ultima versione secondaria del motore DB. L'ultima versione secondaria contiene le ultime correzioni di sicurezza e altri miglioramenti.	Esegui l'aggiornamento alla versione più recente del motore.	Sì	Manutenzione di un cluster database Amazon Aurora
Il monitoraggio avanzato è disattivato	Il monitoraggio avanzato non è attivato per le risorse del database. Il monitoraggio avanzato offre i parametri del sistema operativo in tempo reale per il monitorag	Attiva il monitoraggio avanzato.	No	Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
	gio e la risoluzione dei problemi.			
La crittografia dello storage è disattivata	Amazon RDS supporta la crittografia a riposo per tutti i motori di database utilizzando le chiavi gestite in AWS Key Management Service (AWS KMS). Su un'istanza DB attiva con RDS crittografia Amazon, i dati archiviati a riposo nello storage sono crittografati, in modo simile ai backup automatici, alle repliche di lettura e alle istantanee. Se la crittografia non è attivata durante la creazione di un cluster Aurora DB, è necessario ripristinare un'istantanea decrittografata in un cluster DB crittografato.	Attiva la crittografia dei dati inattivi per il tuo cluster DB.	Sì	Sicurezza in Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Cluster DB con tutte le istanze nella stessa zona di disponibilità	I cluster DB si trovano attualmente in un'unica zona di disponibilità. Utilizza più zone di disponibilità per migliorare la disponibilità.	Aggiungi le istanze DB a più zone di disponibilità del tuo cluster DB.	No	Elevata disponibilità di Amazon Aurora
Istanze DB nei cluster con dimensioni di istanze eterogenee	Ti consigliamo di utilizzare la stessa classe e dimensione di istanze DB per tutte le istanze DB del tuo cluster di database.	Usa la stessa classe e dimensione di istanza per tutte le istanze DB del tuo cluster di database.	Sì	Replica con Amazon Aurora
Istanze DB nei cluster con classi di istanze eterogenee	Ti consigliamo di utilizzare la stessa classe e dimensione di istanze DB per tutte le istanze DB del tuo cluster di database.	Usa la stessa classe e dimensione di istanza per tutte le istanze DB del tuo cluster di database.	Sì	Replica con Amazon Aurora
Istanze DB nei cluster con gruppi di parametri eterogenei	Si consiglia che tutte le istanze DB del cluster DB utilizzino lo stesso gruppo di parametri DB.	Associate l'istanza DB al gruppo di parametri DB associato all'istanza writer nel vostro cluster DB.	No	

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I cluster Amazon RDS DB dispongono di un'istanza DB	Aggiungi almeno un'altra istanza DB al tuo cluster DB per migliorare la disponibilità e le prestazioni.	Aggiungi un'istanza DB reader al tuo cluster DB.	No	Elevata disponibilità di Amazon Aurora
Performance Insights è disattivato	Performance Insights monitora il carico dell'istanza DB per aiutarti ad analizzare e risolvere i problemi di prestazioni del database. Ti consigliamo di attivare Performance Insights.	Attivare Performance Insights.	No	Monitoraggio del carico DB con Performance Insights su Amazon Aurora
RDSrisorse: è richiesto l'aggiornamento delle versioni principali	I database con l'attuale versione principale per il motore DB non saranno supportati. Ti consigliamo di eseguire l'aggiornamento alla versione principale più recente che include nuove funzionalità e miglioramenti.	Esegui l'aggiornamento alla versione principale più recente per il motore DB.	Sì	Aggiornamenti di Amazon Aurora Creazione di una distribuzione blu/verde in Amazon

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I cluster DB supportano solo un volume fino a 64 TiB	I tuoi cluster DB supportano volumi fino a 64 TiB. Le versioni più recenti del motore supportano volumi fino a 128 TiB per il cluster DB. Ti consigliamo di aggiornare la versione del motore del tuo cluster DB alle versioni più recenti per supportare volumi fino a 128 TiB.	Aggiorna la versione del motore dei tuoi cluster DB per supportare volumi fino a 128 TiB.	Sì	Limiti di dimensione Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Cluster DB con tutte le istanze di lettura nella stessa zona di disponibilità	Le zone di disponibilità (AZs) sono ubicazioni distinte l'una dall'altra per garantire l'isolamento in caso di interruzioni all'interno di ciascuna regione. AWS Si consiglia di distribuire l'istanza principale e le istanze di lettura nel cluster DB su più istanze AZs per migliorare la disponibilità del cluster DB. Puoi creare un cluster Multi-AZ utilizzando la console di AWS gestione o Amazon al RDS API momento della creazione del cluster. AWS CLI È possibile modificare il cluster Aurora esistente in un cluster Multi-AZ aggiungendo una nuova istanza di lettura e specificando una AZ diversa.	Il cluster DB ha tutte le istanze di lettura nella stessa zona di disponibilità. Ti consigliamo di distribuire le istanze del lettore su più zone di disponibilità. La distribuzione aumenta la disponibilità e migliora i tempi di risposta riducendo la latenza di rete tra i client e il database.	No	Elevata disponibilità di Amazon Aurora

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
I parametri di memoria DB sono diversi da quelli predefiniti	I parametri di memoria delle istanze DB sono significativamente diversi dai valori predefiniti. Queste impostazioni possono influire sulle prestazioni e causare errori. Si consiglia di ripristinare i parametri di memoria personalizzati per l'istanza DB ai valori predefiniti nel gruppo di parametri DB.	Reimposta i parametri di memoria ai valori predefiniti.	No	

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Il parametro della cache delle query è attivato	Quando le modifiche richiedono l'eliminazione della cache delle query, l'istanza DB sembrerà bloccarsi. La maggior parte dei carichi di lavoro non beneficia della cache delle query. La cache delle query è stata rimossa da My SQL 8.0 e versioni successive. Si consiglia di impostare il parametro query_cache_type su 0.	Imposta il valore del `query_cache_type` parametro su nei gruppi di parametri del database`0`.	Sì	

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`log_output` il parametro è impostato su tabella	Quando `log_output` è impostato su`TABLE`, viene utilizzato più spazio di archiviazione rispetto a quando `log_output` è impostato su`FILE`. Si consiglia di impostare il parametro su`FILE`, per evitare di raggiungere il limite di dimensione di archiviazione. `FILE`Impostato come predefinito in My SQL 8.4 e versioni successive.	Imposta il valore del `log_output` parametro su `FILE` nei gruppi di parametri del database.	No	File di log del database Aurora MySQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`autovacuum` il parametro è disattivato	Il parametro autovacuum è disattivato per i cluster DB delle . La disattivazione dell'autovacuum aumenta il volume della tabella e dell'indice e influisce sulle prestazioni. Ti consigliamo di attivare l'autovacuum nei gruppi di parametri del database.	Attiva il parametro autovacuum nei gruppi di parametri del cluster DB.	No	Informazioni sull'autovacuum negli ambienti Amazon RDS for Postgre SQL nel blog Database AWS
`synchronous_commit` il parametro è disattivato	Quando `synchronous_commit` il parametro è disattivato, i dati possono andare persi in caso di arresto anomalo del database. La durabilità del database è a rischio. Consigliamo di attivare il parametro `synchronous_commit` .	Attiva i `synchronous_commit` parametri nei gruppi di parametri del database.	Sì	SQLParametri di Amazon Aurora Postgre: replica, sicurezza e registrazione nel blog del database AWS

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`track_counts` il parametro è disattivato	Quando il `track_counts` parametro è disattivato, il database non raccoglie le statistiche sull'attività del database. La funzione di autovacuum richiede che queste statistiche funzionino correttamente. Consigliamo di impostare il parametro `track_counts` su 1.	Imposta `track_counts` il parametro su`1`.	No	Statistiche di runtime per Postgre SQL
`enable_indexonlyscan` il parametro è disattivato	Il pianificatore o l'ottimizzatore delle query non possono utilizzare il tipo di piano di scansione basato solo sull'indice quando è disattivato. Si consiglia di impostare il valore del `enable_indexonlyscan` parametro su. 1	Imposta il valore del `enable_indexonlyscan` parametro su`1`.	No	Configurazione del metodo Planner per Postgre SQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`enable_indexscan` il parametro è disattivato	Il pianificatore o l'ottimizzatore delle query non possono utilizzare il tipo di piano di scansione dell'indice quando è disattivato. Si consiglia di impostare il `enable_indexscan` valore su. 1	Imposta il valore del `enable_indexscan` parametro su1.	No	Configurazione del metodo Planner per Postgre SQL
`innodb_flush_log_at_trx` il parametro è disattivato	Il valore del `innodb_flush_log_at_trx` parametro dell'istanza DB non è un valore sicuro. Questo parametro controlla la persistenza delle operazioni di commit su disco. Consigliamo di impostare il parametro `innodb_flush_log_at_trx` su 1.	Imposta il valore del `innodb_flush_log_at_trx` parametro su1.	No	Configurazione della frequenza di svuotamento del buffer dei registri

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`innodb_stats_persistent` il parametro è disattivato	L'istanza database non è configurata per memorizzare le statistiche InnoDB sul disco. Quando le statistiche non vengono archiviate, vengono ricalcolate ogni volta che l'istanza si riavvia e si accede alla tabella. Ciò porta a variazioni nel piano di esecuzione delle query. Puoi modificare il valore di questo parametro globale a livello di tabella. Si consiglia di impostare il valore del `innodb_stats_persistent` parametro su`ON`.	Imposta il valore del `innodb_stats_persistent` parametro su`ON`.	No	

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`innodb_open_files` il parametro è basso	Il `innodb_open_files` parametro controlla il numero di file che InnoDB può aprire contemporaneamente. InnoDB apre tutti i file di log e di tablespace di sistema quando mysqld è in esecuzione. Il valore del numero massimo di file dell'istanza database che InnoDB può aprire contemporaneamente non è sufficiente. Consigliamo di impostare il parametro `innodb_open_files` almeno sul valore 65.	Imposta il `innodb_open_files` parametro su un valore minimo di. 65	Sì	InnoDB apre i file per My SQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`max_user_connections` il parametro è basso	Il valore del numero massimo di connessioni simultanee per ogni account di database dell'istanza database non è sufficiente. Si consiglia di impostare il `max_user_connections` parametro su un numero maggiore di5.	Aumentate il valore del `max_user_connections` parametro portandolo a un numero maggiore di5.	Sì	Impostazione dei limiti delle risorse dell'account per My SQL
Le repliche di lettura sono aperte in modalità scrivibile	L'istanza DB ha una replica di lettura in modalità scrivibile, che consente gli aggiornamenti dai client. Ti consigliamo di impostare il `read_only` parametro in `TrueIfReplica` modo che le repliche di lettura non siano in modalità scrivibile.	Imposta il valore del `read_only` parametro su. `TrueIfReplica`	No	

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`innodb_default_row_format` l'impostazione dei parametri non è sicura	L'istanza DB presenta un problema noto: una tabella creata in una SQL versione My precedente alla 8.0.26 con `COMPACT` o `REDUNDANT` sarà inaccessibile e irrecuperabile quando l'`row_format` indice supera i 767 byte. Si `innodb_default_row_format` consiglia di impostare il valore del parametro su. `DYNAMIC`	Imposta il valore del `innodb_default_row_format` parametro su`DYNAMIC`.	No	Modifiche in My SQL 8.0.26

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
`general_logging` il parametro è attivato	La registrazione generale è attivata per l'istanza DB. Questa impostazione è utile per la risoluzione dei problemi del database. Tuttavia, l'attivazione della registrazione generale aumenta la quantità di operazioni di I/O e lo spazio di archiviazione allocato, il che potrebbe causare conflitti e un peggioramento delle prestazioni. Verifica i tuoi requisiti per l'utilizzo generale della registrazione. Si consiglia di impostare il valore del `general_logging` parametro su`0`.	Verifica i tuoi requisiti per l'utilizzo generale della registrazione. Se non è obbligatorio, ti consigliamo di impostare il valore del `general_logging` parametro su`0`.	No	Panoramica dei registri di database Aurora MySQL

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
Cluster DB con un provisioning insufficiente per il carico di lavoro di lettura	Si consiglia di aggiungere un'istanza DB reader al cluster DB con la stessa classe di istanza e le stesse dimensioni dell'istanza DB writer nel cluster. La configurazione corrente prevede un'istanza DB con un carico di database costantemente elevato causato principalmente da operazioni di lettura. Distribuisci queste operazioni aggiungendo un'altra istanza DB al cluster e indirizzando il carico di lavoro di lettura all'endpoint di sola lettura del cluster DB.	Aggiungi un'istanza DB reader al cluster.	No	Aggiunta di repliche di Aurora a un cluster di database Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora RDSPrezzi Amazon

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiest	Informazioni aggiuntive
RDSistanza la cui capacità di memoria di sistema è insufficiente	Si consiglia di ottimizzare le query in modo da utilizzare meno memoria o utilizzare un tipo di istanza DB con una maggiore quantità di memoria allocata. Quando la memoria dell'istanza sta esaurendo, le prestazioni del database ne risentono.	Utilizza un'istanza DB con una maggiore capacità di memoria	Sì	Scalabilità verticale e orizzontale dell'RDSistanza Amazon sul database Blog AWS Tipi di RDS istanze Amazon RDSPrezzi Amazon
RDSistanza con dotazione insufficiente rispetto alla capacità del sistema CPU	Ti consigliamo di ottimizzare le query in modo da utilizzare meno CPU o di modificare l'istanza DB per utilizzare una classe di istanza DB con allocazioni più elevate. vCPUs Le prestazioni del database potrebbero diminuire quando un'istanza DB si sta esaurendo. CPU	Utilizza un'istanza DB con una CPU capacità maggiore	Sì	Scalabilità verticale e orizzontale dell'RDSistanza Amazon sul database Blog AWS Tipi di RDS istanze Amazon RDSPrezzi Amazon

Tipo	Descrizione	Raccomandazione	Tempo di inattività richiesto	Informazioni aggiuntive
RDSLe risorse non utilizzano correttamente il pool di connessioni	Ti consigliamo di abilitare Amazon RDS Proxy per raggruppare e condividere in modo efficiente le connessioni al database esistenti. Se stai già utilizzando un proxy per il tuo database, configuralo correttamente per migliorare il pool di connessioni e il bilanciamento del carico su più istanze DB. RDS Il proxy può aiutare a ridurre il rischio di esaurimento della connessione e i tempi di inattività, migliorando al contempo la disponibilità e la scalabilità.	Abilita il RDS proxy o modifica la configurazione proxy esistente	No	Scalabilità verticale e orizzontale dell'RDSistanza Amazon sul database Blog AWS Proxy Amazon RDS per Aurora Prezzi Amazon RDS Proxy

Visualizzazione delle metriche nella console Amazon RDS

Amazon RDS si integra con Amazon CloudWatch per visualizzare una varietà di parametri delle istanze DB del RDS cluster Aurora nella console. RDS Alcuni parametri vengono applicati a livello di cluster, mentre altri si applicano a livello di istanza. Per le descrizioni dei parametri a livello di istanza e cluster, consulta Riferimento alle metriche per .

Per il cluster database Aurora, vengono monitorate le seguenti categorie di metriche:

- CloudWatch— Mostra i CloudWatch parametri di Amazon per Aurora a cui puoi accedere dalla console. RDS Puoi accedere a queste metriche anche nella console. CloudWatch Ogni parametro include un grafico che mostra il parametro monitorato in un periodo di tempo specifico. Per un elenco di CloudWatch metriche, consulta.
- Monitoraggio avanzato: mostra un riepilogo delle metriche del sistema operativo quando l' DB del cluster RDS DB Aurora DB ha attivato il monitoraggio avanzato. RDSfornisce i parametri di Enhanced Monitoring al tuo account Amazon CloudWatch Logs. Ogni parametro del sistema operativo include un grafico che visualizza il parametro monitorato in un periodo di tempo specifico. Per una panoramica, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato. Per un elenco di parametri di monitoraggio avanzato, consulta Metriche OS nel monitoraggio avanzato.
- OS Process list (Elenco processi sistema operativo): mostra i dettagli di ogni processo in esecuzione nel cluster di database.
- Performance Insights: apre la dashboard di Amazon RDS Performance Insights per un'istanza DB nel cluster Aurora DB. Performance Insights non è supportato a livello di cluster.Per una panoramica su Performance Insights, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora. Per un elenco di parametri di Performance Insights, consulta CloudWatch Parametri Amazon per Amazon RDS Performance Insights.

Amazon RDS ora fornisce una visualizzazione consolidata di Performance Insights e delle CloudWatch metriche nella dashboard di Performance Insights. Performance Insights deve essere attivato affinché il cluster database possa utilizzare questa visualizzazione. È possibile scegliere la nuova visualizzazione di monitoraggio nella scheda Monitoraggio o Performance Insights nel pannello di navigazione. Per visualizzare le istruzioni per scegliere questa visualizzazione, consultare Visualizzazione delle metriche combinate con la dashboard Performance Insights.

Visualizzazione delle metriche combinate con la dashboard Performance Insights

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights. Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
> Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
> Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Amazon RDS offre una visualizzazione consolidata di Performance Insights e dei CloudWatch parametri per l'istanza DB nella dashboard di Performance Insights. È possibile utilizzare il pannello di controllo preconfigurato o crearne uno personalizzato. Il pannello di controllo preconfigurato fornisce le metriche più comunemente utilizzate per diagnosticare i problemi relativi alle prestazioni di un motore di database. In alternativa, è possibile creare un pannello di controllo personalizzato contenente le metriche di un motore di database che soddisfi i requisiti di analisi definiti. Quindi, utilizza questa dashboard per tutte le istanze DB di quel tipo di motore di database presenti nel tuo account. AWS

Puoi scegliere la visualizzazione di monitoraggio nella scheda Monitoraggio o Performance Insights nel riquadro di navigazione.

Performance Insights deve essere attivato affinché il cluster database possa visualizzare le metriche combinate nel pannello di controllo di Performance Insights. Per ulteriori informazioni sull'attivazione di Performance Insights, consultare Attivazione e disattivazione di Performance Insights per Aurora.

Nelle sezioni seguenti, puoi imparare a visualizzare Performance Insights e CloudWatch metriche.

Argomenti

- Scelta della nuova visualizzazione di monitoraggio dalla scheda Monitoraggio
- Scelta della nuova visualizzazione di monitoraggio dalla pagina Performance Insights
- Creazione di una dashboard personalizzata con Performance Insights
- Scelta della dashboard preconfigurata con Performance Insights

Scelta della nuova visualizzazione di monitoraggio dalla scheda Monitoraggio

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights. Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
> Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.

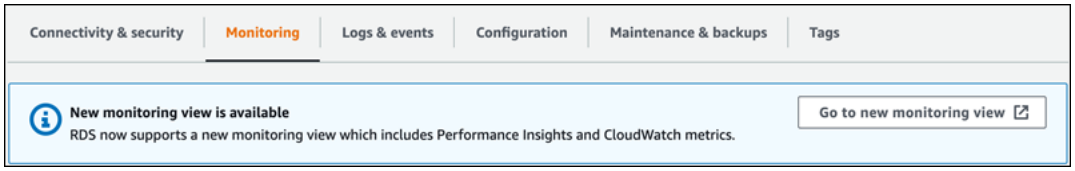
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Dalla console Amazon RDS, puoi scegliere la nuova vista di monitoraggio per visualizzare Performance Insights e CloudWatch metriche per la tua istanza DB.

Per scegliere la nuova visualizzazione di monitoraggio nella scheda Monitoraggio

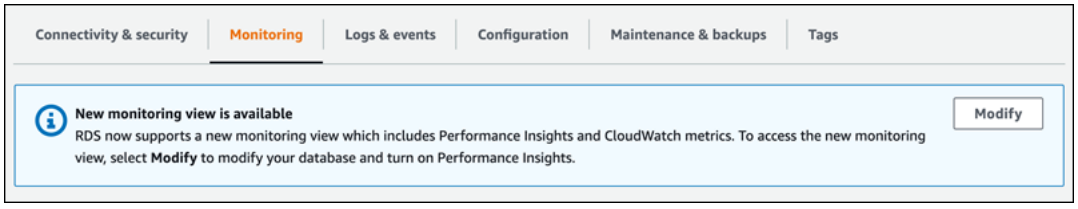
1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione a sinistra, scegliere Database.
3. Scegliere il cluster database Aurora che si desidera monitorare.
4. Scorrere verso il basso e selezionare Monitoraggio.

   Viene visualizzato un banner con l'opzione che consente di scegliere la nuova visualizzazione di monitoraggio. Nell'esempio seguente è rappresentato il banner per scegliere la nuova visualizzazione di monitoraggio.



5. Scegli Vai alla nuova visualizzazione di monitoraggio per aprire la dashboard di Performance Insights con Performance Insights e CloudWatch metriche per l'.
6. (Facoltativo) Se Performance Insights è disattivato per l'istanza database, viene visualizzato un banner con l'opzione per modificare l'istanza database e attivare Performance Insights.

   L'esempio seguente mostra il banner per modificare l'istanza database nella scheda Monitoraggio.



Scegliere Modifica per modificare l'istanza database e attivare Performance Insights. Per ulteriori informazioni sull'attivazione di Performance Insights, consultare Attivazione e disattivazione di Performance Insights per Aurora.

Scelta della nuova visualizzazione di monitoraggio dalla pagina Performance Insights

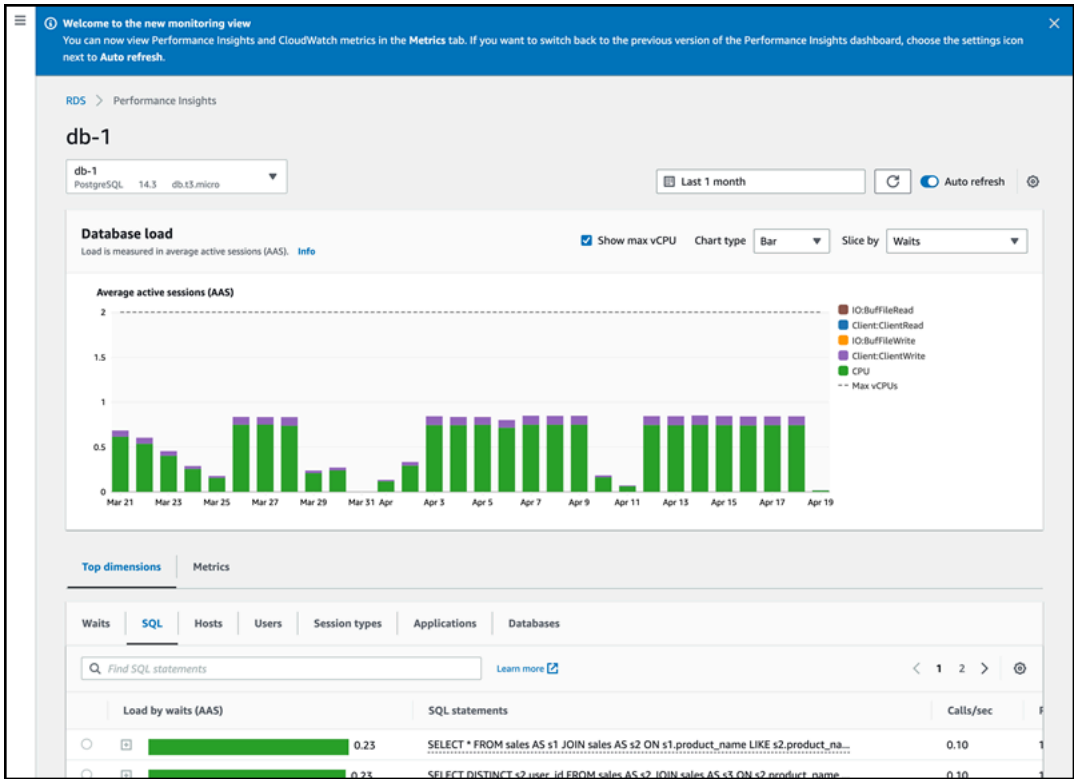
Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
Se non si intraprende alcuna azione, DB che raggruppano in cluster utilizzando Performance Insights utilizzeranno per impostazione predefinita la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Dalla console Amazon RDS, puoi scegliere la nuova vista di monitoraggio per visualizzare Performance Insights e CloudWatch metriche per la tua istanza DB.

Per scegliere la nuova visualizzazione di monitoraggio con Performance Insights nel riquadro di navigazione

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza DB per visualizzare la dashboard di Performance Insights che mostra sia Performance Insights che le CloudWatch metriche per la tua istanza DB.



Creazione di una dashboard personalizzata con Performance Insights

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights. Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre

2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
Se non si intraprende alcuna azione, DB che raggruppano in cluster utilizzando Performance Insights utilizzeranno per impostazione predefinita la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Nella nuova visualizzazione di monitoraggio, è possibile creare un pannello di controllo personalizzato con le metriche necessarie per soddisfare gli specifici requisiti di analisi.

Puoi creare una dashboard personalizzata selezionando Performance Insights e CloudWatch metriche per la tua istanza DB. Puoi utilizzare questa dashboard personalizzata per altre istanze DB dello stesso tipo di motore di database nel tuo AWS account.

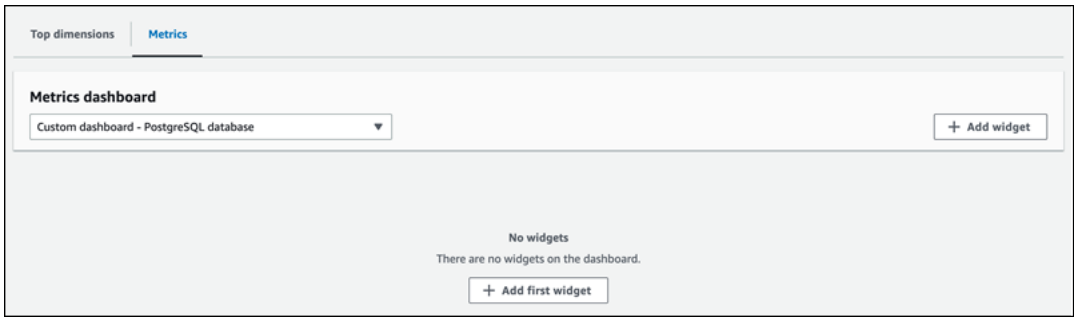
> **Note**
>
> Il pannello di controllo personalizzato supporta fino a 50 metriche.

Usare il menu delle impostazioni del widget per modificare o eliminare il pannello di controllo e spostare o ridimensionare la finestra del widget.

Per creare una dashboard personalizzata con Performance Insights nel riquadro di navigazione

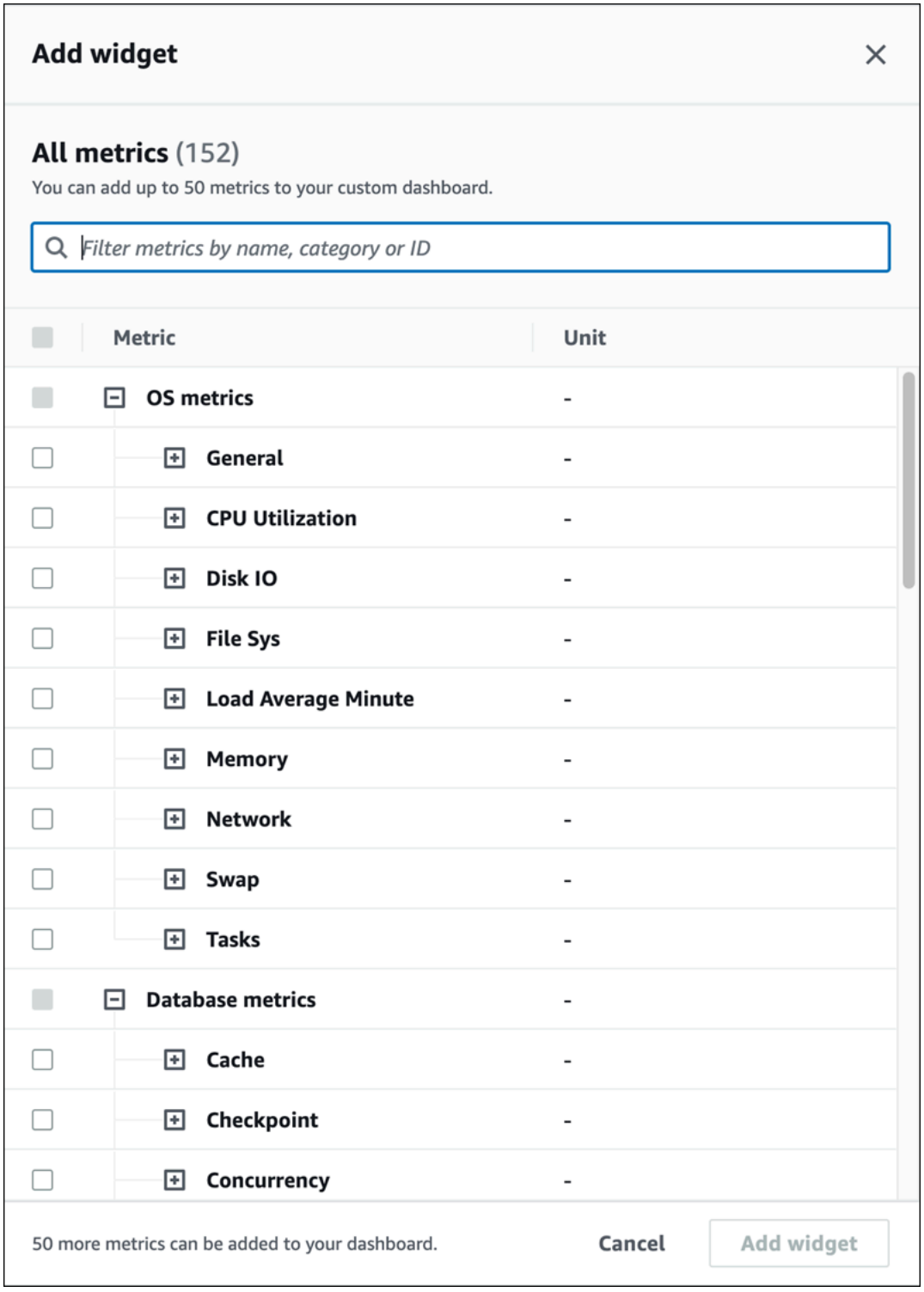
1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.
4. Scorrere verso il basso fino alla scheda Metriche nella finestra.

5. Selezionare il pannello di controllo personalizzato nell'elenco a discesa. Nell'esempio seguente viene illustrata la creazione del pannello di controllo personalizzato.



6. Scegli Aggiungi widget per aprire la finestra Aggiungi widget. Puoi aprire e visualizzare i parametri del sistema operativo (OS), i parametri del database e i parametri disponibili nella CloudWatch finestra.

   L'esempio seguente mostra la finestra Aggiungi widget con le metriche.



7. Selezionare le metriche da visualizzare nel pannello di controllo e scegliere Aggiungi widget. È possibile utilizzare il campo di ricerca per cercare una metrica specifica.

   Le metriche selezionate vengono visualizzate nel pannello di controllo.

8. (Facoltativo) Per modificare o eliminare il pannello di controllo, scegliere l'icona delle impostazioni in alto a destra del widget, quindi selezionare una delle seguenti azioni nel menu.

   - Modifica: consente di modificare l'elenco delle metriche nella finestra. Scegliere Aggiorna widget dopo aver selezionato le metriche da visualizzare nel pannello di controllo.
   - Elimina: consente di eliminare il widget. Nella finestra di conferma scegliere Elimina.

Scelta della dashboard preconfigurata con Performance Insights

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
> Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
> Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
> Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

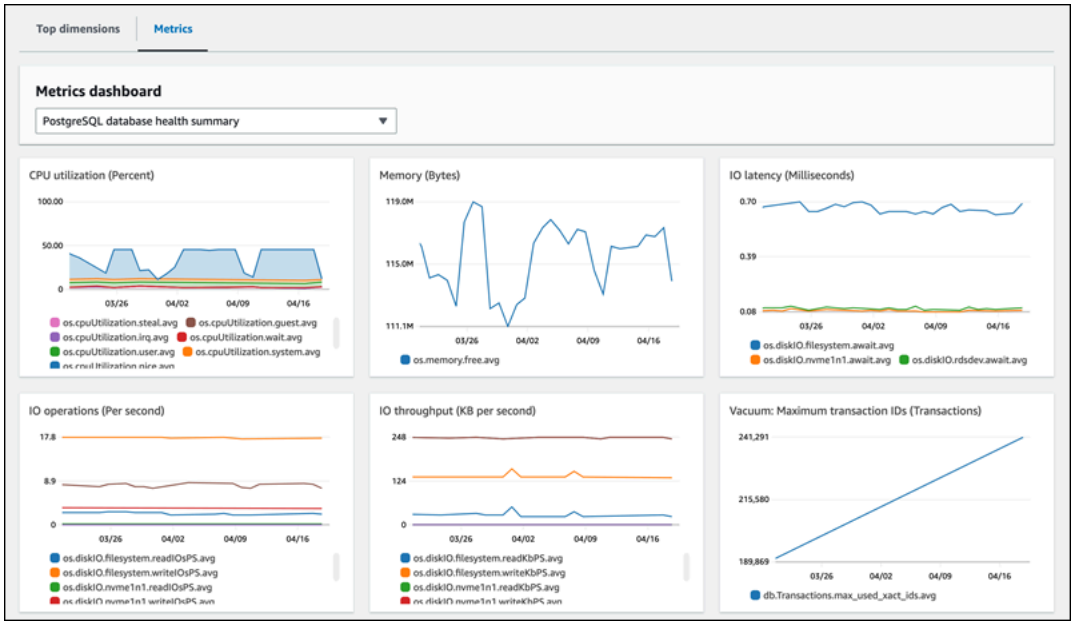
Nel pannello di controllo preconfigurato è possibile visualizzare le metriche di più frequente utilizzo. Questo pannello di controllo aiuta a diagnosticare i problemi di prestazioni di un motore di database e a ridurre il tempo medio di ripristino da ore a minuti.

> **Note**
>
> Questo pannello di controllo non può essere modificato.

Per scegliere la dashboard preconfigurata con Performance Insights nel riquadro di navigazione

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.
4. Scorrere verso il basso fino alla scheda Metriche nella finestra.
5. Selezionare un pannello di controllo preconfigurato nell'elenco a discesa.

   Nel pannello di controllo è possibile visualizzare le metriche per l'istanza database. Nell'esempio seguente viene illustrato un pannello di controllo preconfigurato con le metriche.



Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch

Amazon CloudWatch è un archivio di metriche. Il repository raccoglie ed elabora i dati non elaborati da Amazon Aurora in parametri leggibili quasi in tempo reale. Per un elenco completo dei parametri di Amazon Aurora inviati CloudWatch, Aurora.

Argomenti

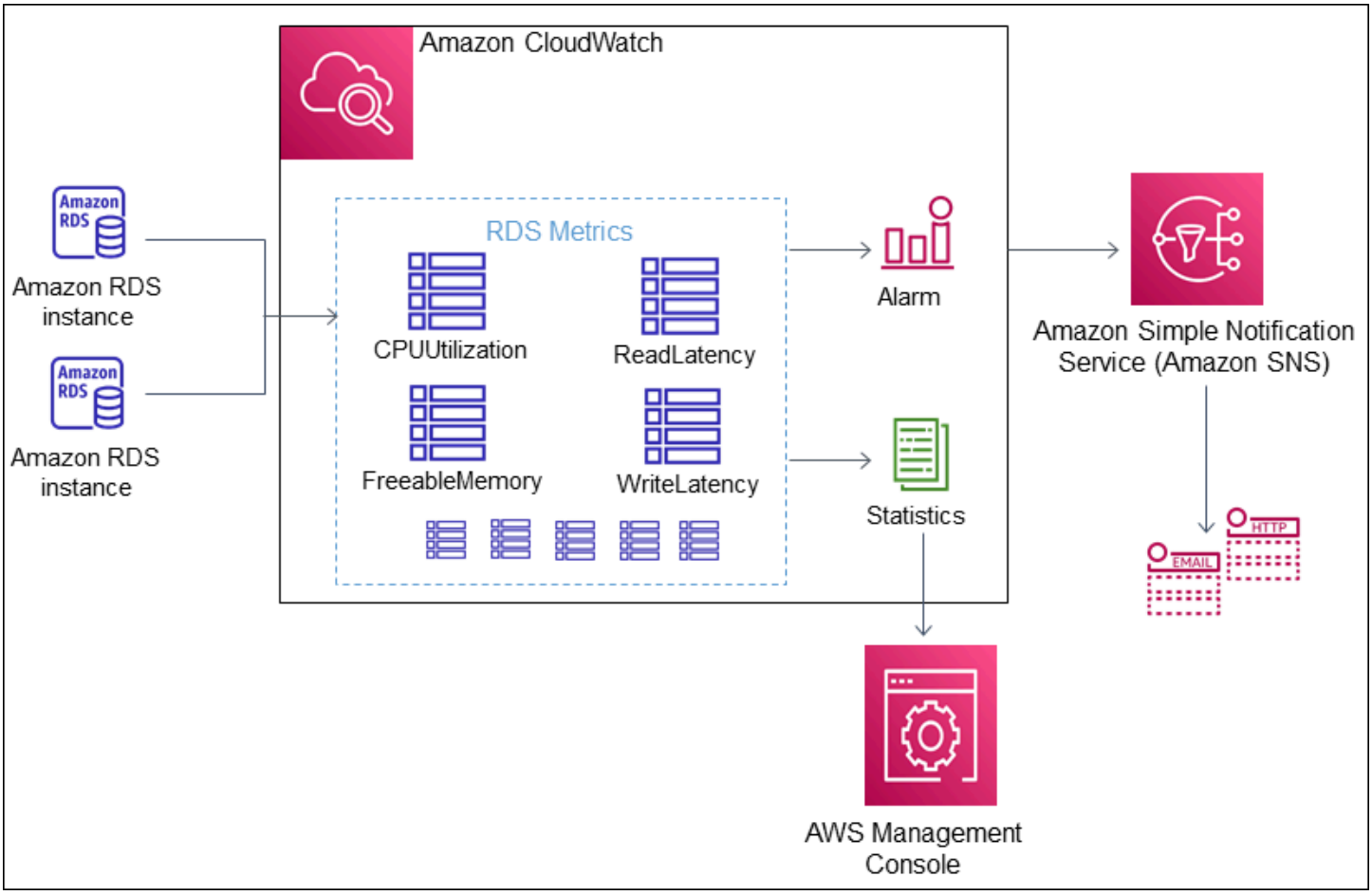
- Panoramica di Amazon Aurora e Amazon CloudWatch
- Visualizzazione delle metriche del cluster di DB nella CloudWatch console e AWS CLI
- Esportazione delle metriche di Performance Insights in CloudWatch
- Creazione di CloudWatch allarmi per monitorare

Panoramica di Amazon Aurora e Amazon CloudWatch

Per impostazione predefinita, Amazon Aurora invia automaticamente i dati CloudWatch delle metriche in periodi di 1 minuto. Ad esempio, il parametro `CPUUtilization` registra la percentuale di utilizzo della CPU per un'istanza database nel tempo. I punti di dati con un periodo di 60 secondi (1 minuto) sono disponibili per 15 giorni. Ciò significa che è possibile accedere alle informazioni della cronologia e visualizzare le prestazioni del servizio o dell'applicazione Web.

Ora puoi esportare i dashboard dei parametri di Performance Insights da Amazon RDS ad Amazon. CloudWatch Puoi esportare le dashboard delle metriche preconfigurate o personalizzate come una nuova dashboard o aggiungerle a una dashboard esistente. CloudWatch La dashboard esportata è disponibile per la visualizzazione nella console. CloudWatch Per ulteriori informazioni su come esportare i dashboard delle metriche di Performance Insights in CloudWatch, consulta. Esportazione delle metriche di Performance Insights in CloudWatch

Come mostrato nel diagramma seguente, puoi impostare allarmi per le tue metriche. CloudWatch Ad esempio, puoi creare un allarme che segnali quando l'utilizzo della CPU per un'istanza è superiore al 70%. Puoi configurare l'invio Amazon Simple Notification Service di un'e-mail quando viene superata la soglia.



Amazon RDS pubblica i seguenti tipi di metriche su Amazon: CloudWatch

- Parametri Aurora a livello di cluster e di istanza

  Per una tabella di questi parametri, consulta CloudWatch Parametri Amazon per Amazon Aurora.
- Parametri Performance Insights

  Per una tabella di questi parametri, consulta CloudWatch Parametri Amazon per Amazon RDS Performance Insights e Parametri contatore di Performance Insights.
- Metriche di monitoraggio avanzate (pubblicate su Amazon CloudWatch Logs)

  Per una tabella di questi parametri, consulta Metriche OS nel monitoraggio avanzato.
- Metriche di utilizzo per le quote del servizio Amazon RDS nel tuo Account AWS

  Per una tabella di questi parametri, consulta Metriche CloudWatch di utilizzo di . Per ulteriori informazioni sulle quote di Amazon RDS, consulta Quote e vincoli per Amazon Aurora.

Per ulteriori informazioni su CloudWatch, consulta What is Amazon CloudWatch? nella Amazon CloudWatch User Guide. Per ulteriori informazioni sulla conservazione CloudWatch delle metriche, consulta Metrics retention.

Visualizzazione delle metriche del cluster di DB nella CloudWatch console e AWS CLI

Di seguito, puoi trovare dettagli su come visualizzare le metriche per la tua istanza DB utilizzando CloudWatch. Per informazioni sul monitoraggio delle metriche per il sistema operativo dell'istanza DB in tempo reale tramite CloudWatch Logs, consulta. Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato

Quando utilizzi le risorse Aurora, Amazon Aurora invia parametri e dimensioni ad Amazon ogni minuto. CloudWatch

Ora puoi esportare i dashboard dei parametri di Performance Insights da RDS Amazon ad Amazon CloudWatch e visualizzarli nella console. CloudWatch Per ulteriori informazioni su come esportare i dashboard delle metriche di Performance Insights in CloudWatch, consulta. Esportazione delle metriche di Performance Insights in CloudWatch

Utilizza le seguenti procedure per visualizzare i parametri per Aurora CloudWatch nella console e. CLI

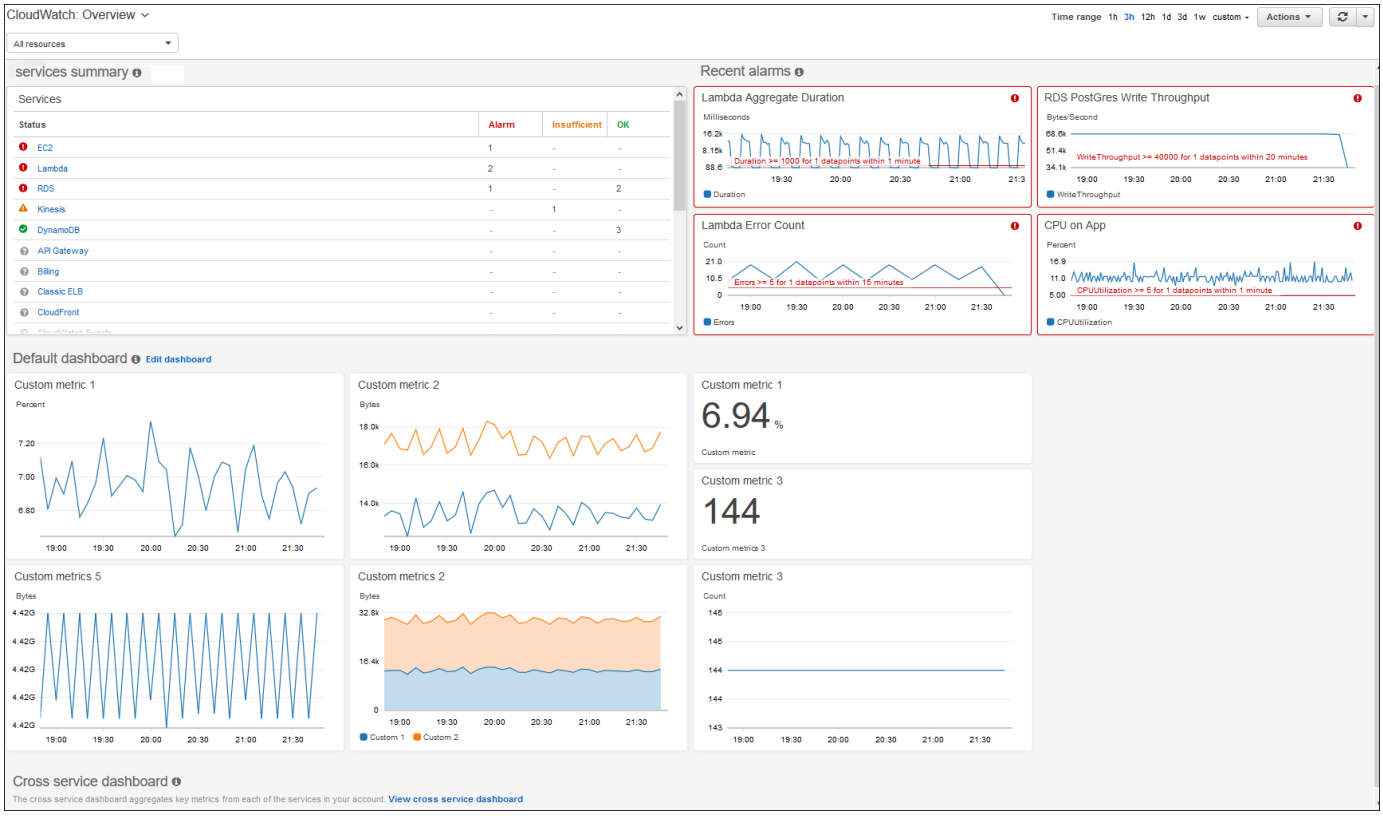
Console

Per visualizzare i parametri utilizzando la console Amazon CloudWatch

I parametri vengono raggruppati prima in base allo spazio dei nomi del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni spazio dei nomi.

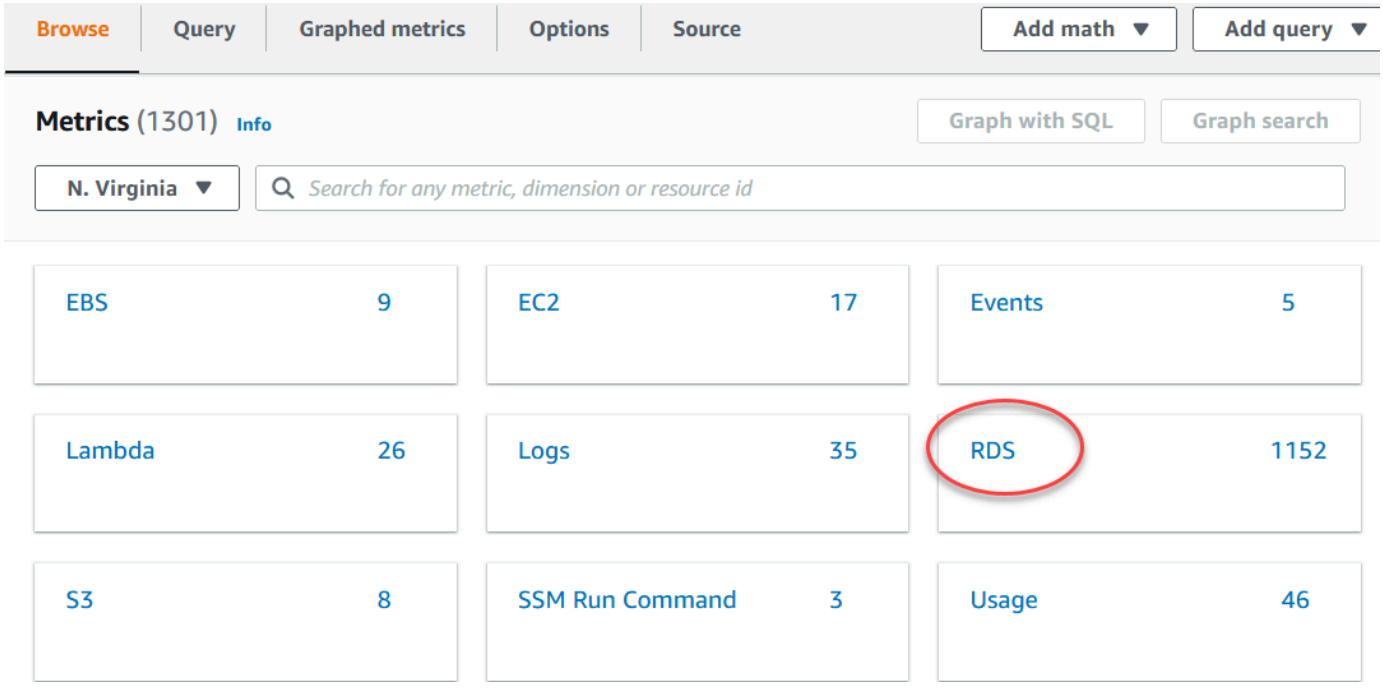
1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.

   Viene visualizzata la home page CloudWatch panoramica.



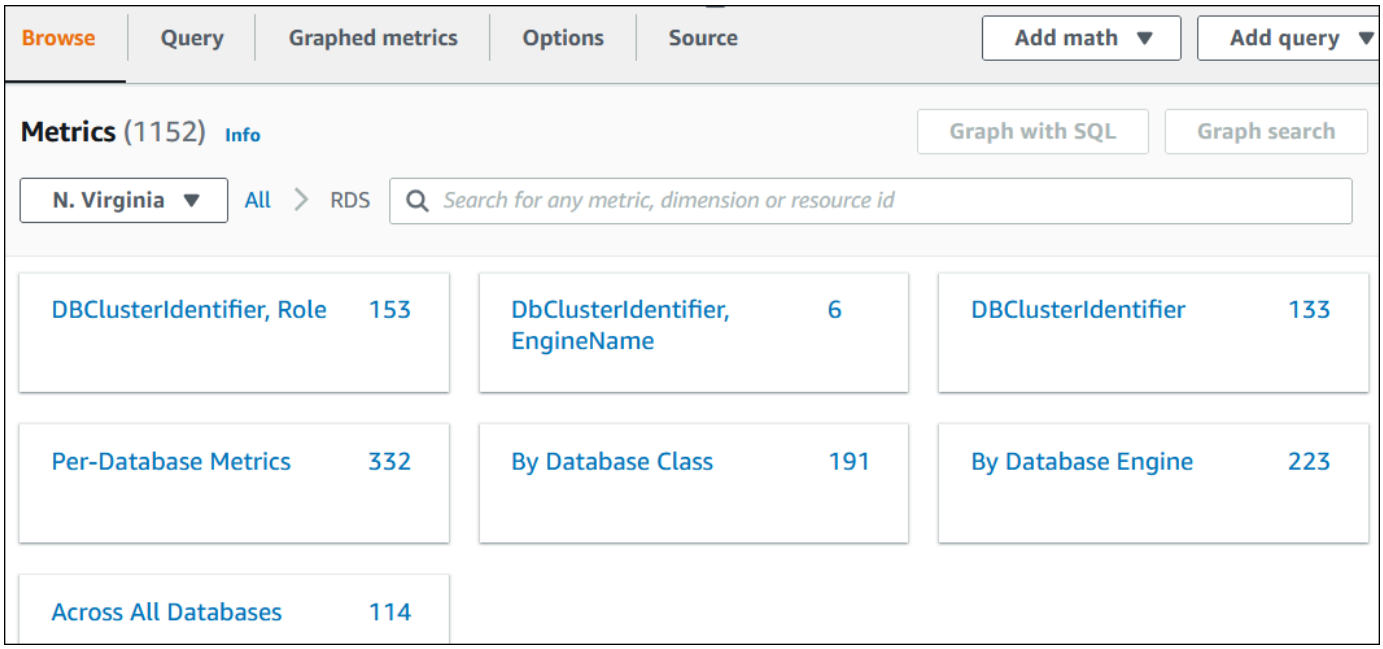
2. Se necessario, modifica Regione AWS. Dalla barra di navigazione, scegli Regione AWS dove si trovano AWS le tue risorse. Per ulteriori informazioni, consulta Regioni ed endpoint.

3. Nel pannello di navigazione, scegli Metrics (Parametri), quindi scegli All metrics (Tutti i parametri).

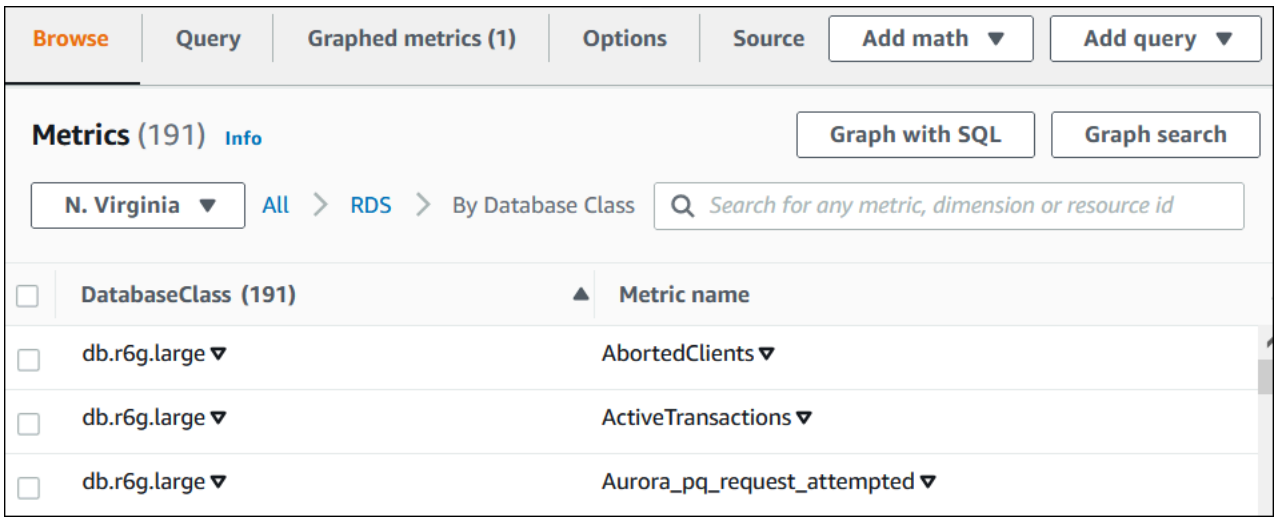


4. Scorri verso il basso e scegli lo spazio dei nomi della RDSmetrica.

   La pagina mostra le dimensioni di Amazon Aurora. Per una descrizione di queste dimensioni, consulta CloudWatch Dimensioni Amazon per Aurora.



5. Scegli una dimensione di parametro, ad esempio By Database Class (Per classe di database).

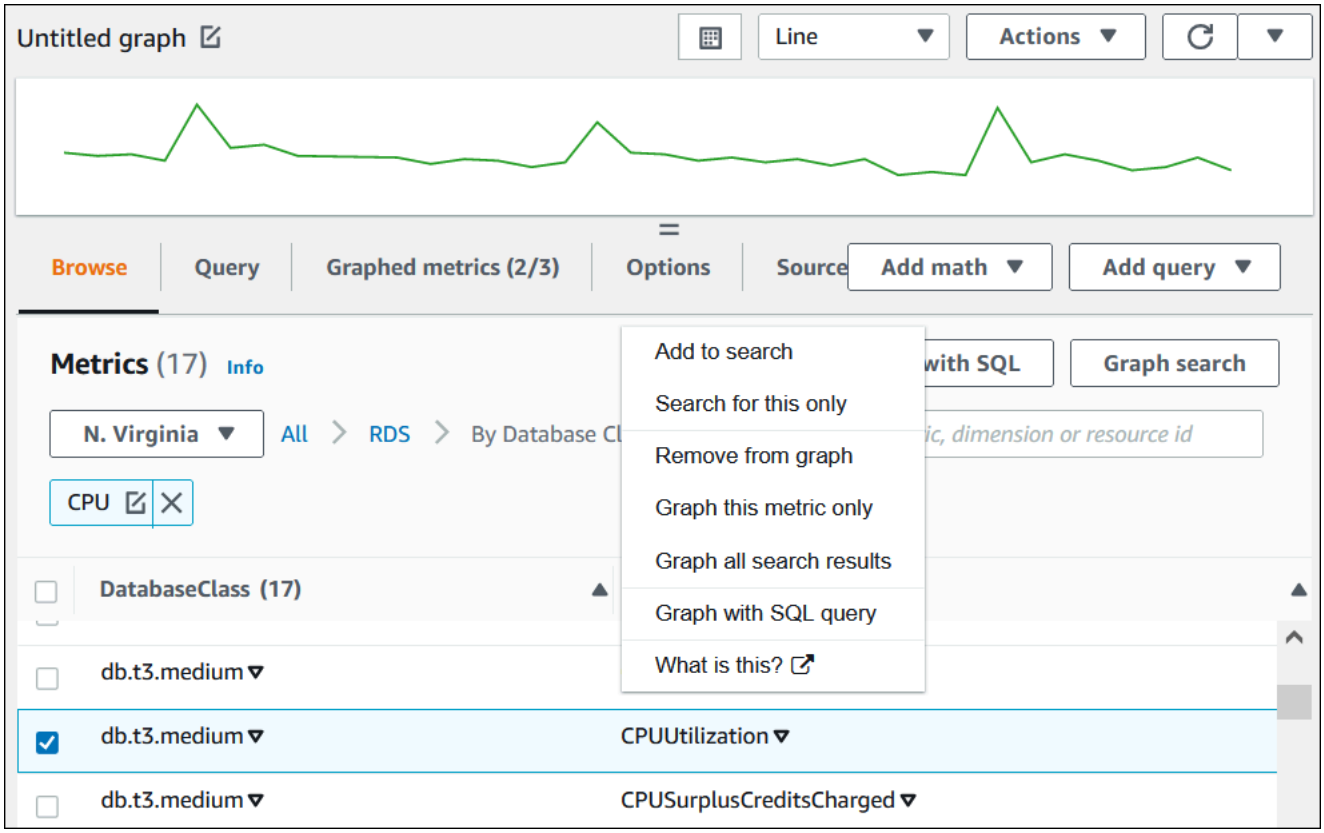


6. Effettua una delle seguenti operazioni:

   - Per ordinare i parametri, utilizza l'intestazione della colonna.
   - Per creare il grafico di un parametro, seleziona la casella di controllo accanto al parametro.
   - Per filtrare in base a una risorsa, scegli l'ID della risorsa e quindi Add to search (Aggiungi alla ricerca).

- Per filtrare in base a un parametro, scegli il nome del parametro e quindi Add to search (Aggiungi alla ricerca).

L'esempio seguente filtra in base alla classe db.t3.medium e rappresenta graficamente la metrica. CPUUtilization



Puoi trovare dettagli su come analizzare l'utilizzo delle risorse per Aurora SQL Postgre utilizzando le metriche. CloudWatch Per ulteriori informazioni, consulta Utilizzo dei CloudWatch parametri di Amazon per analizzare l'utilizzo delle risorse per Aurora PostgreSQL

AWS CLI

Per ottenere informazioni metriche utilizzando il, usa il AWS CLI comando. CloudWatch `list-metrics` Nell'esempio seguente, vengono elencati tutti i parametri nello spazio dei nomi `AWS/RDS`.

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Per ottenere dati metrici, utilizzate il comando. `get-metric-data`

L'esempio seguente ottiene `CPUUtilization` statistiche, ad esempio, `my-instance` su un periodo di 24 ore specifico, con una granularità di 5 minuti.

Crea un JSON file `CPU_metric.json` con i seguenti contenuti.

```
{
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
     "MetricStat" : {
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

Per LinuxmacOS, oUnix:

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

Per Windows:

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

L'output di esempio viene visualizzato come segue:

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
            "Timestamps": [
```

```
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
                ...
            ],
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
                13.02521708695145,
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Per ulteriori informazioni, consulta Ottenere statistiche per una metrica nella Amazon CloudWatch User Guide.

Esportazione delle metriche di Performance Insights in CloudWatch

Performance Insights ti consente di esportare il pannello di controllo delle metriche preconfigurato o personalizzato per la tua istanza DB su Amazon. CloudWatch Puoi esportare la dashboard delle metriche come nuova dashboard o aggiungerla a una dashboard esistente. CloudWatch Quando scegli di aggiungere la dashboard a una CloudWatch dashboard esistente, puoi creare un'etichetta di intestazione in modo che le metriche vengano visualizzate in una sezione separata della dashboard. CloudWatch

Puoi visualizzare la dashboard delle metriche esportate nella console. CloudWatch Se aggiungi nuove metriche a una dashboard delle metriche di Performance Insights dopo averla esportata, devi esportare nuovamente questa dashboard per visualizzare le nuove metriche nella console. CloudWatch

Puoi anche selezionare un widget metrico nella dashboard di Performance Insights e visualizzare i dati delle metriche nella CloudWatch console.

Per ulteriori informazioni sulla visualizzazione delle metriche nella CloudWatch console, consulta. Visualizzazione delle metriche del cluster di DB nella CloudWatch console e AWS CLI

Nelle sezioni seguenti, esporta le metriche di Performance Insights in CloudWatch un pannello di controllo nuovo o esistente e visualizza le metriche di Performance Insights in. CloudWatch

Argomenti

- Esportazione delle metriche di Performance Insights come nuova dashboard in CloudWatch
- Aggiungere metriche di Performance Insights a una dashboard esistente CloudWatch
- Visualizzazione di un widget metrico Performance Insights in CloudWatch

Esportazione delle metriche di Performance Insights come nuova dashboard in CloudWatch

Scegli una dashboard delle metriche preconfigurata o personalizzata dalla dashboard di Performance Insights ed esportala come nuova dashboard in. CloudWatch Puoi visualizzare la dashboard esportata nella console. CloudWatch

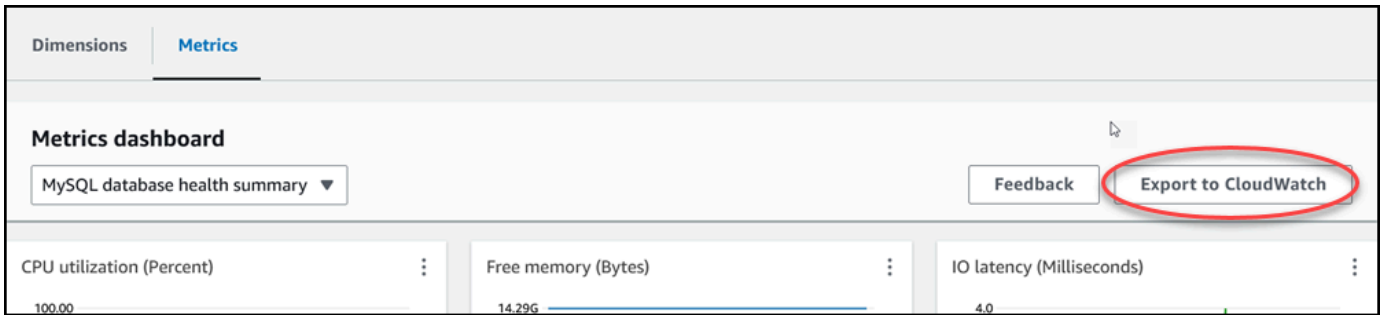
Per esportare una dashboard metrica di Performance Insights come nuova dashboard in CloudWatch

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

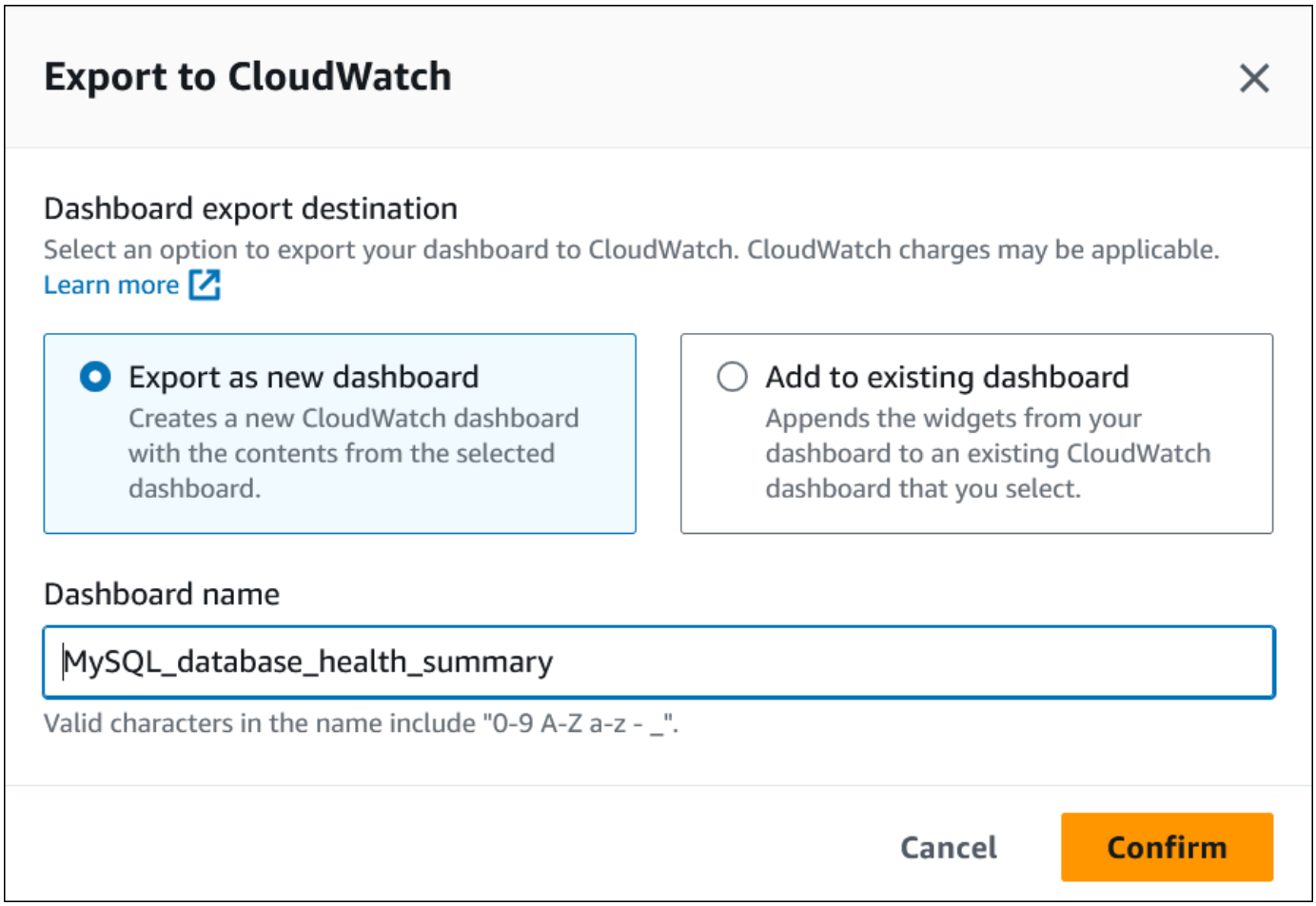
   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso e scegli Parametri.

   Per impostazione predefinita, viene visualizzata la dashboard preconfigurata con i parametri di Approfondimenti sulle prestazioni.
5. Scegli una dashboard preconfigurata o personalizzata, quindi scegli Esporta in CloudWatch.

   Viene visualizzata la CloudWatch finestra Esporta in.

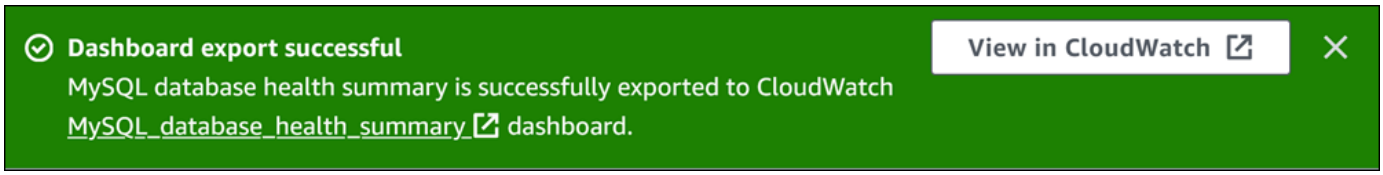


6. Scegli Esporta come nuova dashboard.



7. Immetti un nome per la nuova dashboard nel campo Nome della dashboard e scegli Conferma.

   Un banner mostra un messaggio al completamento dell'esportazione della dashboard.



8. Scegli il link o Visualizza CloudWatch nel banner per visualizzare la dashboard delle metriche nella CloudWatch console.

Aggiungere metriche di Performance Insights a una dashboard esistente CloudWatch

Aggiungi un pannello di controllo delle metriche preconfigurato o personalizzato a un pannello di controllo esistente. CloudWatch Puoi aggiungere un'etichetta alla dashboard delle metriche da visualizzare in una sezione separata della dashboard. CloudWatch

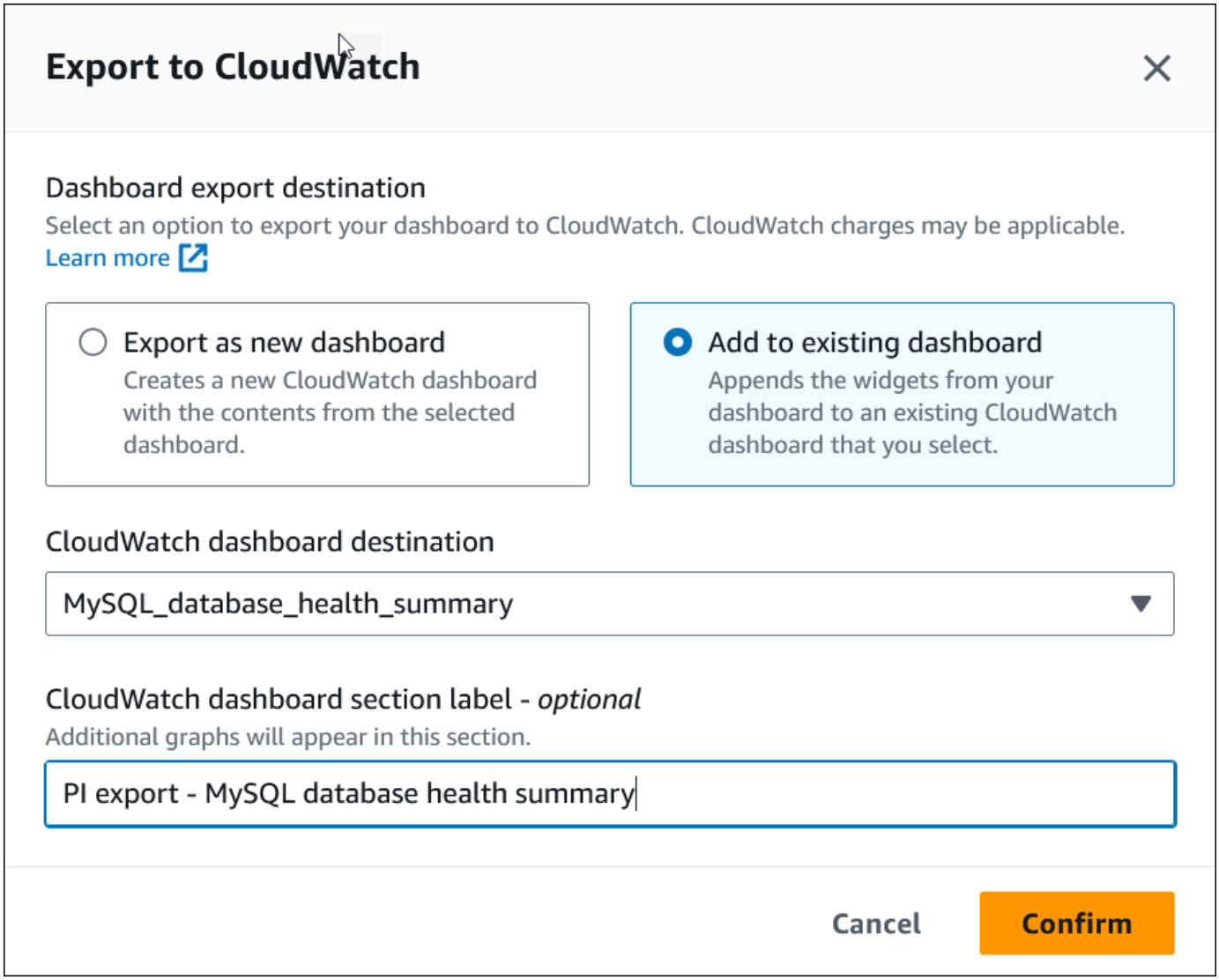
Per esportare le metriche in una dashboard esistente CloudWatch

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso e scegli Parametri.

   Per impostazione predefinita, viene visualizzata la dashboard preconfigurata con i parametri di Approfondimenti sulle prestazioni.
5. Scegli la dashboard preconfigurata o personalizzata, quindi scegli Esporta in CloudWatch.

   Viene visualizzata la CloudWatch finestra Esporta in.
6. Scegli Aggiungi alla dashboard esistente.



7. Specifica la destinazione e l'etichetta della dashboard, quindi scegli Conferma.

   - CloudWatch destinazione del pannello di controllo: scegli un CloudWatch pannello di controllo esistente.
   - CloudWatch etichetta della sezione dashboard - opzionale - Inserisci un nome per le metriche di Performance Insights da visualizzare in questa sezione del CloudWatch dashboard.

   Un banner mostra un messaggio al completamento dell'esportazione della dashboard.

8. Scegli il link o Visualizza CloudWatch nel banner per visualizzare la dashboard delle metriche nella CloudWatch console.

Visualizzazione di un widget metrico Performance Insights in CloudWatch

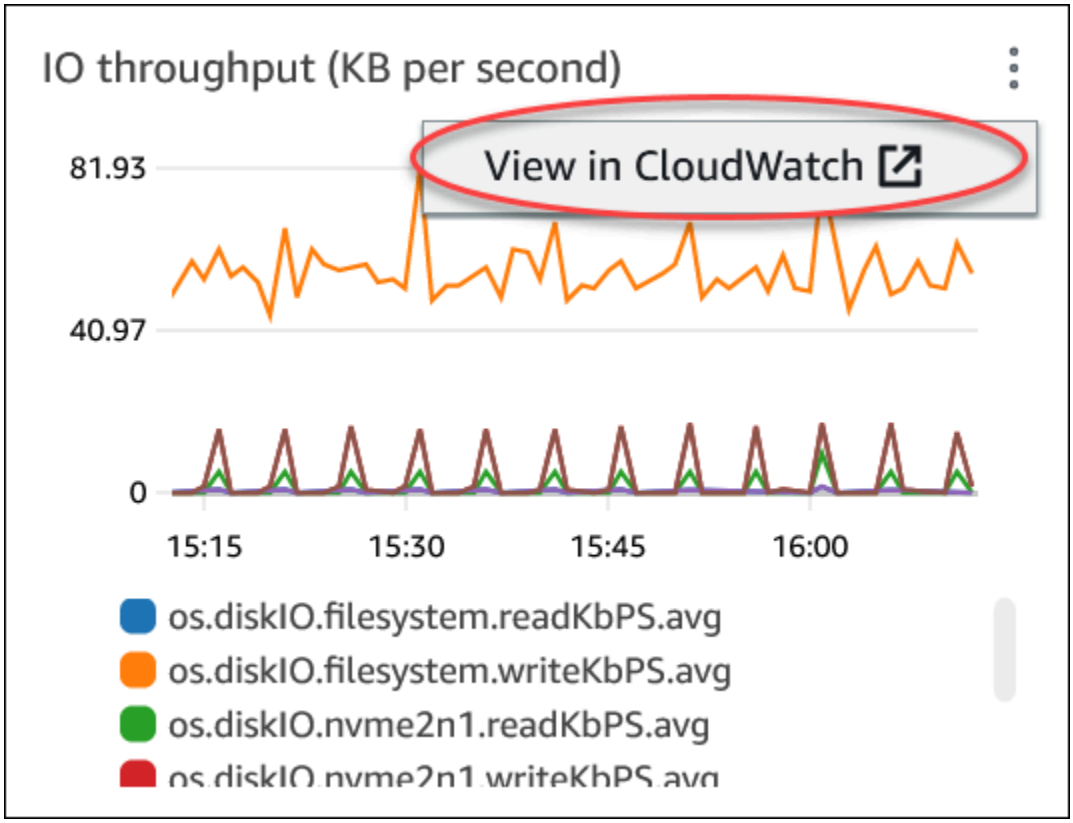
Seleziona un widget metrico Performance Insights nella dashboard di Amazon RDS Performance Insights e visualizza i dati delle metriche nella CloudWatch console.

Per esportare un widget metrico e visualizzare i dati delle metriche nella console CloudWatch

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso fino a Parametri.

   Per impostazione predefinita, viene visualizzata la dashboard preconfigurata con i parametri di Approfondimenti sulle prestazioni.
5. Scegli un widget metrico, quindi scegli Visualizza CloudWatch nel menu.



   I dati metrici vengono visualizzati nella CloudWatch console.

Creazione di CloudWatch allarmi per monitorare

Puoi creare un CloudWatch allarme che invii un SNS messaggio Amazon quando l'allarme cambia stato. Un allarme monitora un singolo parametro per un periodo di tempo specificato. L'allarme può anche eseguire una o più operazioni basate sul valore del parametro relativo a una soglia prestabilita per un certo numero di periodi. L'azione è una notifica inviata a un SNS argomento di Amazon o a una politica di Amazon EC2 Auto Scaling.

Gli allarmi richiamano azioni solo per modifiche di stato sostenute. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare. È necessario che lo stato cambi e rimanga costante per un periodo specificato

> **Note**
>
> Per Aurora, utilizza `WRITER` o utilizza i parametri del ruolo `READER` per impostare gli avvisi anziché affidarti ai parametri per istanze DB specifiche. I ruoli di istanza database Aurora possono modificare i ruoli nel tempo. Puoi trovare queste metriche basate sui ruoli nella console. CloudWatch
> L'Auto Scaling di Aurora imposta automaticamente gli avvisi in base ai parametri del ruolo `READER`. Per ulteriori informazioni sull'Auto Scaling Aurora, consultare Auto Scaling di Amazon Aurora con repliche Aurora.

Puoi utilizzare la funzione matematica DB_ PERF _ INSIGHTS metric nella CloudWatch console per interrogare i parametri dei contatori di Amazon for Performance RDS Insights. La INSIGHTS funzione DB_ PERF _ include anche la metrica a intervalli inferiori al minutoDBLoad. È possibile impostare CloudWatch allarmi su queste metriche.

Per maggiori dettagli su come creare un allarme, consulta Creare un allarme sulle metriche dei contatori di Performance Insights da un AWS database.

Per impostare un allarme utilizzando il AWS CLI

- Chiamare `put-metric-alarm`. Per ulteriori informazioni, consulta il Riferimento ai comandi AWS CLI.

Per impostare una sveglia utilizzando il CloudWatch API

- Chiama `PutMetricAlarm`. Per ulteriori informazioni, consulta Amazon CloudWatch API Reference

Per ulteriori informazioni sulla configurazione degli SNS argomenti di Amazon e sulla creazione di allarmi, consulta Using Amazon CloudWatch alarms.

Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch

Monitora il carico del database (DB Load) per la tua flotta di istanze database Amazon Aurora con Database Insights. DB Load misura il livello di attività della sessione nel database. Puoi utilizzare Database Insights per analizzare e risolvere i problemi delle prestazioni dei tuoi database Amazon su larga scala.

Con Database Insights, puoi visualizzare il carico del database sulla tua flotta di database e filtrare il carico per attese, istruzioni SQL, host o utenti.

Per impostazione predefinita, RDS abilita la modalità Standard di Database Insights per i database Amazon RDS. Quando attivi la modalità avanzata di Database Insights per un cluster DB, RDS abilita Database Insights per ogni istanza DB del cluster.

Per informazioni sull'utilizzo di Database Insights nella CloudWatch console Amazon, consulta CloudWatch Database Insights nella Amazon CloudWatch User Guide.

Prezzi

Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Argomenti

- Supporto del motore Amazon Aurora, della regione e della classe di istanza di Amazon Aurora per Database Insights
- Attivazione della modalità avanzata di Database Insights per
- Attivazione della modalità Standard di Database Insights per
-
- Considerazioni per Database Insights per

Supporto del motore Amazon Aurora, della regione e della classe di istanza di Amazon Aurora per Database Insights

La tabella seguente fornisce i motori Amazon Aurora DB che supportano Database Insights.

Motore database Amazon Aurora	Versioni motore e regioni supportate	Restrizioni delle classi di istanza
Amazon Aurora edizione compatibile con MySQL	Per ulteriori informazioni sulla disponibilità della versione e della regione di Database Insights con Aurora MySQL, consulta. Performance Insights con Aurora MySQL	Database Insights presenta le seguenti restrizioni relative alla classe di motori: • db.t2: non supportato • db.t3: non supportato • db.t4g.micro e db.t4g.small: non supportate
Amazon Aurora PostgreSQL-Compatible Edition	Per ulteriori informazioni sulla disponibilità della versione e della regione di Database Insights con Aurora PostgreSQL, consulta. Performance Insights con Aurora PostgreSQL	Non applicabile
Database Aurora PostgreSQL Limitless	Per ulteriori informazioni sull'util izzo di Database Insights con Aurora PostgreSQL Limitless Database, vedere. Monitoraggio del database Aurora PostgreSQ L Limitless con Database Insights CloudWatch	Non applicabile

Database Insights supportaAmazon Aurora Serverless v2.

Aurora Amazon supporta Database Insights nei seguenti casi. Regioni AWS

- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti orientali (Ohio)
- Stati Uniti occidentali (California settentrionale)

- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacifico (Hong Kong)
- Asia Pacific (Hyderabad)
- Asia Pacifico (Giacarta)
- Asia Pacifico (Malesia)
- Asia Pacifico (Melbourne)
- Asia Pacifico (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Europa (Francoforte)
- Europa (Irlanda)
- Europa (Londra)
- Europa (Milano)
- Europa (Parigi)
- Europa (Spagna)
- Europa (Stoccolma)
- Europa (Zurigo)
- Israele (Tel Aviv)
- Medio Oriente (Bahrein)
- Medio Oriente (Emirati Arabi Uniti)
- Sud America (San Paolo)
- AWS GovCloud (Stati Uniti orientali)

- AWS GovCloud (Stati Uniti occidentali)

Attivazione della modalità avanzata di Database Insights per

Per attivare la modalità avanzata di Database Insights per Amazon Aurora, utilizza le seguenti procedure.

Attivazione della modalità avanzata di Database Insights durante la creazione di un'

Attiva la modalità avanzata di Database Insights quando crei un database per .

Console

Nella console, puoi attivare la modalità avanzata di Database Insights quando crei un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità avanzata di Database Insights quando si crea un' utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona la modalità avanzata. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15-24 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegliere Crea database.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la creazione di un cluster DB, chiamate il create-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--database-insights-mode advanced`per attivare la modalità avanzata di Database Insights.
- `--engine`— Il motore di database per il cluster DB.
- `--db-cluster-identifier` — L'identificatore per il cluster DB .
- `--enable-performance-insights`per attivare Performance Insights for Database Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione dei dati per l'. Per attivare Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.

L'esempio seguente abilita la modalità avanzata di Database Insights durante la creazione di un cluster DB.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode advanced \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Per Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode advanced ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Per attivare la modalità avanzata di Database Insights quando crei un', specifica i seguenti parametri per l'operazione Create DBCluster Amazon RDS API.

- `DatabaseInsightsMode` Da a `advanced`
- `EnablePerformanceInsights` Da a `True`
- `PerformanceInsightsRetentionPeriod`ad almeno 465 giorni

Attivazione della modalità avanzata di Database Insights durante la modifica di un'

Attiva Database Insights quando modifichi un database per Aurora.

> **Note**
>
> Per abilitare Database Insights, ogni istanza DB in un deve avere le stesse impostazioni di Performance Insights e Enhanced Monitoring.

Console

Nella console, è possibile attivare la modalità avanzata di Database Insights quando si modifica un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità avanzata di Database Insights quando si modifica un' utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Databases (Database).
3. Scegliere un'istanza database e scegliere Modify (Modifica).
4. Nella sezione Database Insights, seleziona la modalità avanzata. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15-24 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se si sceglie Applica durante la successiva finestra di manutenzione programmata, il database ignora questa impostazione e attiva immediatamente la modalità avanzata di Database Insights.

7. Scegli Modifica del cluster.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la modifica di un cluster DB, chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--database-insights-mode advanced`per attivare la modalità avanzata di Database Insights.
- `--db-cluster-identifier` — L'identificatore per il cluster DB .
- `--enable-performance-insights`per attivare Performance Insights for Database Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione dei dati per l'. Per attivare la modalità avanzata di Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.

L'esempio seguente abilita la modalità avanzata di Database Insights quando si modifica un'istanza DB del cluster .

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Per attivare la modalità avanzata di Database Insights quando modifichi un', specifica i seguenti parametri per l'operazione Amazon RDS API.

- `DatabaseInsightsMode` Da a `advanced`
- `EnablePerformanceInsights` Da a `True`
- `PerformanceInsightsRetentionPeriod`per almeno 465 giorni

Attivazione della modalità Standard di Database Insights per

Per attivare la modalità Standard di Database Insights per Amazon Aurora, utilizza le seguenti procedure.

Attivazione della modalità Standard di Database Insights durante la creazione di un'

Attiva la modalità Standard di Database Insights quando crei un database per .

Console

Nella console, puoi attivare la modalità Standard di Database Insights quando crei un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità Standard di Database Insights quando si crea un' utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona la modalità Standard. Quindi, scegli una delle seguenti opzioni per attivare o disattivare Performance Insights:
   - Per disattivare Performance Insights, deseleziona Abilita Performance Insights.
   - Per attivare Performance Insights, seleziona Abilita Performance Insights. Per configurare Performance Insights, specificare le seguenti opzioni:
     - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di almeno 7 giorni.
     - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegliere Crea database.

AWS CLI

Per attivare la modalità Standard di Database Insights durante la creazione di un cluster DB, chiamate il create-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--database-insights-mode standard`per attivare la modalità Standard di Database Insights.
- `--engine`— Il motore di database per il cluster DB.
- `--db-cluster-identifier` — L'identificatore per il cluster DB .
- `--enable-performance-insights`so `--no-enable-performance-insights` per attivare o disattivare Performance Insights. Se si specifica`--enable-performance-insights`, è necessario specificare anche il `--performance-insights-retention-period` — Il periodo di conservazione dei dati per l'. Il periodo di conservazione deve essere di almeno 7 giorni.

L'esempio seguente abilita la modalità Standard di Database Insights e Performance Insights durante la creazione di un'.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Per Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

L'esempio seguente abilita la modalità Standard di Database Insights e disabilita Performance Insights durante la creazione di un'.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Per Windows:

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Per attivare la modalità Standard di Database Insights quando crei un', specifica i seguenti parametri per l'operazione Create DBCluster Amazon RDS API.

- `DatabaseInsightsMode` Da a `standard`
- `EnablePerformanceInsights`a `True` o. `False` Se lo `EnablePerformanceInsights` impostiTrue, devi `PerformanceInsightsRetentionPeriod` impostarlo su almeno 7 giorni.

Attivazione della modalità Standard di Database Insights durante la modifica di un'

Attiva la modalità Standard di Database Insights quando modifichi un database per Aurora.

> **Note**
>
> Per abilitare Database Insights, ogni istanza DB in un deve avere le stesse impostazioni di Performance Insights e Enhanced Monitoring.

Console

Nella console, è possibile attivare la modalità Standard di Database Insights quando si modifica un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità Standard di Database Insights quando si modifica un' utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Databases (Database).
3. Scegliere un'istanza database e scegliere Modify (Modifica).
4. Nella sezione Database Insights, seleziona la modalità Standard. Quindi, scegli tra le seguenti opzioni:
   - Per disattivare Performance Insights, deseleziona Abilita Performance Insights.
   - Per attivare Performance Insights, seleziona Abilita Performance Insights. Per configurare Performance Insights, specificare le seguenti opzioni:
     - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di almeno 7 giorni.
     - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se si sceglie Applica durante la successiva finestra di manutenzione programmata, il database ignora questa impostazione e attiva immediatamente la modalità Standard di Database Insights.
7. Scegli Modifica del cluster.

AWS CLI

Per attivare la modalità Standard di Database Insights durante la modifica di un cluster DB, chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--database-insights-mode standard`per attivare la modalità Standard di Database Insights.
- `--db-cluster-identifier` — L'identificatore per il cluster DB .

- `--enable-performance-insights`o `--no-enable-performance-insights` per attivare o disattivare Performance Insights. Se si specifica`--enable-performance-insights`, è necessario specificare anche il `--performance-insights-retention-period` — Il periodo di conservazione dei dati per l'. Il periodo di conservazione deve essere di almeno 7 giorni.

L'esempio seguente abilita la modalità Standard di Database Insights e abilita Performance Insights quando si modifica un'.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

L'esempio seguente abilita la modalità Standard di Database Insights e disabilita Performance Insights quando si modifica un'istanza DB del cluster .

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
```

```
    --no-enable-performance-insights
```

RDS API

Per attivare la modalità Standard di Database Insights quando modifichi un', specifica i seguenti parametri per l'operazione Amazon RDS API.

- `DatabaseInsightsMode` Da a `standard`
- `EnablePerformanceInsights`a `True` o. `False` Se lo `EnablePerformanceInsights` impostiTrue, devi `PerformanceInsightsRetentionPeriod` impostarlo su almeno 7 giorni.

Per monitorare le query SQL lente per il database, puoi utilizzare la sezione Query SQL lente nella dashboard di Database Insights. Prima di configurare il database per monitorare le query SQL lente, la sezione Slow SQL Queries è vuota.

Per ulteriori informazioni sul monitoraggio delle query SQL lente nella dashboard di Database Insights, consulta Visualizzazione del dashboard dell'istanza di CloudWatch database per Database Insights nella Amazon CloudWatch User Guide.

Per configurare il database per monitorare le query SQL lente con Database Insights, completa i seguenti passaggi:

1. Abilita le esportazioni dei log in CloudWatch Logs.
2. Crea o modifica il gruppo di parametri del cluster DB per il gruppo di parametri DB del cluster .

Per informazioni sulla configurazione delle esportazioni di log, consulta Pubblicazione dei log del database su Amazon Logs CloudWatch nella Amazon Aurora User Guide.

Per creare o modificare il gruppo di parametri del cluster DB, consulta i seguenti argomenti.

- Creazione di un gruppo di parametri del cluster DB in Amazon Aurora
- Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora

Amazon Aurora MySQL

Per configurare il cluster Amazon Aurora MySQL DB per monitorare le query SQL lente, imposta i seguenti parametri.

- `slow_query_log`— impostato su `1`
- `long_query_time`— impostato su `1.0`
- `log_output`— impostato su `FILE`

Amazon Aurora PostgreSQL

Per configurare il cluster Amazon Aurora PostgreSQL DB per monitorare le query SQL lente, imposta i seguenti parametri. Tieni presente che l'impostazione di questi parametri potrebbe ridurre le prestazioni del tuo cluster DB.

- `log_min_duration_statement`— impostato su `1000`
- `log_statement`— impostato su `none`
- `log_destination`— impostato su `stderr`

> **Note**
>
> Per , è possibile configurare il parametro con una granularità di 1 microsecondo. `long_query_time` Ad esempio, è possibile `0.000001` impostare questo parametro su. A seconda della quantità di query sull'istanza DB, il valore del parametro `long_query_time` può ridurre le prestazioni. Inizia con il valore `1.0` e modificalo in base al carico di lavoro. Quando imposti questo parametro su`0`, Database Insights registra tutte le query.

Per informazioni sui log di Aurora MySQL e Aurora PostgreSQL, vedere quanto segue.

- File di log del database Aurora MySQL
- File di log del database Aurora PostgreSQL

Considerazioni per Database Insights per

Di seguito sono riportate alcune considerazioni per Database Insights per Amazon Aurora.

- Non è possibile gestire Database Insights per un'istanza DB in un cluster DB .
- Per abilitare la modalità avanzata di Database Insights, è necessario abilitare Performance Insights e impostare il periodo di conservazione di Performance Insights su almeno 465 giorni (15 mesi). Oltre al costo di Database Insights, non sono previsti costi aggiuntivi per impostare il periodo

di conservazione di Performance Insights su 15 mesi. Per informazioni sui prezzi di Database Insights, consulta la pagina CloudWatch dei prezzi di Amazon.

- Per abilitare Database Insights, ogni istanza DB in un deve avere le stesse impostazioni di Performance Insights e Enhanced Monitoring.

Monitoraggio del carico DB con Performance Insights su Amazon Aurora

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Performance Insights si espande sulle caratteristiche di monitoraggio esistenti di Amazon Aurora per illustrare e aiutare ad analizzare le prestazioni del cluster. Con il pannello di controllo di Performance Insights, puoi visualizzare il carico del database sul cluster Amazon Aurora e filtrare il carico in base alle attese, alle istruzioni SQL, agli host o agli utenti. Per informazioni sull'uso di Performance Insights con Amazon DocumentDB, consulta Guida per gli sviluppatori di Amazon DocumentDB.

Argomenti

- Panoramica di Performance Insights su Amazon Aurora

- Attivazione e disattivazione di Performance Insights per Aurora
- Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL
- Configurazione delle policy di accesso per Performance Insights
- Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights
- Visualizzazione dei consigli proattivi di Performance Insights
- Recupero delle metriche con Performance Insights for Aurora API
- Registrazione delle chiamate Performance Insights utilizzando AWS CloudTrail
- Performance Insights API ed VPC endpoint di interfaccia ()AWS PrivateLink

Panoramica di Performance Insights su Amazon Aurora

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con

Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Per impostazione predefinita, RDS abilita Performance Insights nella procedura guidata di creazione della console per tutti i motori Amazon RDS. Se si attiva Performance Insights a livello di cluster DB, RDS abilita Performance Insights per ogni istanza DB del cluster.Se in un'istanza database sono presenti più database, Performance Insights aggrega i dati sulle prestazioni.

Puoi trovare una panoramica di Performance Insights per Amazon Aurora nel seguente video.

Uso di Performance Insights per analizzare le prestazioni di Amazon Aurora PostgreSQL

Argomenti

- Caricamento database
- Massimo CPU
- Supporto di classe di istanza, regione e motore di database Amazon Aurora per Performance Insights
- Prezzi e conservazione dei dati per Performance Insights

Caricamento database

Il carico del database (carico DB) misura il livello di attività della sessione nel database. `DBLoad`è la metrica chiave di Performance Insights e Performance Insights raccoglie il carico del DB ogni secondo.

Argomenti

- Sessioni attive
- Media delle sessioni attive
- Media delle esecuzioni attive
- Dimensioni

Sessioni attive

Una sessione database rappresenta il dialogo di un'applicazione con un database relazionale. Una sessione attiva è una connessione che ha inviato lavoro a un motore del database ed è in attesa di una risposta dal motore del database.

Una sessione è attiva quando è in esecuzione CPU o in attesa che una risorsa diventi disponibile per poter procedere. Ad esempio, una sessione attiva potrebbe attendere che una pagina (o un blocco) venga letta in memoria e quindi CPU consumarla mentre legge i dati dalla pagina.

Media delle sessioni attive

La media delle sessioni attive (AAS) è l'unità per la `DBLoad` metrica in Performance Insights. Misura quante sessioni sono attive contemporaneamente nel database.

Ogni secondo, Performance Insights esegue il campionamento del numero di sessioni che eseguono contemporaneamente una query. Per ogni sessione attiva, Performance Insights raccoglie i seguenti dati:

- SQLdichiarazione
- stato della sessione (in esecuzione CPU o in attesa)
- Host
- Utente che esegue il SQL

Performance Insights calcola AAS dividendo il numero totale di sessioni per il numero di campioni per un periodo di tempo specifico. Ad esempio, nella tabella seguente vengono riportati 5 campioni consecutivi di una query in esecuzione, dove ogni campione viene acquisito a intervalli di 1 secondo.

Project N.E.M.O.	Numero di sessioni che eseguono query	AAS	Calcolo
1	2	2	2 sessioni totali / 1 campione
2	0	1	2 sessioni totali / 2 campioni
3	4	2	6 sessioni totali / 3 campioni
4	0	1.5	6 sessioni totali / 4 campioni

Project N.E.M.O.	Numero di sessioni che eseguono query	AAS	Calcolo
5	4	2	10 sessioni totali / 5 campioni

Nell'esempio precedente, il carico del DB per l'intervallo di tempo era 2. AAS Questa misurazione significa che, in media, sono state attive 2 sessioni alla volta durante il periodo in cui sono stati acquisiti i 5 campioni.

Media delle esecuzioni attive

La media delle esecuzioni attive (AAE) al secondo è correlata a. AAS Per calcolareAAE, Performance Insights divide il tempo totale di esecuzione di una query per l'intervallo di tempo. La tabella seguente mostra il AAE calcolo per la stessa query nella tabella precedente.

Tempo trascorso (sec)	Tempo di esecuzione totale (sec)	AAE	Calcolo
60	120	2	120 secondi di esecuzione/60 secondi trascorsi
120	120	1	120 secondi di esecuzione/120 secondi trascorsi
180	380	2.11	380 secondi di esecuzione/180 secondi trascorsi
240	380	1.58	380 secondi di esecuzione/240 secondi trascorsi
300	600	2	600 secondi di esecuzione/300 secondi trascorsi

Nella maggior parte dei casi, gli AAS e AAE per una query sono all'incirca uguali. Tuttavia, poiché gli input per i calcoli sono origini dati diverse, i calcoli spesso variano leggermente.

Dimensioni

Il parametro `db.load` è diverso dagli altri parametri di serie temporali in quanto può essere suddiviso in sottocomponenti detti dimensioni. Le dimensioni possono essere considerate come categorie "slice by" (dividi per) per le diverse caratteristiche del parametro `DBLoad`.

Quando si diagnosticano problemi di prestazioni, le dimensioni seguenti sono spesso le più utili:

Argomenti

- Eventi di attesa
- In alto SQL

Per un elenco completo delle dimensioni dei motori Aurora, consulta. Carico del database suddiviso per dimensioni

Eventi di attesa

Un evento di attesa fa sì che un'SQListruzione attenda il verificarsi di un evento specifico prima di poter continuare a funzionare. Gli eventi di attesa sono una dimensione o una categoria importante per il caricamento del database perché indicano dove il lavoro è impedito.

Ogni sessione attiva è in esecuzione CPU o in attesa. Ad esempio, le sessioni consumano CPU quando cercano un buffer nella memoria, eseguono un calcolo o eseguono codice procedurale. Quando le sessioni non si consumanoCPU, potrebbero attendere che un buffer di memoria si liberi, che venga letto un file di dati o che venga scritto un registro. Più tempo impiega una sessione per attendere le risorse, minore è il tempo di esecuzione su. CPU

Quando si sintonizza un database, si tenta spesso di scoprire le risorse che le sessioni sono in attesa. Ad esempio, due o tre eventi di attesa potrebbero rappresentare il 90% del carico DB. Questa misura significa che, in media, le sessioni attive trascorrono la maggior parte del tempo in attesa di un numero limitato di risorse. Se riesci a scoprire la causa di queste attese, puoi provare a fornire una soluzione.

Gli eventi di attesa variano in base al motore database:

- Per un elenco degli eventi di attesa comuni per Aurora MySQL, vedi. Eventi Aurora My wait SQL Per informazioni su come sintonizzarsi utilizzando questi eventi di attesa, consulta Ottimizzazione di Aurora MySQL.
- Per informazioni su tutti gli eventi e SQL My wait, vedi Tabelle di riepilogo degli eventi Wait nella documentazione personale. SQL
- Per un elenco degli eventi di attesa comuni per Aurora SQL Postgre, vedi. Eventi di attesa Amazon Aurora PostgreSQL Per informazioni su come sintonizzarsi utilizzando questi eventi di attesa, consulta Sintonizzazione degli eventi di attesa per Aurora PostgreSQL.
- Per informazioni su tutti gli eventi di SQL attesa di Postgre, consulta The Statistics Collector > Wait Event tables nella documentazione di Postgre. SQL

In alto SQL

Laddove gli eventi di attesa mostrano problemi, la parte superiore SQL mostra quali sono le query che contribuiscono maggiormente al carico del DB. Ad esempio, molte query potrebbero essere attualmente in esecuzione nel database, ma una singola query potrebbe consumare il 99 percento del carico DB. In questo caso, il carico elevato potrebbe indicare un problema con la query.

Per impostazione predefinita, la console Performance Insights visualizza le principali SQL query che contribuiscono al carico del database. La console mostra anche le statistiche pertinenti per ogni istruzione. Per diagnosticare problemi di prestazioni per un'istruzione specifica, è possibile esaminarne il piano di esecuzione.

Massimo CPU

Nel dashboard, il grafico di caricamento del database raccoglie, aggrega e visualizza le informazioni sulla sessione. Per vedere se le sessioni attive superano il limite massimoCPU, guarda la loro relazione con la CPU linea Max v. Performance Insights determina il CPU valore Max v in base al numero di v core CPU (virtualiCPU) per l'istanza DB. Per Aurora Serverless v2, Max v CPU rappresenta il numero stimato di. vCPUs

Un processo può essere eseguito su una v CPU alla volta. Se il numero di processi supera il numero divCPUs, i processi iniziano a mettersi in coda. Quando l'accodamento aumenta, le prestazioni del database diminuiscono. Se il carico del DB è spesso superiore alla CPU linea Max v e lo stato di attesa principale èCPU, viene sovraccaricato. CPU In questo caso, potresti voler limitare le connessioni all'istanza, ottimizzare le SQL query con un CPU carico elevato o prendere in considerazione una classe di istanza più ampia. Istanze elevate e costanti di qualsiasi stato di attesa

indicano che possono verificarsi colli di bottiglia o problemi di conflitto delle risorse da risolvere. Questo può essere vero anche se il carico del DB non supera la linea Max v. CPU

Supporto di classe di istanza, regione e motore di database Amazon Aurora per Performance Insights

Important

AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Nella tabella seguente vengono forniti i motori di database Amazon Aurora che supportano Performance Insights.

Motore database Amazon Aurora	Versioni motore e regioni supportate	Restrizioni delle classi di istanza
Amazon Aurora edizione compatibile con MySQL	Per ulteriori informazioni sulla disponibilità di versioni e Regioni di Approfondimenti sulle prestazioni con Aurora MySQL, consulta Performance Insights con Aurora MySQL.	Performance Insights ha le seguenti restrizioni della classe di motori: • db.t2: non supportato • db.t3: non supportato • db.t4g.micro e db.t4g.small: non supportate
Amazon Aurora PostgreSQL-Compatible Edition	Per ulteriori informazioni sulla disponibilità di versioni e Regioni di Approfondimenti sulle prestazioni con Aurora PostgreSQL, consulta Performance Insights con Aurora PostgreSQL.	N/D

Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Nella tabella seguente vengono forniti i motori di database Amazon Aurora che supportano funzionalità Performance Insights.

Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
Statistiche SQL per Performance Insights	Tutti	Tutti	Tutti	Tutti
Analisi delle prestazioni del	Solo livello a pagamento	Tutti	Tutti	

Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
database per un periodo di tempo				Tutti tranne db.serverless (Aurora Serverless v2)

Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
Visualizzazione dei consigli proattivi di Performance Insights	Solo livello a pagamento	• Stati Uniti orientali (Ohio) • Stati Uniti orientali (Virginia settentrionale) • Stati Uniti occidentali (California settentrionale) • Stati Uniti occidentali (Oregon) • Asia Pacifico (Mumbai) • Asia Pacifico (Seoul) • Asia Pacifico (Singapore) • Asia Pacifico (Sydney) • Asia Pacifico (Tokyo) • Canada (Centrale) • Europa (Francoforte) • Europa (Irlanda)	Tutti	Tutti tranne db.serverless (Aurora Serverless v2)

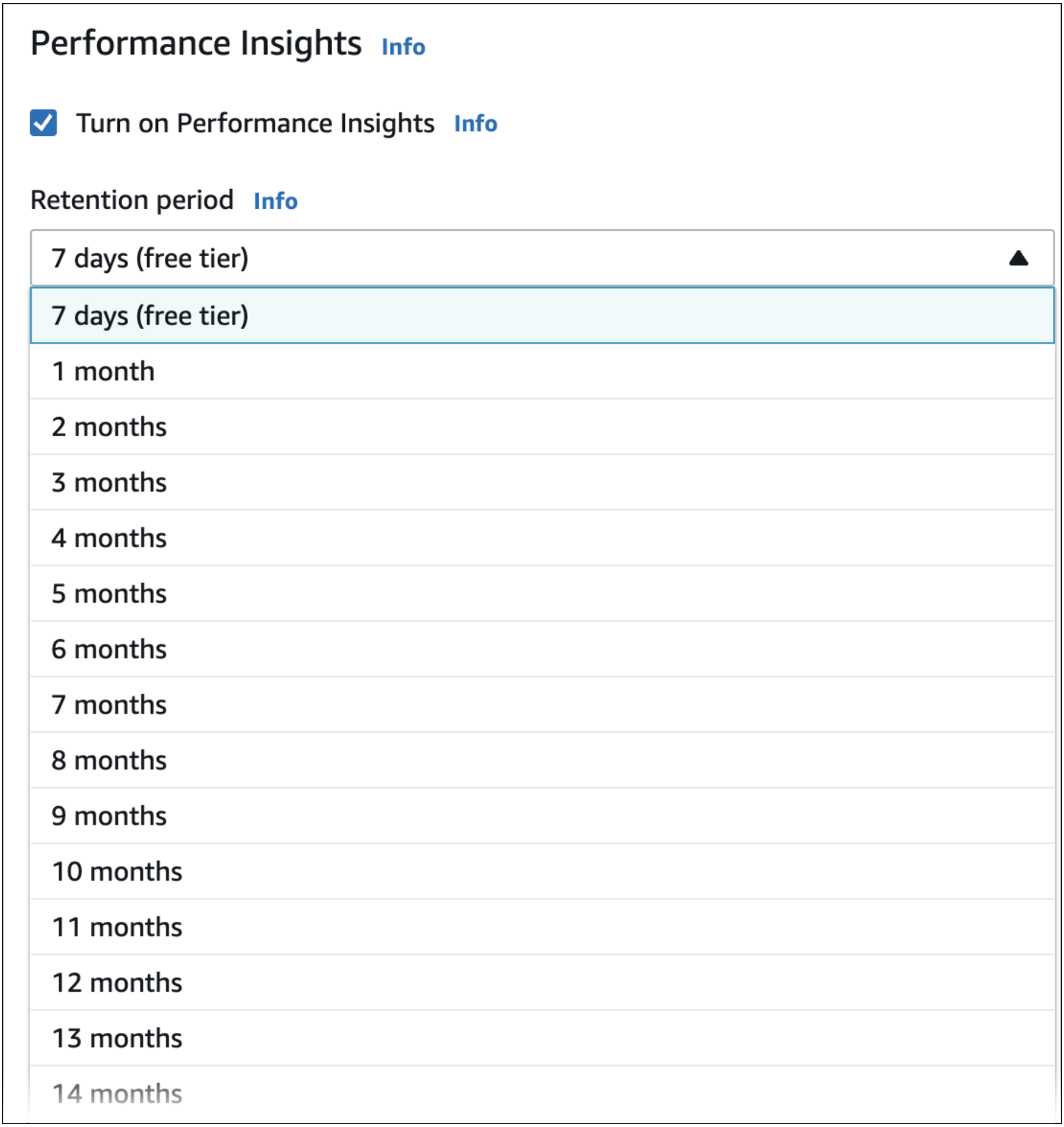
Funzionalità	Livello di prezzi	Regioni supportate	Motori di database supportati	Classi di istanza supportate
		• Europe (London) • Europe (Paris) • Europa (Stoccolma) • Sud America (San Paolo)		

Prezzi e conservazione dei dati per Performance Insights

Per impostazione predefinita, Performance Insights offre un piano gratuito che include 7 giorni di cronologia dei dati sulle prestazioni e 1 milione di API richieste al mese. Puoi anche acquistare periodi di conservazione più lunghi. Per informazioni sui prezzi, consulta Prezzi di Performance Insights.

Nella RDS console, puoi scegliere uno dei seguenti periodi di conservazione per i tuoi dati di Performance Insights:

- Default (7 giorni)
- ***n***mesi, dove ***n***è un numero compreso tra 1 e 24



Per informazioni su come impostare un periodo di conservazione utilizzando il AWS CLI, consulta. Attivazione e disattivazione di Performance Insights per Aurora

> **Note**
>
> L'arresto di un' con Performance Insights abilitato non influisce sulla conservazione dei dati. Mentre un' viene interrotto, Performance Insights non raccoglie alcun dato.

Attivazione e disattivazione di Performance Insights per Aurora

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
> Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
> Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
> Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Puoi attivare Performance Insights per l' al momento della creazione. Se necessario, puoi disattivarlo in un secondo momento modificando l' dalla console. L'attivazione e la disattivazione di Performance Insights non determina tempi di inattività, riavvio o failover.

> **Note**
>
> Performance Schema è uno strumento di prestazioni opzionale utilizzato da Aurora MySQL. Se si attiva o disattiva Performance Schema, è necessario riavviare il sistema. Se si attiva o disattiva Performance Insights, tuttavia, non è necessario riavviare. Per ulteriori informazioni, consulta Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

Se utilizzi Performance Insights con i database globali Aurora, attiva Performance Insights singolarmente per i database di ciascuno di essi. Regione AWS Per informazioni dettagliate, consultare Monitoraggio di un database globale Amazon Aurora con Amazon Performance Insights RDS.

L'agente Performance Insights consuma CPU e memoria limitate sull'host DB. Quando il carico del DB è elevato, l'agente limita l'impatto sulle prestazioni raccogliendo i dati meno frequentemente.

Console

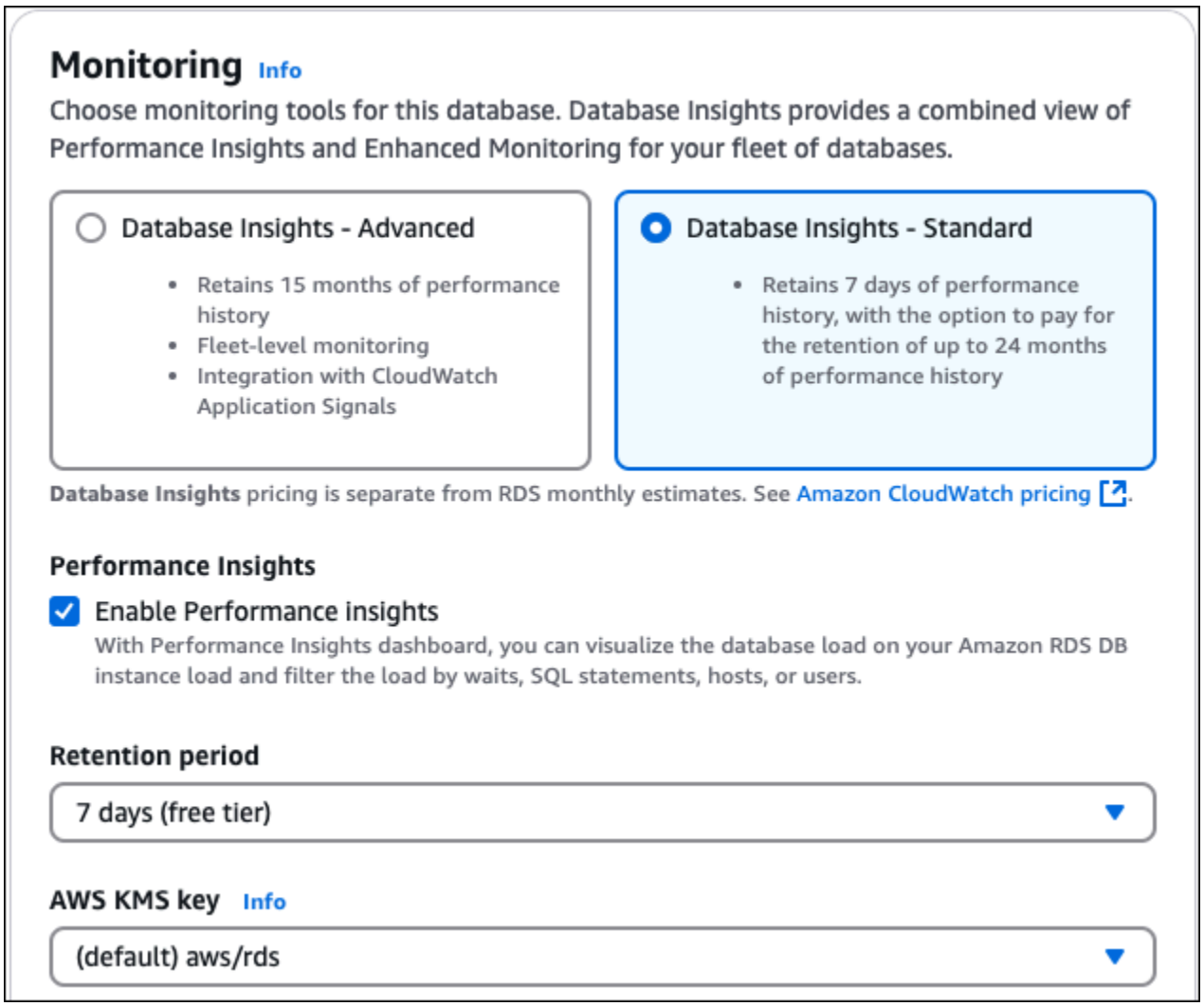
Nella console è possibile attivare o disattivare la funzionalità Approfondimenti sulle prestazioni quando crei o modifichi un cluster database. L'attivazione di Performance Insights consente di gestire le impostazioni e le opzioni di Performance Insights per il cluster DB. Le impostazioni a livello di cluster si applicano a tutte le istanze DB del cluster.

Attivazione o disattivazione di Performance Insights durante la creazione di un'

Dopo aver creato una nuova , Amazon RDS abilita Performance Insights per impostazione predefinita. Per disattivare Performance Insights per il tuo cluster DB, scegli l'opzione Database Insights — Standard e deseleziona l'opzione Abilita Performance Insights.

Per creare un cluster DB, segui le istruzioni relative al tuo motore DB inCreazione di un cluster database Amazon Aurora.

L'immagine seguente mostra la sezione Performance Insights.



Quando selezioni Abilita Performance Insights, sono disponibili le opzioni seguenti:

- Conservazione (solo per la modalità Standard di Database Insights): la quantità di tempo per conservare i dati di Performance Insights. L'impostazione del periodo di conservazione nel livello gratuito è Default (7 days) (Impostazione predefinita [7 giorni]). Per mantenere i dati sulle prestazioni più a lungo, specifica da 1 a 24 mesi. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.
- AWS KMS key— Specificate il vostro AWS KMS key. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Modifica di una AWS KMS politica per Performance Insights.

Attivazione o disattivazione di Performance Insights quando si modifica un'

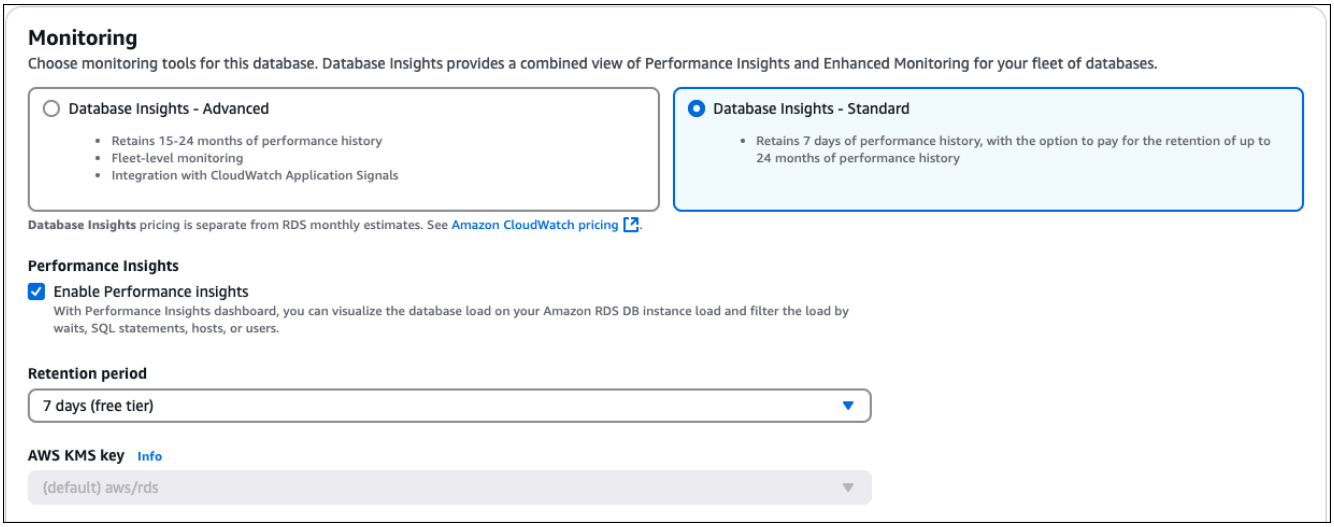
Nella console, è possibile modificare un cluster DB, un'istanza DB o un cluster per gestire Performance Insights.

Per gestire Performance Insights per un' utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli un' e scegli Modifica.
4. Per attivare Performance Insights, seleziona Abilita Performance Insights. Per disattivare Performance Insights per il tuo cluster DB, scegli l'opzione Database Insights — Standard e deseleziona l'opzione Abilita Performance Insights.

   Quando selezioni Abilita Performance Insights, sono disponibili le opzioni seguenti:

   - Conservazione (solo per la modalità Standard di Database Insights): la quantità di tempo per conservare i dati di Performance Insights. L'impostazione del periodo di conservazione nel livello gratuito è Default (7 days) (Impostazione predefinita [7 giorni]). Per mantenere i dati sulle prestazioni più a lungo, specifica da 1 a 24 mesi. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.



5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegli Apply immediately (Applica immediatamente). Se scegli Apply (Applica) durante la prossima finestra di manutenzione pianificata, l'istanza ignora questa impostazione e attiva immediatamente Performance Insights.
7. Scegli Modify instance (Modifica istanza).

AWS CLI

Quando usi il create-db-instance AWS CLI comando, attiva Performance Insights specificando `--enable-performance-insights` e impostando `--database-insights-mode` su `advanced` o`standard`. Per disattivare Performance Insights, specificare `--no-enable-performance-insights` e `database-insights-mode` impostare su`standard`.

È inoltre possibile specificare questi valori utilizzando i seguenti AWS CLI comandi:

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-replica
- modify-db-instance
- restore-db-instance-from-s3

Per gestire Performance Insights per un cluster DB utilizzando AWS CLI

- Chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:
  - `--db-cluster-identifier` - Il nome dell'istanza database nel cluster di database.
  - `--enable-performance-insights` per attivare o `--no-enable-performance-insights` per disattivare
  - `database-insights-mode`— La modalità di Database Insights per l'. Per disattivare Performance Insights, imposta questo valore su`standard`.

  Il seguente esempio attiva Performance Insights per `sample-db-cluster`.

  Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights
```

Per Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights
```

Quando attivi Performance Insights nella CLI, puoi specificare, in via facoltativa, il periodo di tempo, in giorni, per cui mantenere i dati di Performance Insights con l'opzione `--performance-insights-retention-period`. È possibile specificare7, *`month`* * 31 (dove *`month`* è un numero compreso tra 1 e 23) o. 731 Ad esempio, se desideri mantenere i dati sulle prestazioni per 3 mesi, specifica 93, che è 3 * 31. L'impostazione di default è 7 giorni. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.

Il seguente esempio attiva Performance Insights per `sample-db-cluster` e specifica che i dati Performance Insights sono mantenuti per 93 giorni (3 mesi).

PerLinux, omacOS: Unix

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Per Windows:

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
    --performance-insights-retention-period 93
```

Se il periodo di conservazione specificato, ad esempio 94 giorni, non è un valore valido, RDS genera un errore.

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> È possibile attivare Performance Insights solo per un'istanza in un cluster DB in cui Performance Insights non è gestito a livello di cluster.

RDS API

Quando crei una nuova istanza DB nel tuo cluster DB utilizzando l'DBInstanceoperazione Create operation Amazon RDS API, attiva Performance Insights `EnablePerformanceInsights` impostando su. `True` Per disattivare Performance Insights, imposta `EnablePerformanceInsights` `False` e imposta `DatabaseInsightsMode` `standard`.

Puoi anche specificare il valore `EnablePerformanceInsights` utilizzando le seguenti operazioni API:

- CreaDBCluster
- ModificaDBCluster
- ModificaDBInstance
- CreaDBInstanceReadReplica
- Ripristina DBInstance da S3

Quando si attiva Performance Insights, è possibile specificare, in via facoltativa, il periodo di tempo, in giorni, per cui conservare i dati Performance Insights con il parametro `PerformanceInsightsRetentionPeriod`. È possibile specificare7, *`month`* * 31 (dove *`month`* è un numero compreso tra 1 e 23) o. `731` Ad esempio, se desideri mantenere i dati sulle prestazioni per 3 mesi, specifica 93, che è 3 * 31. L'impostazione di default è 7 giorni. Per altre informazioni sui periodi di conservazione, consulta Prezzi e conservazione dei dati per Performance Insights.

Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL

Performance Schema è una funzionalità facoltativa per il monitoraggio delle prestazioni di runtime di Aurora MySQL a un dettaglio di basso livello. Performance Schema è progettato per avere un impatto minimo sulle prestazioni del database. Performance Insights è una funzionalità separata che puoi utilizzare con o senza Performance Schema.

Argomenti

- Panoramica dello schema di prestazioni
- Performance Insights e lo schema di prestazioni
- Gestione automatica di Performance Schema da parte di Performance Insights
- Effetto di un riavvio su Performance Schema
- Determinazione della gestione di Performance Schema da parte di Performance Insights
- Attiva lo schema delle prestazioni per Aurora, MySQL, Amazon RDS MariaDB o MySQL.

Panoramica dello schema di prestazioni

Performance Schema monitora gli eventi nei database Aurora MySQL. Un evento è un'azione del server di database che consuma tempo ed è stata strumentata in modo che possano essere raccolte le informazioni di temporizzazione. Ecco alcuni esempi di eventi:

- Chiamate di funzione
- Attendi il sistema operativo
- Fasi dell'esecuzione SQL
- Gruppi di istruzioni SQL

Il motore di archiviazione `PERFORMANCE_SCHEMA` è un meccanismo per l'implementazione della funzionalità Performance Schema. Questo motore raccoglie i dati degli eventi utilizzando la strumentazione nel codice sorgente del database. Il motore memorizza gli eventi raccolti nelle tabelle in memoria nel database `performance_schema`. È possibile interrogare `performance_schema` proprio come puoi interrogare qualsiasi altra tabella. Per ulteriori informazioni, consulta Performance Schema di MySQL nel Manuale di riferimento di MySQL.

Performance Insights e lo schema di prestazioni

Performance Insights e Performance Schema sono funzionalità separate, ma sono connesse. Il comportamento di Performance Insights per Aurora MySQL varia a seconda che lo schema di prestazioni sia attivato e, in questo caso, se Performance Insights gestisce automaticamente lo schema di prestazioni. Il comportamento viene descritto nella tabella seguente.

Schema di prestazioni attivato	Modalità di gestione di Performance Insights	Comportamento di Performance Insights
Sì	Automatica	• Raccoglie informazioni dettagliate di monitoraggio a basso livello • Raccoglie le metriche di sessione attive ogni secondo • Visualizza il carico del database classificato in base a eventi di attesa dettagliati, che è possibile utilizzare per identificare i colli di bottiglia
Sì	Manuale	• Raccoglie gli eventi di attesa e le metriche per SQL • Raccoglie le metriche di sessione attive ogni secondo • Segnala gli stati utente, ad esempio l'inserimento e l'invio, che non consentono di identificare i colli di bottiglia
No	N/D	• Non raccoglie eventi di attesa, metriche per SQL o altre informazioni dettagliate di monitoraggio di basso livello • Raccoglie le metriche di sessione attive ogni cinque secondi anziché ogni secondo •

Schema di prestazioni attivato	Modalità di gestione di Performance Insights	Comportamento di Performance Insights
		Segnala gli stati utente, ad esempio l'inserimento e l'invio, che non consentono di identificare i colli di bottiglia

Gestione automatica di Performance Schema da parte di Performance Insights

Quando crei un'istanza database Aurora MySQL con Performance Insights abilitato, anche la funzionalità Performance Schema viene abilitata. In questo caso, Performance Insights gestisce automaticamente i parametri di Performance Schema. Questa è la configurazione consigliata.

Quando Performance Insights gestisce automaticamente lo schema delle prestazioni, il Source of `performance_schema` è`System default`.

> **Note**
>
> La gestione automatica dello schema di prestazioni non è supportata per la classe di istanza t4g.medium.

Puoi anche gestire Performance Schema manualmente. Se scegli questa opzione, imposta i parametri in base ai valori nella tabella riportata di seguito.

Nome del parametro	Valore del parametro
`performance_schema`	1 (la colonna Source (Fonte) è impostata al valore `Modified`)
`performance-schema-consumer-events-waits-current`	`ON`
`performance-schema-instrument`	`wait/%=ON`

Nome del parametro	Valore del parametro
`performance_schema_consumer_global_instrumentation`	1
`performance_schema_consumer_thread_instrumentation`	1

Se modifichi il manualmente il parametro `performance_schema` e in seguito desideri ripristinare la gestione automatica, consulta Attiva lo schema delle prestazioni per Aurora, MySQL, Amazon RDS MariaDB o MySQL..

> **Important**
>
> Quando Performance Insights abilita Performance Schema, non modifica i valori del gruppo di parametri. Tuttavia, i valori vengono modificati sulle istanze database in esecuzione. L'unico modo per vedere i valori modificati è eseguire il comando `SHOW GLOBAL VARIABLES`.

Effetto di un riavvio su Performance Schema

Performance Insights e Performance Schema differiscono per i requisiti relativi al riavvio delle istanze DB:

Performance Schema

Per attivare o disattivare questa funzionalità, è necessario riavviare l'istanza database.

Approfondimenti sulle prestazioni

Per attivare o disattivare questa funzionalità, non è necessario riavviare l'istanza database.

Se Performance Schema non è attualmente attivato e si attiva Performance Insights senza riavviare l'istanza database, Performance Schema non verrà attivato.

Determinazione della gestione di Performance Schema da parte di Performance Insights

Per scoprire se Performance Insights gestisce attualmente lo schema delle prestazioni per tutte le principali versioni del motore supportate, consulta la tabella seguente.

Impostazione del parametro performance_schema	Impostazione della colonna Source	Performance Insights sta gestendo Performance Schema?
`0`	`System default`	Sì
`0` o `1`	`Modified`	No

Nella procedura seguente, si determina se Performance Insights gestisce automaticamente lo schema delle prestazioni.

Per determinare se Performance Insights sta gestendo automaticamente Performance Schema

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Gruppi di parametri.
3. Selezionare il nome del gruppo di parametri per l'istanza database.
4. Inserire **`performance_schema`** nella barra di ricerca.
5. Verifica se Source è l'impostazione predefinita del sistema e se Value è 0. In tal caso, Performance Insights gestisce automaticamente Performance Schema.

   Nell'esempio mostrato qui, Performance Insights non gestisce automaticamente lo schema delle prestazioni.

Attiva lo schema delle prestazioni per Aurora, MySQL, Amazon RDS MariaDB o MySQL.

Supponiamo che Performance Insights sia attivato per l'istanza database ma al momento non stia gestendo Performance Schema. Se desideri consentire a Performance Insights di gestire automaticamente Performance Schema, completa la procedura seguente.

Configurazione di Performance Schema per la gestione automatica

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Gruppi di parametri.
3. Selezionare il nome del gruppo di parametri per l'istanza database.
4. Scegli Modifica.
5. Inserire **`performance_schema`** nella barra di ricerca.
6. Selezionare il parametro `performance_schema`.
7. Scegli Imposta sul valore predefinito.
8. Conferma scegliendo Imposta valori predefiniti.
9. Seleziona Salva modifiche.
10. Riavviare l'istanza database.

> **Important**
>
> Ogni volta che si abilita o disabilita Performance Schema, è necessario riavviare l'istanza database.

Per ulteriori informazioni sulla modifica dei parametri di un'istanza, consulta Modifica dei parametri in un gruppo di parametri DB in . Per ulteriori informazioni sulle pagine del pannello di controllo, consulta Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights. Per ulteriori informazioni sullo schema delle prestazioni di MySQL, vedere MySQL Performance Schema (per 8.0) e MySQL Performance Schema (per 8.4) nella documentazione MySQL.

Configurazione delle policy di accesso per Performance Insights

Per accedere a Performance Insights, un principale deve disporre delle autorizzazioni appropriate di AWS Identity and Access Management (IAM). Puoi garantire l'accesso secondo le seguenti modalità:

- Collega la policy gestita `AmazonRDSPerformanceInsightsReadOnly` a un set di autorizzazioni o a un ruolo per accedere a tutte le operazioni di sola lettura dell'API di Performance Insights. Allega le seguenti CloudWatch autorizzazioni:`GetMetricStatistics`,, e. `ListMetrics` `GetMetricData` Per ulteriori informazioni sulle CloudWatch autorizzazioni, consulta Amazon CloudWatch permissions reference.
- Collega la policy gestita `AmazonRDSPerformanceInsightsFullAccess` a un set di autorizzazioni o a un ruolo per accedere a tutte le operazioni dell'API di Performance Insights. Allega le seguenti CloudWatch autorizzazioni:,, e. `GetMetricStatistics` `ListMetrics` `GetMetricData` Per ulteriori informazioni sulle CloudWatch autorizzazioni, consulta Amazon CloudWatch permissions reference.
- Crea una policy IAM personalizzata e collegala a un set di autorizzazioni o un ruolo.

Se hai specificato una chiave gestita dal cliente quando hai attivato Performance Insights, assicurati che gli utenti del tuo account dispongano delle `kms:GenerateDataKey` autorizzazioni `kms:Decrypt` e su. AWS KMS key

Nelle sezioni seguenti, collega una policy AWS gestita a un principale IAM, crea una policy IAM personalizzata, modifica una policy e concedi un AWS KMS accesso granulare a Performance Insights.

Argomenti

- Allegare il AmazonRDSPerformanceInsightsReadOnly politica a un preside IAM
- Allegare il AmazonRDSPerformanceInsightsFullAccess politica a un preside IAM
- Creazione di una IAM policy personalizzata per Performance Insights
- Modifica di una AWS KMS politica per Performance Insights
- Concessione di un accesso granulare a Performance Insights

Allegare il AmazonRDSPerformanceInsightsReadOnly politica a un preside IAM

`AmazonRDSPerformanceInsightsReadOnly`è una policy AWS gestita che garantisce l'accesso a tutte le operazioni di sola lettura dell'API Amazon RDS Performance Insights.

Se si collega `AmazonRDSPerformanceInsightsReadOnly` a un set di autorizzazioni o un ruolo, il destinatario può utilizzare Performance Insights insieme ad altre funzionalità della console.

Per ulteriori informazioni, consulta AWS politica gestita: A mazonRDSPerformance InsightsReadOnly.

Allegare il AmazonRDSPerformanceInsightsFullAccess politica a un preside IAM

`AmazonRDSPerformanceInsightsFullAccess`è una policy AWS gestita che garantisce l'accesso a tutte le operazioni dell'API Amazon RDS Performance Insights.

Se si collega `AmazonRDSPerformanceInsightsFullAccess` a un set di autorizzazioni o un ruolo, il destinatario può utilizzare Performance Insights insieme ad altre funzionalità della console.

Per ulteriori informazioni, consulta AWS politica gestita: A mazonRDSPerformance InsightsFullAccess.

Creazione di una IAM policy personalizzata per Performance Insights

Agli utenti che non dispongono della `AmazonRDSPerformanceInsightsFullAccess` policy `AmazonRDSPerformanceInsightsReadOnly` o, puoi concedere l'accesso a Performance Insights creando o modificando una policy gestita dall'utenteIAM. Quando alleghi la policy a un set di IAM autorizzazioni o a un ruolo, il destinatario può utilizzare Performance Insights.

Per creare una policy personalizzata

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella pagina Crea policy, scegli l'JSONopzione.
5. Copia e incolla il testo fornito nella sezione del documento relativo alla JSON policy della AWS Managed Policy Reference Guide per AmazonRDSPerformanceInsightsReadOnly o AmazonRDSPerformanceInsightsFullAccesspolitica.
6. Scegliere Review policy (Esamina policy).
7. Specifica un nome per la policy e, facoltativamente, una descrizione e quindi scegli Create policy (Crea policy).

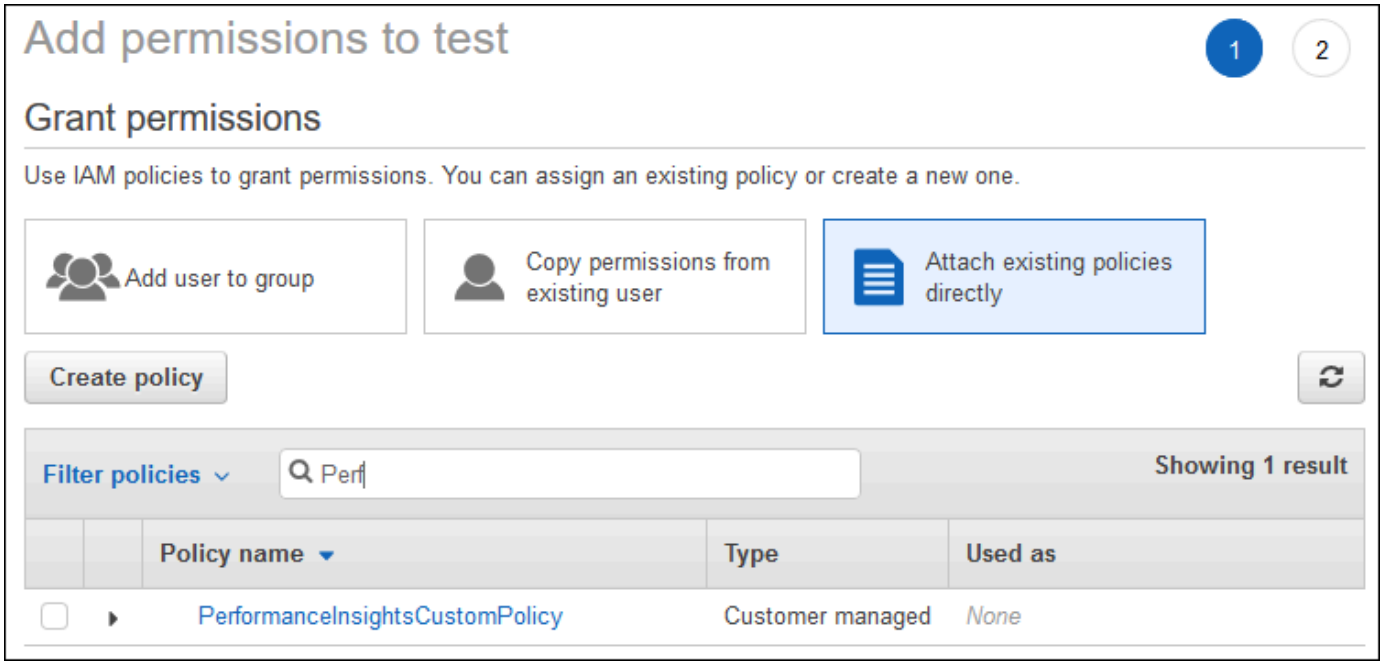
Ora è possibile collegare la policy a un set di autorizzazioni o un ruolo. La seguente procedura presuppone che si disponga già di un utente disponibile allo scopo.

Per collegare la policy a un utente

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione, seleziona Users (Utenti).
3. Seleziona un utente esistente dall'elenco.

> **Important**
>
> Per utilizzare Performance Insights, assicurati di avere accesso RDS ad Amazon oltre alla politica personalizzata. Ad esempio, la policy `AmazonRDSPerformanceInsightsReadOnly` predefinita fornisce l'accesso in sola lettura ad Amazon. RDS Per ulteriori informazioni, consulta Gestione dell'accesso con policy.

4. Nella pagina Summary (Riepilogo), scegli Add permissions (Aggiungi autorizzazioni).
5. Scegli Attach existing policies directly (Collega direttamente le policy esistenti). Per Search, digita i primi caratteri del nome della tua polizza, come mostrato nell'immagine seguente.



6. Scegli la policy e quindi seleziona Next: Review (Successivo: Rivedi).
7. Scegli Add Permissions (Aggiungi autorizzazioni).

Modifica di una AWS KMS politica per Performance Insights

Performance Insights utilizza an AWS KMS key per crittografare i dati sensibili. Quando abiliti Performance Insights tramite API o la console, puoi eseguire una delle seguenti operazioni:

- Scegli l'impostazione predefinita Chiave gestita da AWS.

  Amazon RDS utilizza il Chiave gestita da AWS per la tua nuova istanza DB. Amazon ne RDS crea uno Chiave gestita da AWS per te Account AWS. Il tuo Account AWS ne ha uno diverso Chiave gestita da AWS per Amazon RDS per ognuno Regione AWS.
- Scegli una chiave gestita dal cliente.

  Se specifichi una chiave gestita dal cliente, gli utenti del tuo account che chiamano Performance Insights API necessitano delle `kms:GenerateDataKey` autorizzazioni `kms:Decrypt` e sulla KMS chiave. È possibile configurare queste autorizzazioni tramite IAM policy. Tuttavia, ti consigliamo di gestire queste autorizzazioni tramite la tua politica KMS chiave. Per ulteriori informazioni, consulta Policy delle chiavi in AWS KMS nella Guida per gli sviluppatori di AWS Key Management Service .

Example

L'esempio seguente mostra come aggiungere dichiarazioni alla politica KMS chiave. Queste istruzioni consentono l'accesso a Performance Insights. A seconda di come usi la KMS chiave, potresti voler modificare alcune restrizioni. Prima di aggiungere istruzioni alle policy, ai criteri, rimuovi tutti i commenti.

```
{
"Version" : "2012-10-17",
 "Id" : "your-policy",
 "Statement" : [ {
    //This represents a statement that currently exists in your policy.
 }
 ....,
 //Starting here, add new statement to your policy for Performance Insights.
 //We recommend that you add one new statement for every RDS instance
{
    "Sid" : "Allow viewing RDS Performance Insights",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            //One or more principals allowed to access Performance Insights
```

```
            "arn:aws:iam::444455556666:role/Role1"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition" : {
        "StringEquals" : {
            //Restrict access to only RDS APIs (including Performance Insights).
            //Replace region with your AWS Region.
            //For example, specify us-west-2.
            "kms:ViaService" : "rds.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
            //Restrict access to only data encrypted by Performance Insights.
            "kms:EncryptionContext:aws:pi:service": "rds",
            "kms:EncryptionContext:service": "pi",

            //Restrict access to a specific RDS instance.
            //The value is a DbiResourceId.
           "kms:EncryptionContext:aws:rds:db-id": "db-AAAAABBBBBCCCCDDDDDEEEEE"
        }
    }
}
```

In che modo Performance Insights utilizza la chiave gestita dal AWS KMS cliente

La funzionalità Approfondimenti sulle prestazioni utilizza una chiave gestita dal cliente per crittografare i dati sensibili. Quando attivi Performance Insights, puoi fornire una AWS KMS chiave tramiteAPI. Performance Insights crea KMS autorizzazioni su questa chiave. Utilizza la chiave ed esegue le operazioni necessarie per elaborare i dati sensibili. I dati sensibili includono campi come utente, database, applicazione e testo della SQL query. La funzionalità Approfondimenti sulle prestazioni garantisce che i dati rimangano crittografati mentre sono sia in transito che inattivi..

Come IAM funziona Performance Insights con AWS KMS

IAMconcede autorizzazioni a persone specificheAPIs. Performance Insights ha il seguente pubblicoAPIs, che puoi limitare utilizzando IAM le policy:

- `DescribeDimensionKeys`

- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

È possibile utilizzare le seguenti API richieste per ottenere dati sensibili.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetrics`

Quando si utilizza il API per ottenere dati sensibili, Performance Insights sfrutta le credenziali del chiamante. Questo controllo garantisce che l'accesso ai dati sensibili sia limitato a coloro che hanno accesso alla chiave. KMS

Quando li chiamiAPIs, sono necessarie le autorizzazioni per richiamarli API tramite la IAM policy e le autorizzazioni per richiamare l'`kms:decrypt`azione tramite la AWS KMS policy chiave.

`GetResourceMetrics`APIPossono restituire dati sensibili e non sensibili. I parametri della richiesta determinano se la risposta deve includere dati sensibili. APIRestituisce dati sensibili quando la richiesta include una dimensione sensibile nel filtro o nei parametri di raggruppamento.

Per ulteriori informazioni sulle dimensioni che è possibile utilizzare con `GetResourceMetrics`API, vedere. DimensionGroup

Example Esempi

L'esempio seguente richiede i dati sensibili per il gruppo `db.user`:

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
```

```
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
        "Limit": 2
      }
    }
  ],
  "StartTime": 1693872000,
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

L'esempio seguente richiede i dati non sensibili per la metrica `db.load.avg`:

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
```

```
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

Concessione di un accesso granulare a Performance Insights

Il controllo granulare degli accessi offre metodi aggiuntivi per controllare l'accesso a Performance Insights. Questo controllo di accesso può consentire o negare l'accesso alle singole dimensioni e alle azioni di `GetResourceMetrics` `GetDimensionKeyDetails` Performance Insights. `DescribeDimensionKeys` Per utilizzare un accesso granulare, specifica le dimensioni nella IAM policy utilizzando le chiavi di condizione. La valutazione dell'accesso segue la logica di valutazione delle IAM politiche. Per ulteriori informazioni, vedere Logica di valutazione delle politiche nella Guida IAM per l'utente. Se l'IAMinformativa non specifica alcuna dimensione, controlla l'accesso a tutte le dimensioni per l'azione specificata. Per l'elenco delle dimensioni disponibili, vedere DimensionGroup.

Per scoprire a quali dimensioni le tue credenziali sono autorizzate ad accedere, utilizza il `AuthorizedActions` parametro in `ListAvailableResourceDimensions` e specifica l'azione. I valori consentiti per `AuthorizedActions` sono i seguenti:

- `GetResourceMetrics`
- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`

Ad esempio, se si specifica `GetResourceMetrics` il `AuthorizedActions` parametro, `ListAvailableResourceDimensions` restituisce l'elenco delle dimensioni a cui l'`GetResourceMetrics`azione è autorizzata ad accedere. Se specificate più azioni nel `AuthorizedActions` parametro, `ListAvailableResourceDimensions` restituisce un'intersezione di dimensioni a cui tali azioni sono autorizzate ad accedere.

Example

L'esempio seguente fornisce l'accesso alle dimensioni `GetResourceMetrics` e alle `DescribeDimensionKeys` azioni specificate.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    // only these dimensions are allowed. Dimensions not included in
                    // a policy with "Allow" effect will be denied
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

Di seguito è riportata la risposta per la dimensione richiesta:

```
 // ListAvailableResourceDimensions API
```

```
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

L'esempio seguente specifica un accesso consentito e due negato l'accesso per le dimensioni.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },
```

```
          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
```

```
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }
    ]
}
```

Di seguito sono riportate le risposte per le dimensioni richieste:

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },
```

```
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // allowed for DescribeDimensionKeys because our IAM Policy
                  // denies it only for GetResourceMetrics
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
```

```
        ] }
    ]
}
```

Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights

> **Important**
>
> AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili (1-24 mesi) e i relativi prezzi. L'API Performance Insights continuerà a esistere senza modifiche ai prezzi. I costi per l'API Performance Insights verranno visualizzati nella AWS fattura insieme al costo di CloudWatch Database Insights.
> Si consiglia di aggiornare tutte DB Clusters di database utilizzando il livello a pagamento di Performance Insights alla modalità Advanced di Database Insights prima del 30 novembre 2025. Per informazioni sull'aggiornamento alla modalità avanzata di Database Insights, consulta. Attivazione della modalità avanzata di Database Insights per
> Se non si esegue alcuna azione, le istanze DB raggruppano utilizzando Performance Insights per impostazione predefinita utilizzando la modalità Standard di Database Insights. Con la modalità Standard di Database Insights, potresti perdere l'accesso alla cronologia dei dati sulle prestazioni per oltre 7 giorni e potresti non essere in grado di utilizzare i piani di esecuzione e le funzionalità di analisi su richiesta nella console Amazon RDS. Dopo il 30 novembre 2025, solo la modalità Advanced di Database Insights supporterà i piani di esecuzione e l'analisi su richiesta.
> Con CloudWatch Database Insights, puoi monitorare il carico del database per la tua flotta di database e analizzare e risolvere i problemi delle prestazioni su larga scala. Per ulteriori informazioni su Database Insights, consulta. Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch Per informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

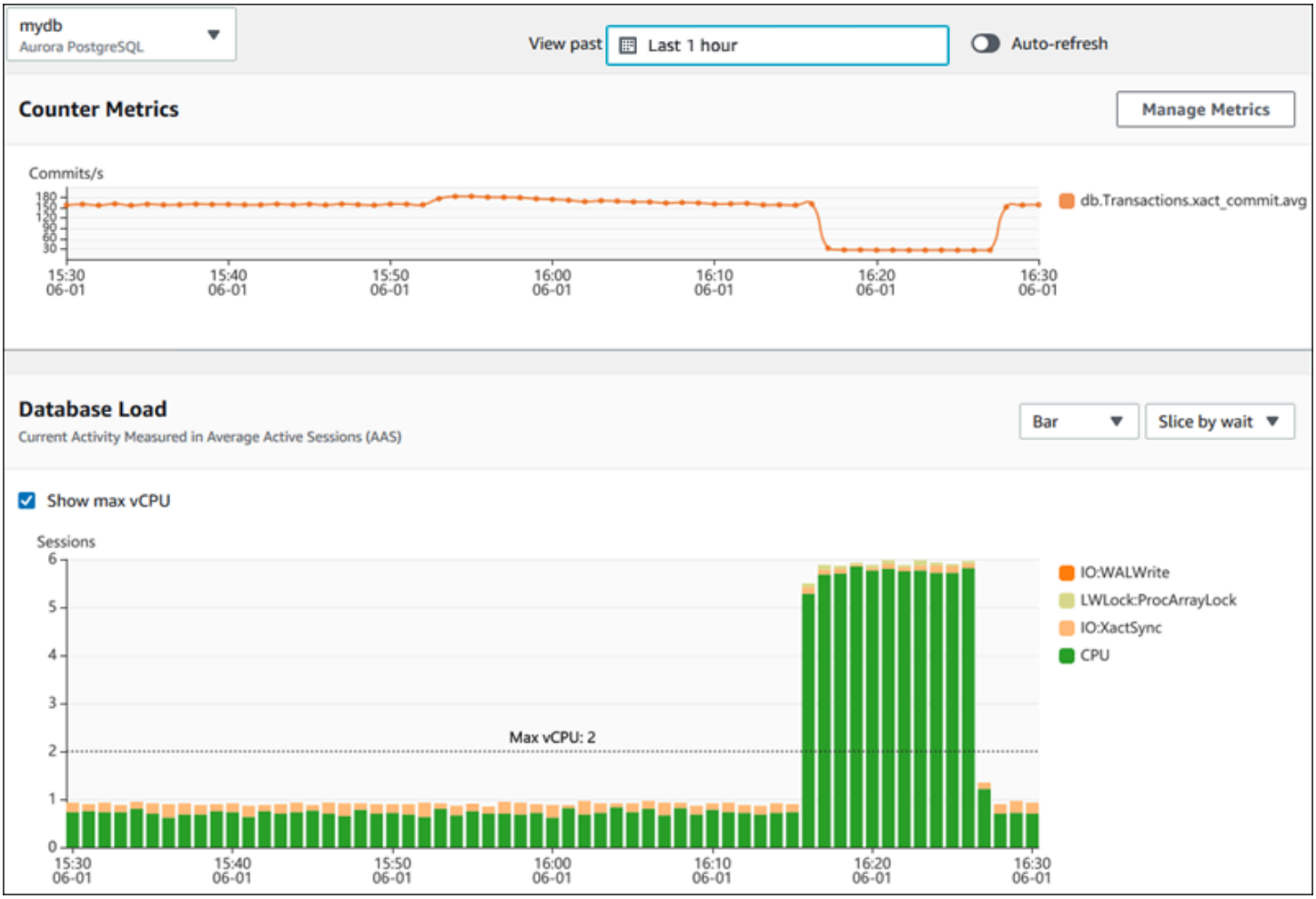
Il pannello di controllo di Performance Insights contiene informazioni sulle performance del database, per consentire di analizzare e risolvere i problemi di performance. Nella pagina principale del pannello di controllo è possibile visualizzare le informazioni relative al carico del database. Puoi "dividere" il carico del database per dimensioni come eventi di attesa o SQL.

Pannello di controllo di Performance Insights

- Panoramica del pannello di controllo di Performance Insights
- Accesso al pannello di controllo di Performance Insights
- Analisi del carico del database per eventi di attesa
- Analisi delle prestazioni del database per un periodo di tempo
- Analisi delle query con la SQL scheda Top di Performance Insights

Panoramica del pannello di controllo di Performance Insights

Il pannello di controllo è il modo più semplice per interagire con Performance Insights. L'esempio seguente mostra la dashboard per un'istanza SQL DB Postgre.



Argomenti

- Filtro intervallo temporale

- Grafico Parametri contatore
- Grafico di carico database
- Tabella dimensioni superiori

Filtro intervallo temporale

Di default, il pannello di controllo di Performance Insights mostra il carico del database relativo all'ultima ora. Puoi regolare questo intervallo di tempo da 5 minuti o fino a 2 anni. Puoi inoltre selezionare un intervallo relativo personalizzato.

Puoi selezionare un intervallo assoluto con data e ora di inizio e fine. L'esempio seguente mostra l'intervallo di tempo che inizia alla mezzanotte del 25/09/24 e termina alle 23:59 del 28/09/24.

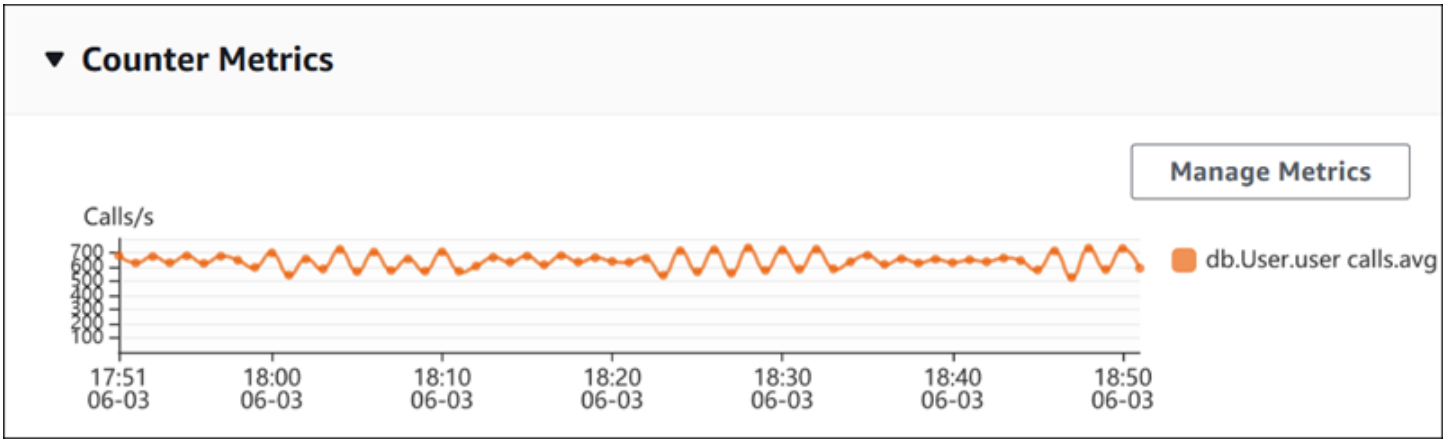
Per impostazione predefinita, il fuso orario per il pannello di controllo Performance Insights è Coordinated Universal Time (UTC). Puoi anche scegliere il fuso orario locale.

Grafico Parametri contatore

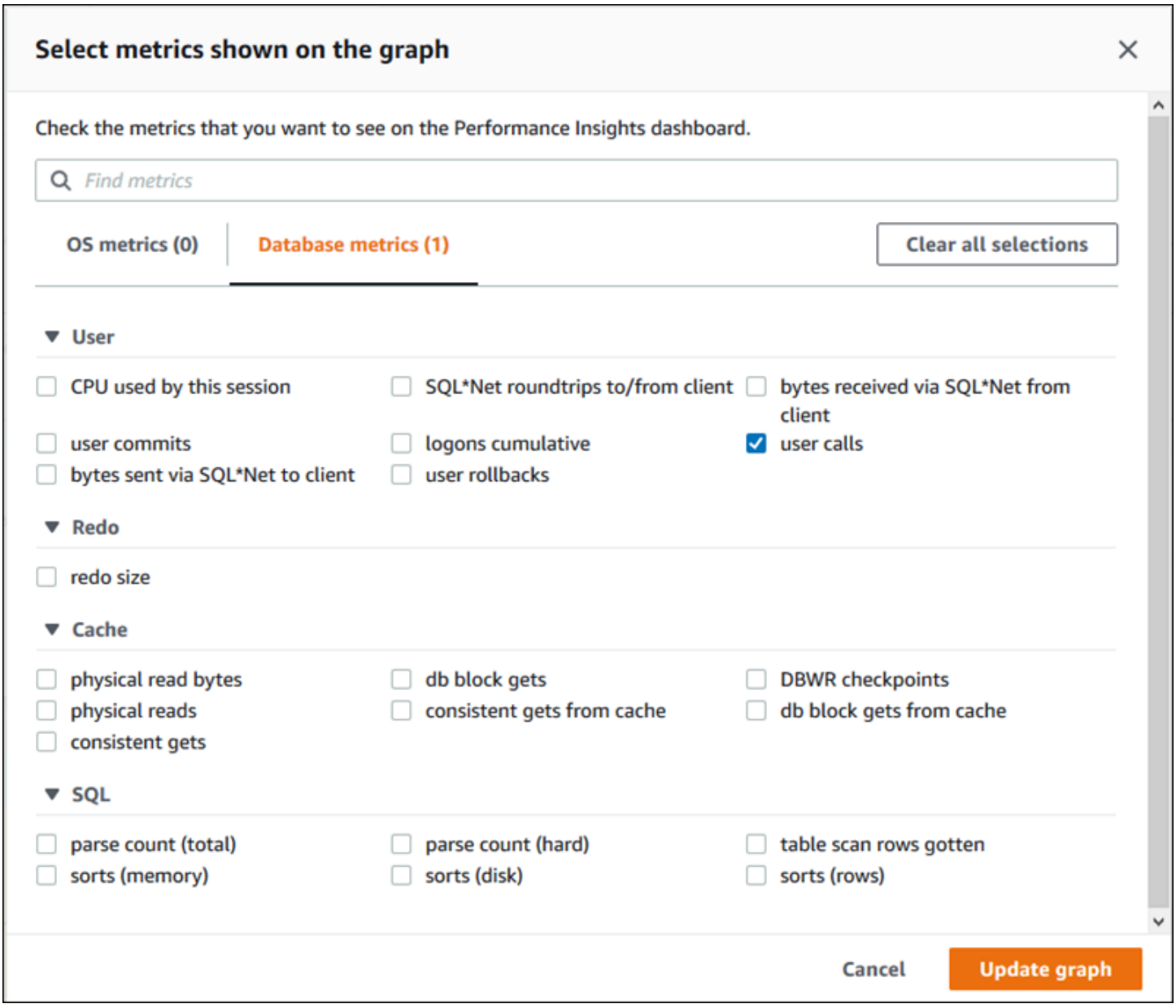
Con i parametri contatore, puoi personalizzare il pannello di controllo di Performance Insights per includere fino a 10 grafici aggiuntivi. Questi grafici mostrano una selezione di decine di parametri prestazionali di sistema operativo e database. Queste informazioni possono essere correlate ai carichi dei database per agevolare l'individuazione e l'analisi di problemi legati alle prestazioni.

Il grafico Counter Metrics (Parametri contatore) visualizza i dati per i contatori delle prestazioni. I parametri predefiniti dipendono dal motore DB:

- Aurora Mia — SQL `db.SQL.Innodb_rows_read.avg`
- Aurora Postgre — SQL `db.Transactions.xact_commit.avg`



Per modificare i contatori delle prestazioni, scegli Manage Metrics (Gestisci parametri). È possibile selezionare più parametri del sistema operativo o metriche del database, come mostrato nello screenshot seguente. Per visualizzare i dettagli relativi a qualsiasi metrica, passare il mouse sul nome della metrica.

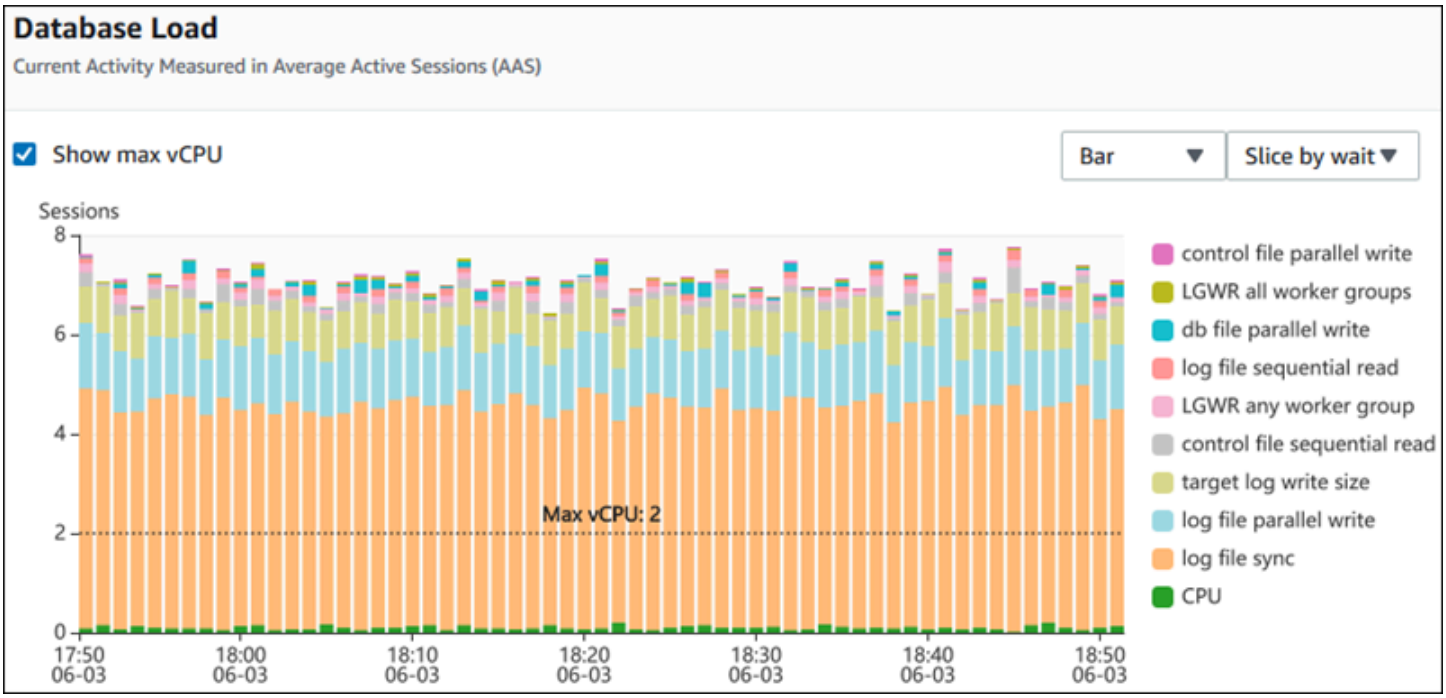


Per le descrizioni dei parametri contatore che è possibile aggiungere per ciascun motore database, consultare Parametri contatore di Performance Insights.

Grafico di carico database

Il grafico di carico del database mostra il confronto tra l'attività del database e la capacità dell'istanza DB, rappresentata dalla riga Max v. CPU Per impostazione predefinita, il grafico a linee in pila

rappresenta il carico DB come sessioni attive medie per unità di tempo. Il carico DB viene suddiviso (raggruppato) in base agli stati di attesa.

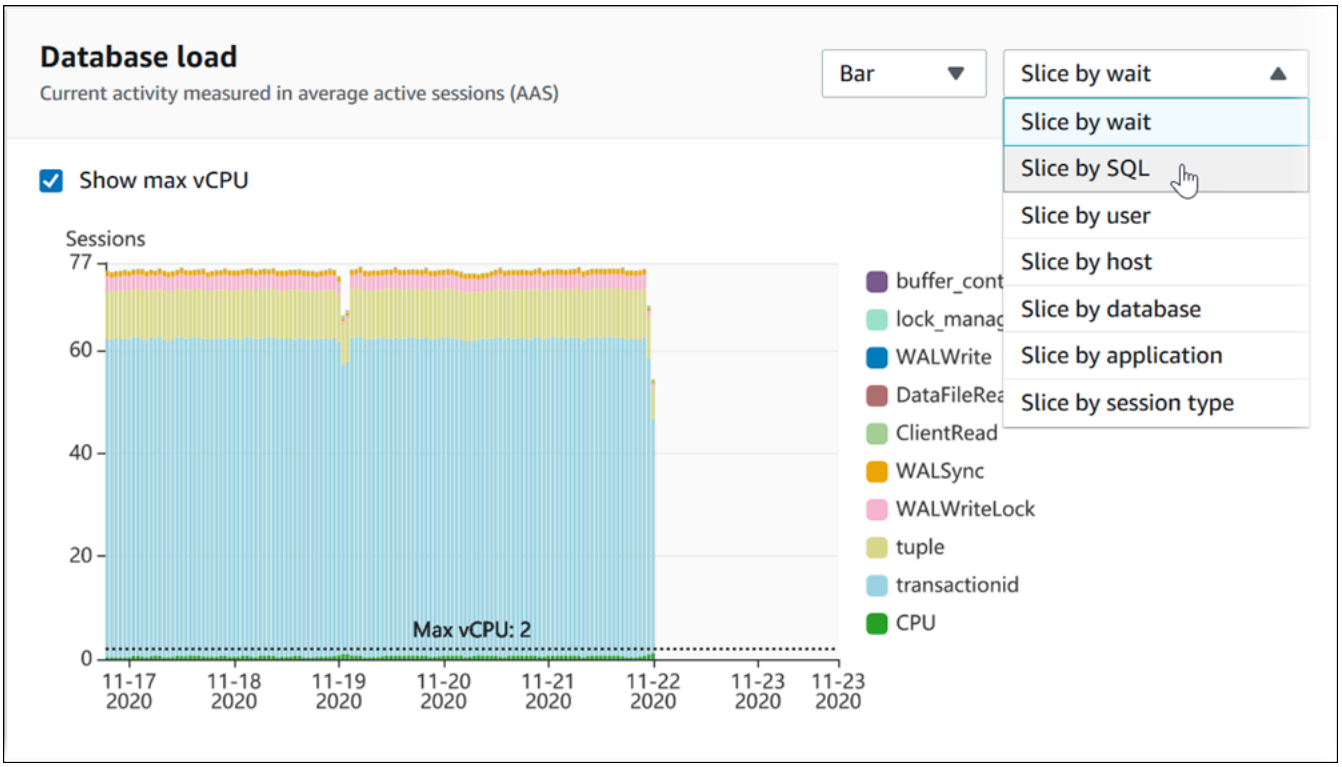


Carico del database suddiviso per dimensioni

È possibile scegliere di visualizzare il carico sotto forma di sessioni attive raggruppate in base alle dimensioni supportate. La tabella seguente mostra le dimensioni supportate per i diversi motori.

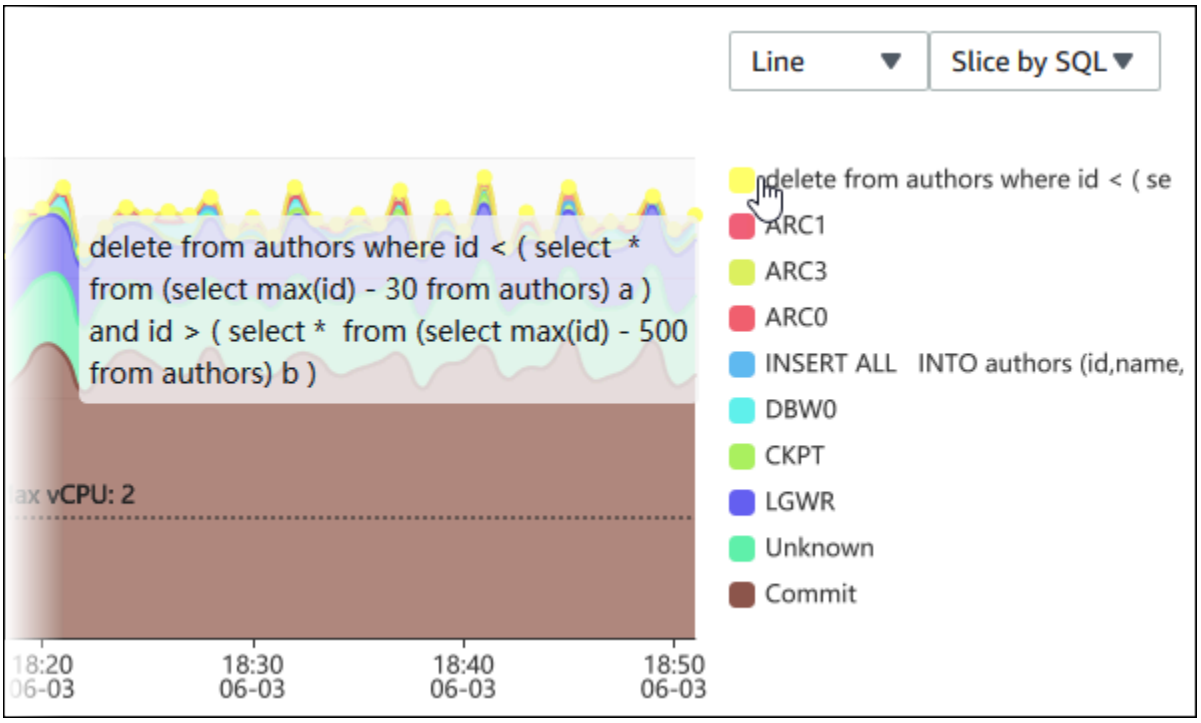
Dimensione	Aurora Postger SQL	Aurora Mia SQL
Host	Sì	Sì
SQL	Sì	Sì
Utente	Sì	Sì
Stati di attesa	Sì	Sì
Applicazione	Sì	No
Database	Sì	Sì
Tipo di sessione	Sì	No

L'immagine seguente mostra le dimensioni di un'istanza DB Postgre. SQL



Dettagli del carico DB per un elemento della dimensione

Per visualizzare i dettagli su un elemento del carico del database all'interno di una dimensione, passa il mouse sul nome dell'elemento. L'immagine seguente mostra i dettagli di una SQL dichiarazione.



Per visualizzare i dettagli relativi a qualsiasi elemento per il periodo di tempo selezionato nella legenda, passa il mouse su tale elemento.

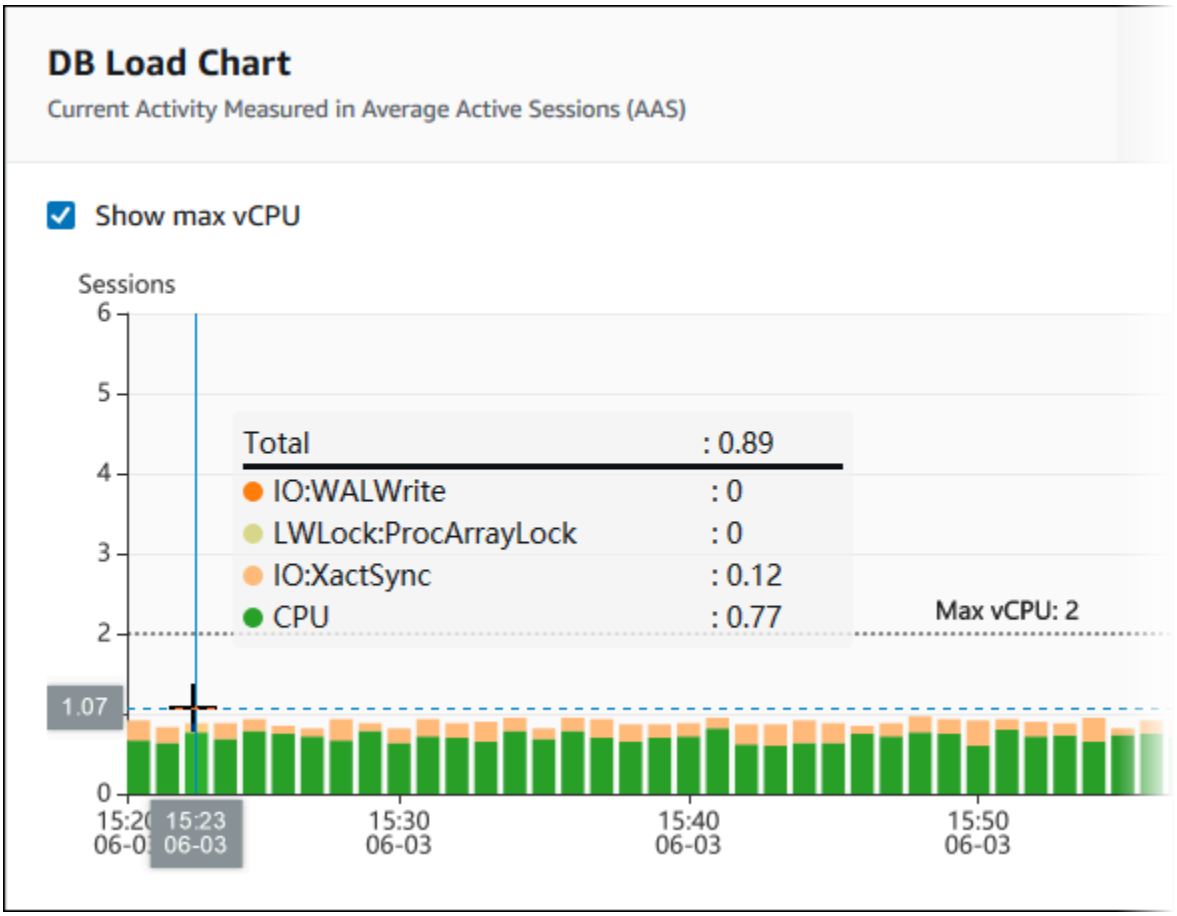
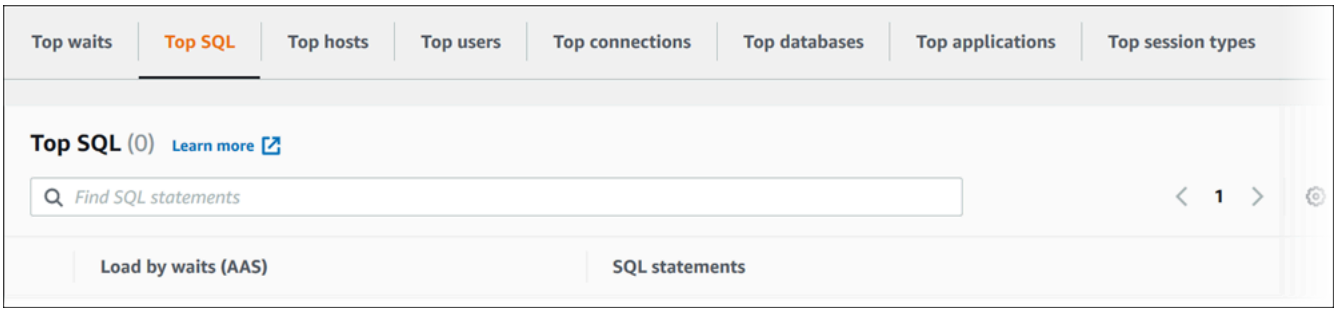


Tabella dimensioni superiori

La tabella delle dimensioni superiori seziona il carico DB in diverse dimensioni. Una dimensione è una categoria o una suddivisione per le diverse caratteristiche del carico del database. Se la dimensione èSQL, Top SQL mostra le SQL istruzioni che contribuiscono maggiormente al carico del DB.



Scegli una delle seguenti schede di dimensione.

Scheda	Descrizione	Motori supportati
In alto SQL	Le SQL dichiarazioni attualmente in esecuzione	Tutti
Principali stati d'attesa	L'evento per il quale il back-end del database è in attesa	Tutti
Host principali	Il nome host del client connesso	Tutti
Utenti principali	L'utente collegato al database	Tutti
Applicazioni principali	Il nome dell'applicazione connessa al database	Solo Aurora SQL
Tipi di sessione principali	Il tipo di sessione corrente	Solo Aurora Postgre SQL

Per informazioni su come analizzare le interrogazioni utilizzando la scheda Superiore SQL, consulta. Panoramica della scheda Top SQL

Accesso al pannello di controllo di Performance Insights

Amazon RDS fornisce una visualizzazione consolidata di Performance Insights e delle CloudWatch metriche nella dashboard di Performance Insights.

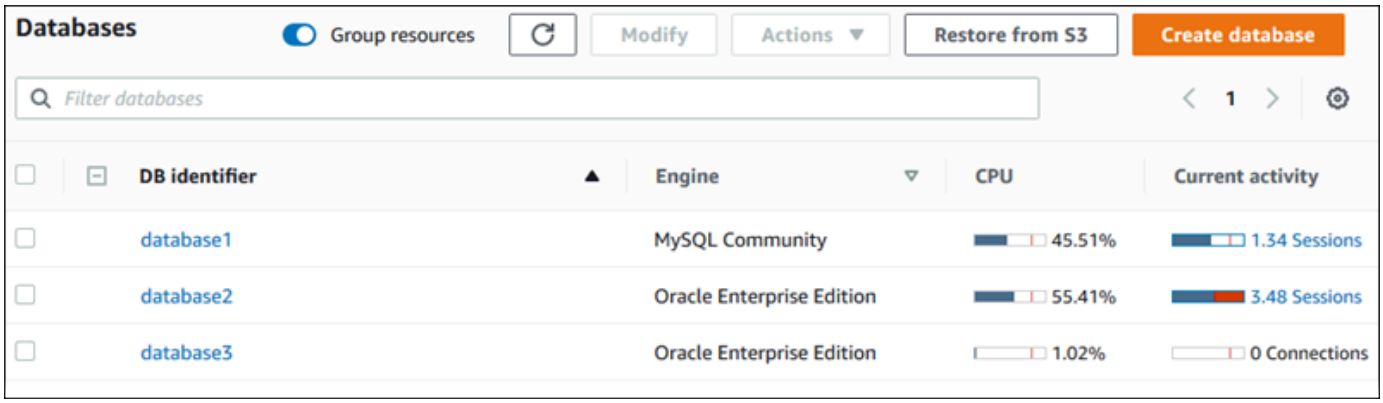
Per visualizzare il pannello di controllo di Performance Insights, procedi come indicato di seguito.

Per visualizzare il dashboard di Performance Insights nella console AWS di gestione

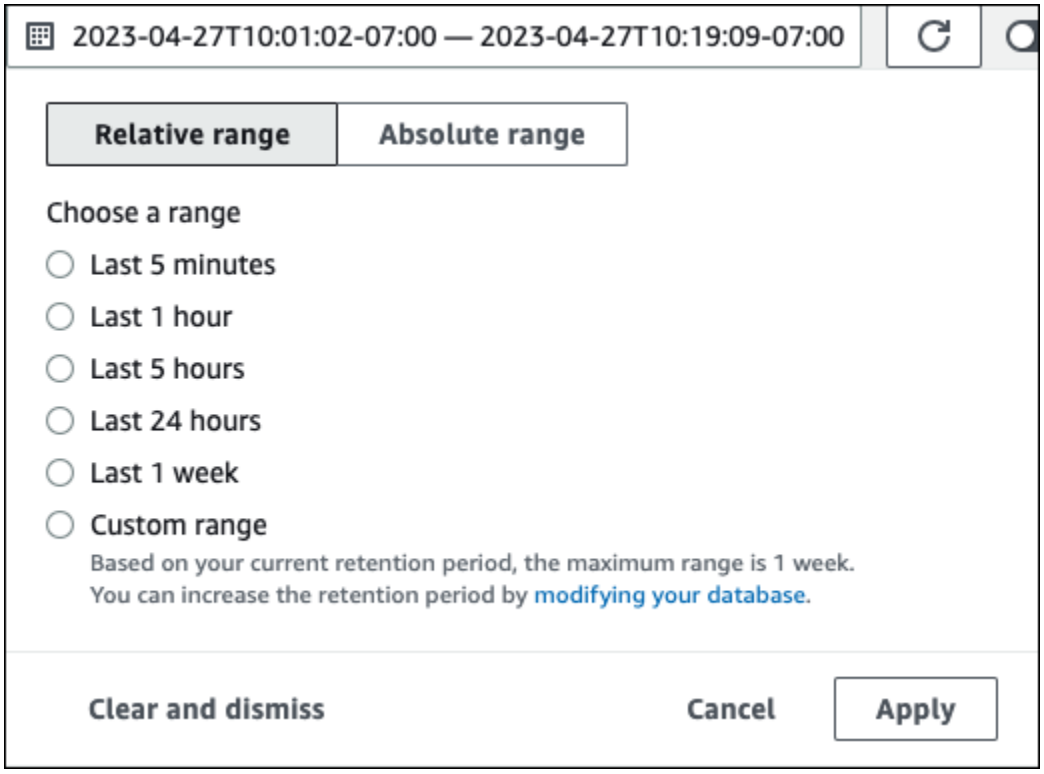
1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Per le istanze DB con Performance Insights attivata, puoi anche accedere alla dashboard di Performance Insights scegliendo la voce Sessioni nell'elenco delle istanze DB. In Current activity (Attività corrente) la voce Sessions (Sessioni) mostra il carico del database in sessioni attive medie negli ultimi cinque minuti. Il grafico mostra graficamente il carico: Quando la barra è vuota, l'istanza database è inattiva. Con l'aumentare del carico, la barra si riempie ed è di colore blu.

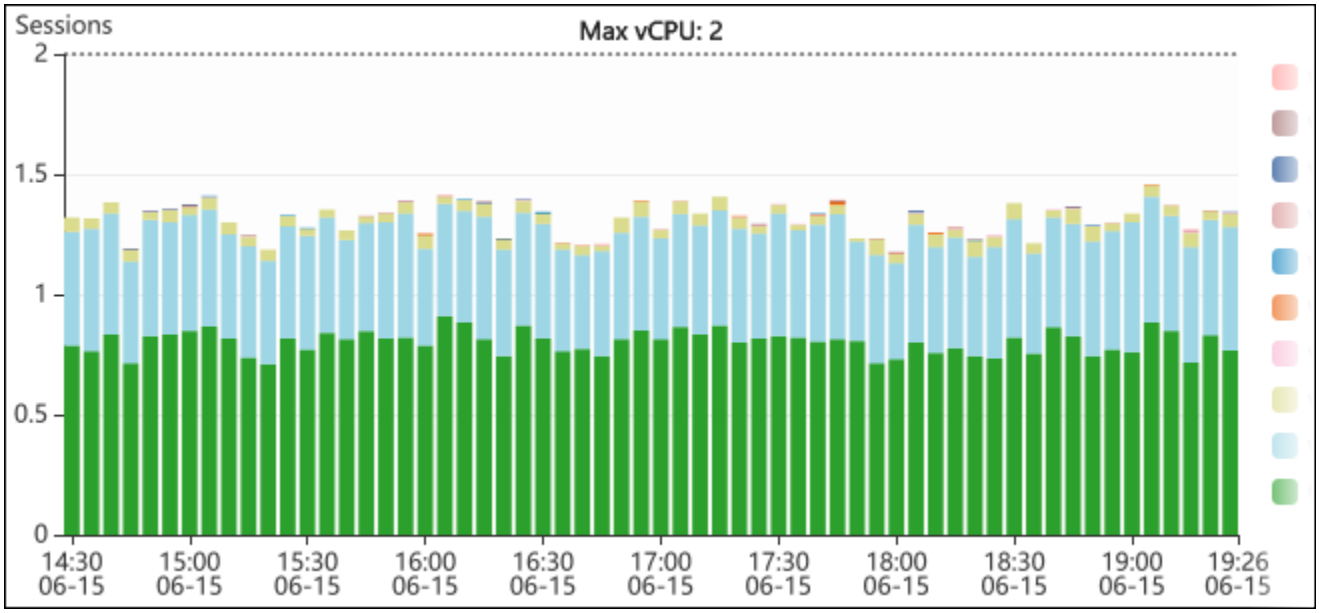
Quando il carico supera il numero di virtual CPUs (vCPUs) sulla classe dell'istanza DB, la barra diventa rossa, indicando un potenziale collo di bottiglia.



4. (Facoltativo) Scegliere la data o l'intervallo di ore in alto a destra e specificare un intervallo di tempo relativo o assoluto diverso. È ora possibile specificare un periodo di tempo e generare un report di analisi delle prestazioni del database. Il report fornisce le informazioni e i suggerimenti identificati. Per ulteriori informazioni, consulta Creazione di un rapporto di analisi delle prestazioni in Performance Insights.

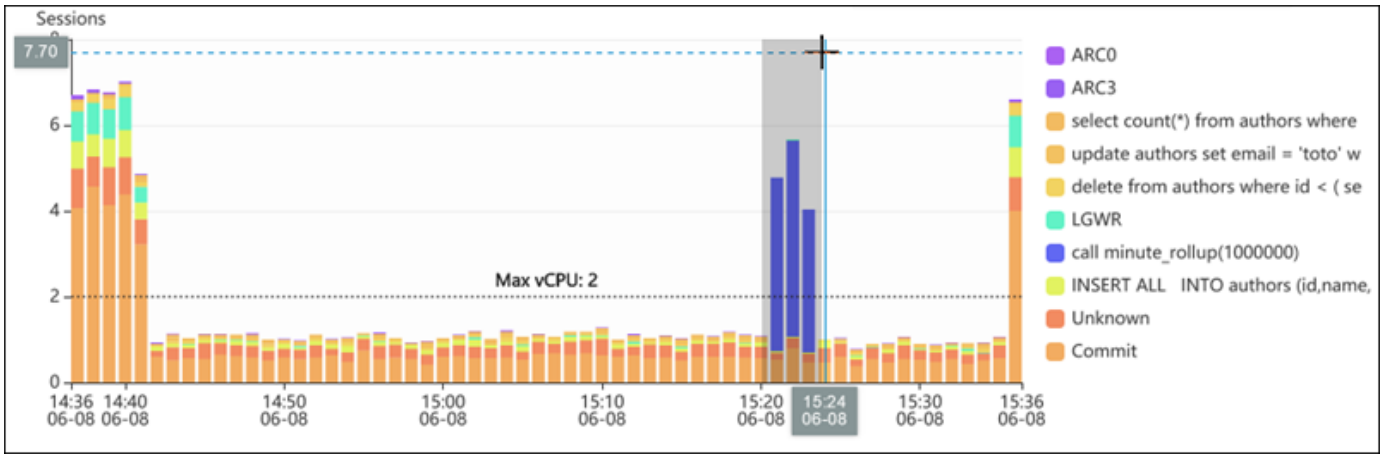


Nella schermata seguente, l'intervallo di caricamento DB è di 5 ore.

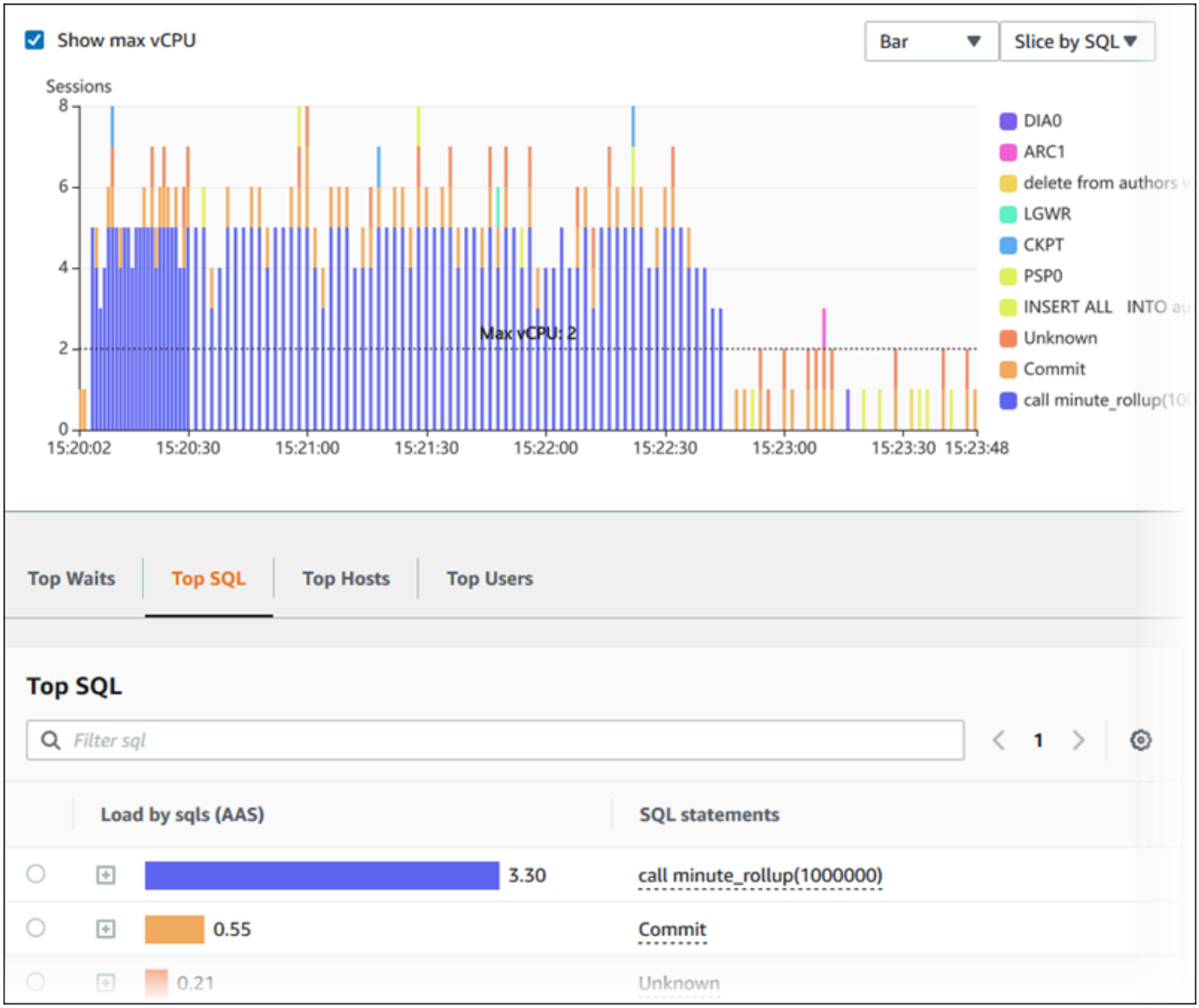


5. (Facoltativo) Per ingrandire una parte del grafico di carico del database, scegli l'ora di inizio e trascina fino alla fine del periodo di tempo che ti interessa.

   L'area selezionata viene evidenziata nel grafico del carico del database.



   Quando rilasciate il mouse, il diagramma di carico del DB ingrandisce la AWS regione selezionata e la tabella delle dimensioni principali viene ricalcolata.



6. (Facoltativo) Per aggiornare automaticamente i dati, selezionare Aggiornamento automatico.

Il pannello di controllo di Performance Insights si aggiorna automaticamente in base ai nuovi dati. La frequenza di aggiornamento dipende dalla quantità di dati visualizzati:

- Se scegli 5 minuti, l'aggiornamento avviene ogni 10 secondi.
- Se scegli 1 ora, l'aggiornamento avviene ogni 5 minuti.
- Se scegli 5 ore, l'aggiornamento avviene ogni 5 minuti.
- Se scegli 24 ore, l'aggiornamento avviene ogni 30 minuti.
- Se scegli 1 settimana, l'aggiornamento avviene ogni giorno.

- Se scegli 1 mese, l'aggiornamento avviene ogni giorno.

Analisi del carico del database per eventi di attesa

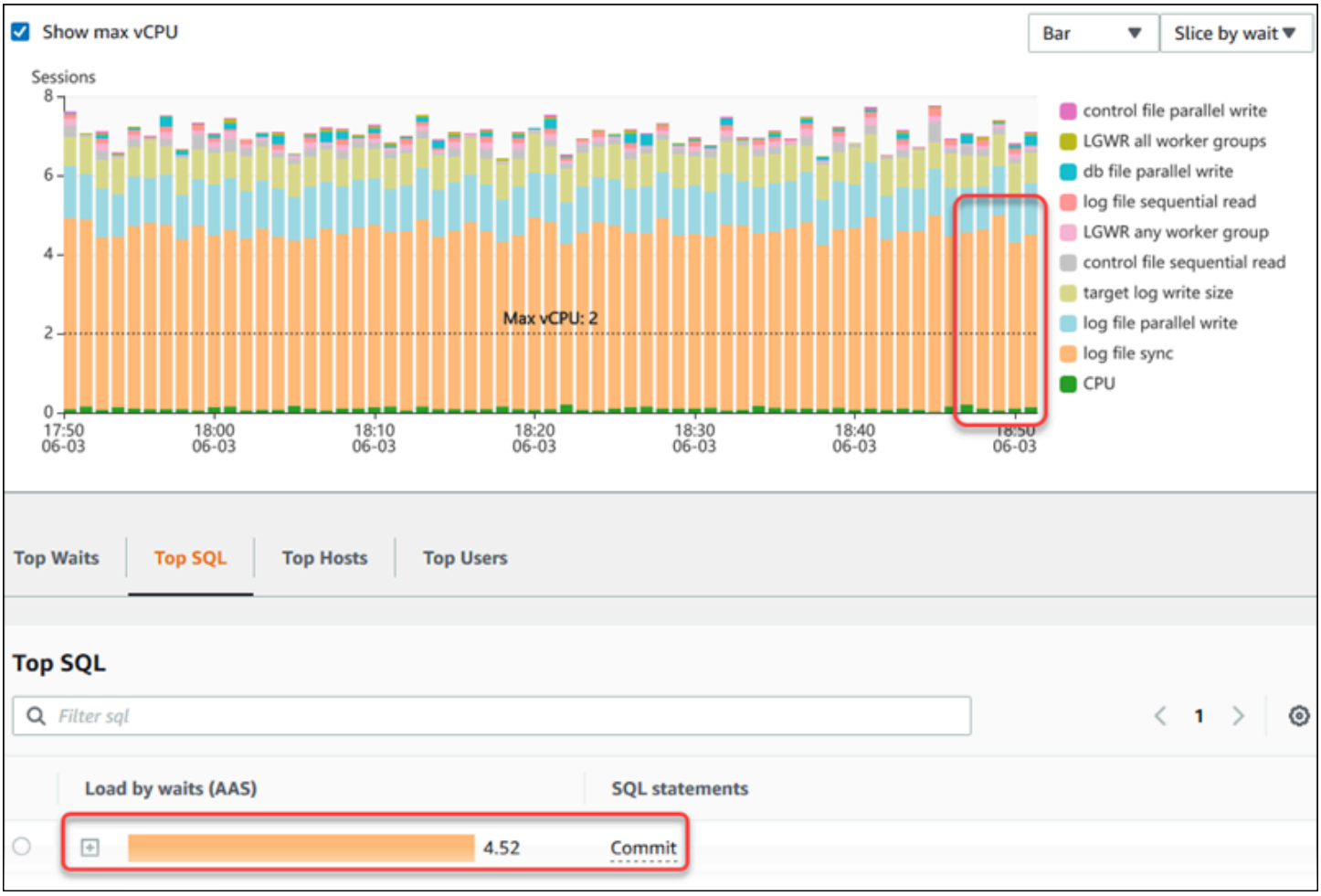
Se il grafico Database load (Caricamento database) mostra un collo di bottiglia, puoi identificare l'origine del carico. A questo scopo, osserva la tabella Top Load Items (Elementi con carico) sotto la tabella Database load (Caricamento database). Scegli un elemento particolare, ad esempio una SQL query o un utente, per approfondire quell'elemento e visualizzarne i dettagli.

Il carico del database raggruppato per attese e SQL query principali è la visualizzazione predefinita del dashboard di Performance Insights. Questa combinazione offre di norma il maggior numero di informazioni sui problemi di prestazioni. Il carico del database raggruppato in base alle attese mostra la presenza di eventuali colli di bottiglia nel database relativamente alle risorse o alla simultaneità. In questo caso, la SQLscheda della tabella degli elementi di caricamento più frequenti mostra quali query generano tale carico.

Il flusso di lavoro tipico per diagnosticare problemi di performance è il seguente:

1. Esamina il grafico di caricamento del database e verifica se ci sono incidenti di carico del database che superano la riga Max. CPU
2. Se sono presenti, osservare il grafico Database load (Caricamento database) e individuare lo stato o gli stati di attesa che sono i principali responsabili.
3. Identifica le query di riepilogo che causano il caricamento visualizzando quali delle query (la SQLscheda nella tabella degli elementi di caricamento superiore) contribuiscono maggiormente a tali stati di attesa. È possibile identificarle attraverso la colonna DB Load by Waits (Carico del database in base alle attese).
4. Scegliete una di queste query di riepilogo nella SQLscheda per espanderla e visualizzare le query secondarie da cui è composta.

Ad esempio, nel dashboard seguente, la sincronizzazione file di registro attende account per la maggior parte del carico DB. Anche l'attesa di LGWRtutti i gruppi di lavoratori è elevata. Il SQL grafico in alto mostra la causa delle attese di sincronizzazione dei file di registro: `COMMIT` dichiarazioni frequenti. In questo caso, il commit meno frequentemente ridurrà il carico del DB.



Analisi delle prestazioni del database per un periodo di tempo

Analizza le prestazioni del database con l'analisi su richiesta creando un rapporto di analisi delle prestazioni per un periodo di tempo. Visualizza i report di analisi delle prestazioni per individuare problemi relativi alle prestazioni, come problemi di risorse o modifiche a una query nell'istanza DB. Il pannello di controllo di Performance Insights consente di selezionare un periodo di tempo e creare un report di analisi delle prestazioni. Puoi anche aggiungere uno o più tag al report.

Per utilizzare questa funzionalità, devi utilizzare il periodo di conservazione del piano a pagamento. Per ulteriori informazioni, consulta Prezzi e conservazione dei dati per Performance Insights

Il report è disponibile nella scheda Report di analisi delle prestazioni - nuovi per la selezione e la visualizzazione. Il report contiene informazioni dettagliate, parametri correlati e suggerimenti per risolvere il problema relativo alle prestazioni. Il report è disponibile per la visualizzazione per tutta la durata del periodo di conservazione di Performance Insights.

Il report viene eliminato se l'ora di inizio del periodo di analisi del report è esterna al periodo di conservazione. È anche possibile eliminare il report prima della fine del periodo di conservazione.

Per individuare i problemi di prestazioni e generare il report di analisi per l'istanza database, è necessario attivare Performance Insights. Per ulteriori informazioni sull'attivazione di Performance Insights, consultare Attivazione e disattivazione di Performance Insights per Aurora.

Per informazioni sull'assistenza alla regione, al motore di database e alla classe di istanza per questa funzionalità, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Nelle sezioni seguenti, puoi creare, visualizzare, aggiungere tag ed eliminare un rapporto di analisi delle prestazioni.

Argomenti

- Creazione di un rapporto di analisi delle prestazioni in Performance Insights
- Visualizzazione di un rapporto di analisi delle prestazioni in Performance Insights
- Aggiungere tag a un rapporto di analisi delle prestazioni in Performance Insights
- Eliminazione di un rapporto di analisi delle prestazioni in Performance Insights

Creazione di un rapporto di analisi delle prestazioni in Performance Insights

È possibile creare un report di analisi delle prestazioni per un periodo specifico nella pannello di controllo di Performance Insights. È possibile selezionare un periodo di tempo e aggiungere uno o più tag al report di analisi.

Il periodo di analisi può variare da 5 minuti a 6 giorni. Occorre disporre di almeno 24 ore di dati di prestazioni prima dell'ora di inizio dell'analisi.

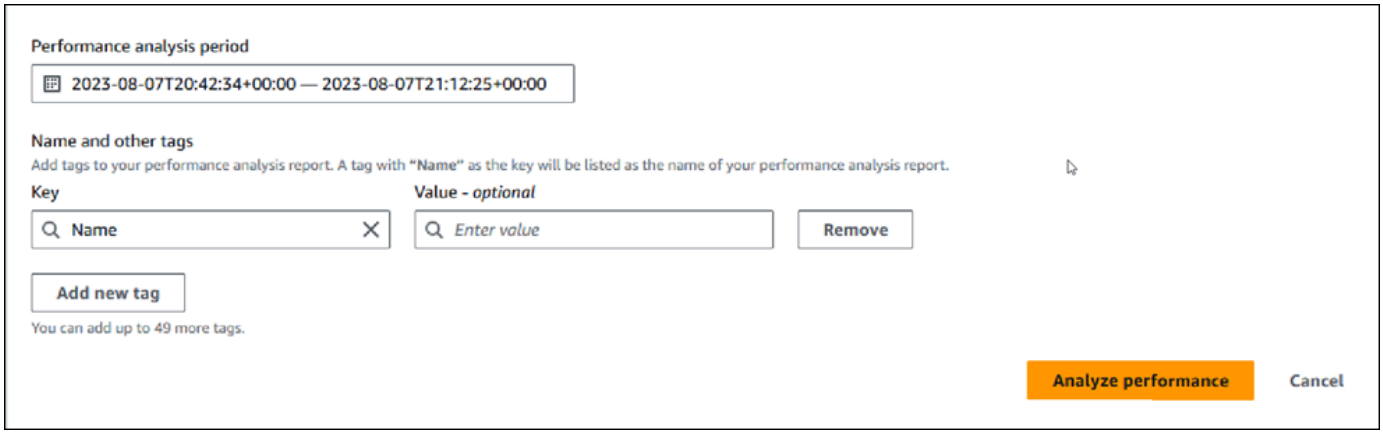
Per informazioni sull'assistenza alla regione, al motore di database e alla classe di istanza per questa funzionalità, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Per creare un report di analisi delle prestazioni per un periodo di tempo

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

4. Scegli Analizza le prestazioni nella sezione Caricamento del database nella dashboard di Performance Insights.

   Vengono visualizzati i campi per impostare il periodo di tempo e aggiungere uno o più tag al report di analisi delle prestazioni.

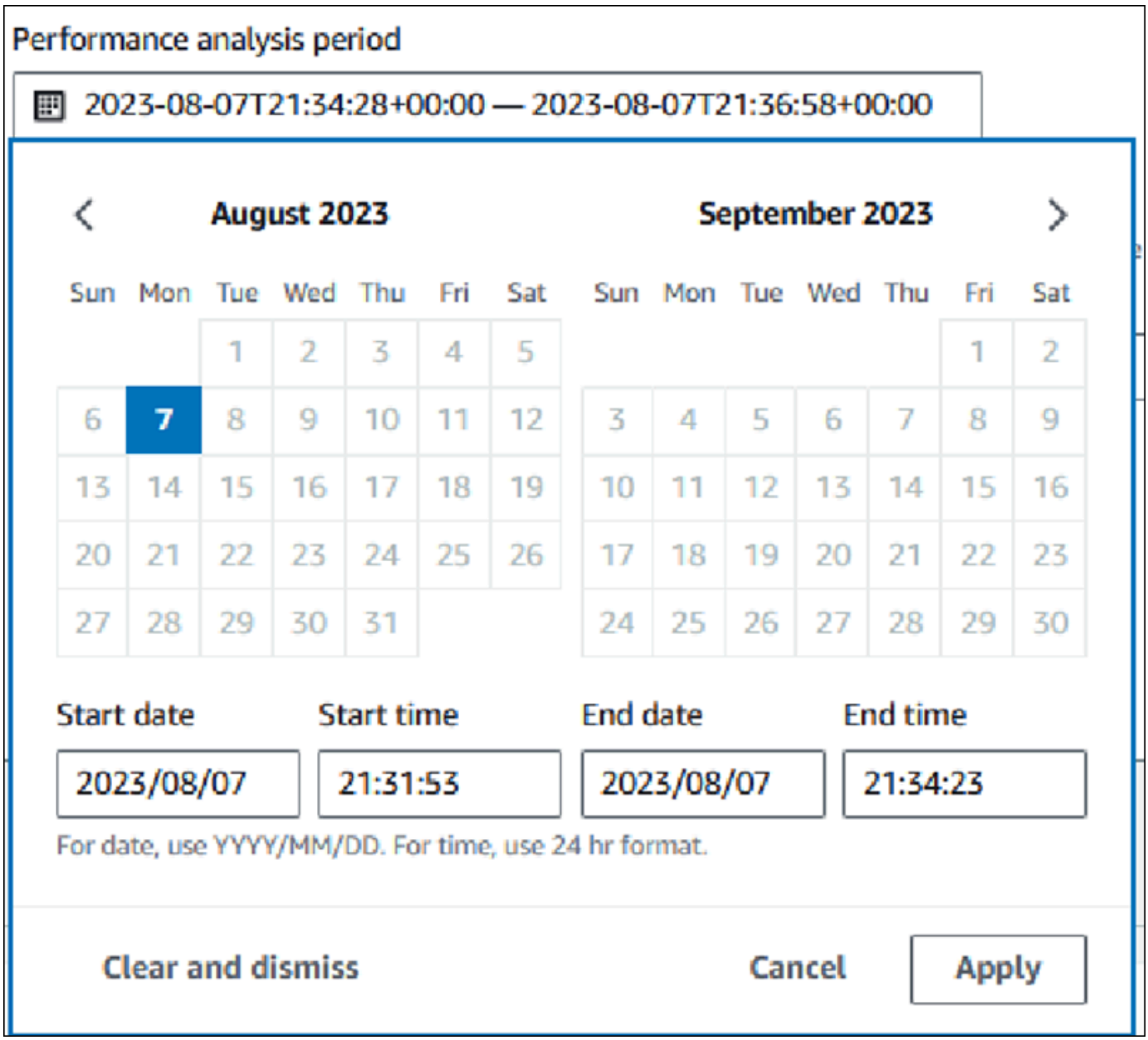


5. Scegli il periodo di tempo. Se imposti un periodo di tempo nel campo Intervallo relativo o Intervallo assoluto nell'angolo in alto a destra, puoi inserire o selezionare la data e l'ora del report di analisi solo entro questo periodo di tempo. Se selezioni il periodo di analisi al di fuori di questo periodo di tempo, viene visualizzato un messaggio di errore.

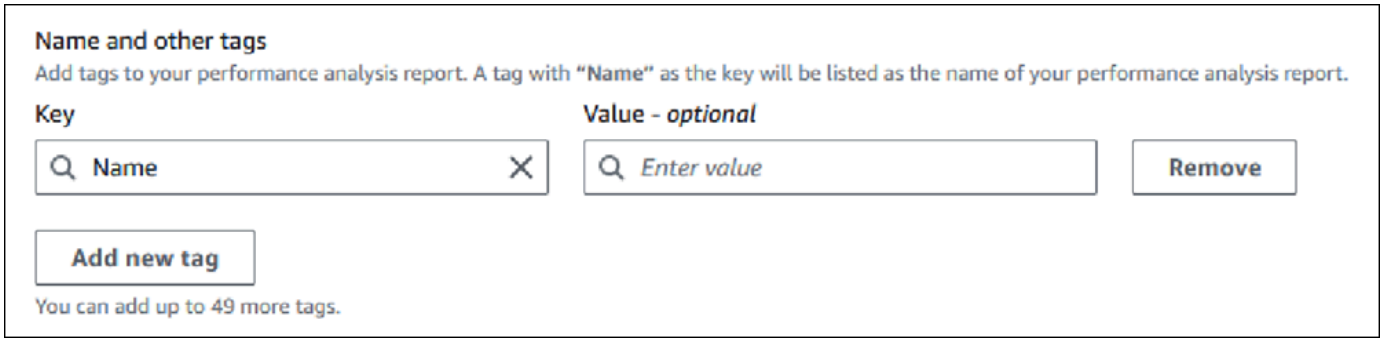
   Per impostare il periodo di tempo, puoi effettuare una delle seguenti operazioni:

   - Premi e trascina uno qualsiasi dei dispositivi di scorrimento sul grafico del carico del database.

     Nella casella Periodo di analisi delle prestazioni viene visualizzato il periodo di tempo selezionato e il grafico del carico del database evidenzia il periodo di tempo selezionato.
   - Scegli Data di inizio, Ora di inizio, Data di fine e Ora di fine nella casella Periodo di analisi delle prestazioni.



6. (Facoltativo) Inserisci Chiave e Valore-opzionale per aggiungere un tag per il report.



7. Scegli Analizza le prestazioni.

   Un banner mostra un messaggio a prescindere dall'esito della generazione del report. Il messaggio fornisce anche il collegamento per visualizzare il report.

   L'esempio seguente mostra il banner con il messaggio di creazione del report completata.

   Il report è disponibile per la visualizzazione nella scheda Report di analisi delle prestazioni - nuovi.

Puoi creare un report di analisi delle prestazioni utilizzando la AWS CLI. Per un esempio su come creare un report utilizzando AWS CLI, consultaCreazione di un report di analisi delle prestazioni per un periodo di tempo.

Visualizzazione di un rapporto di analisi delle prestazioni in Performance Insights

Nella scheda Report di analisi delle prestazioni - nuovi vengono elencati tutti i report creati per l'istanza database. Per ogni report viene visualizzato quanto segue:

- ID: identificatore univoco del report.
- Nome: chiave di tag aggiunta al report.
- Ora di creazione del report: ora in cui il report è stato creato.
- Ora di inizio dell'analisi: ora di inizio dell'analisi nel report.
- Ora di fine dell'analisi: ora di fine dell'analisi nel report.

Per visualizzare un report di analisi delle prestazioni

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database per la quale desideri visualizzare il report di analisi.
4. Scorri verso il basso e scegli Rapporti di analisi delle prestazioni: nuova scheda nella dashboard di Performance Insights.

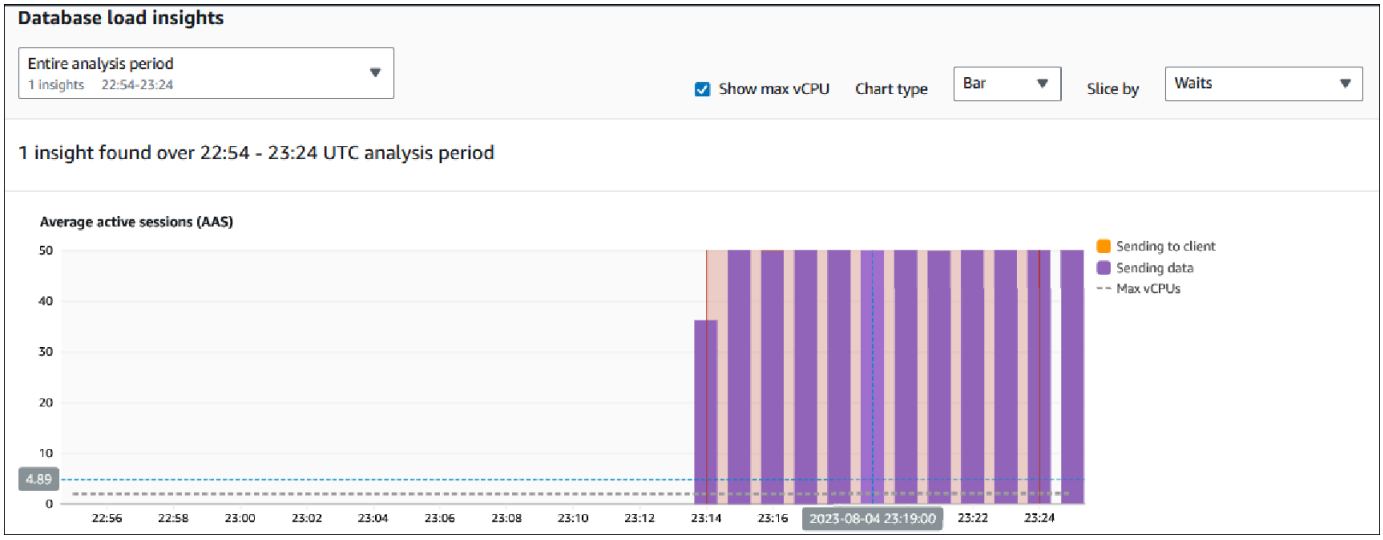
   Vengono visualizzati tutti i report di analisi per i diversi periodi di tempo.

5. Scegli ID del report che desideri visualizzare.

   Il grafico del carico del database mostra l'intero periodo di analisi per impostazione predefinita se vengono identificati più approfondimenti. Se il report ha identificato un approfondimento, il grafico del carico del database visualizza l'approfondimento per impostazione predefinita.

   Nel pannello di controllo vengono elencati anche i tag per il report nella sezione Tag.

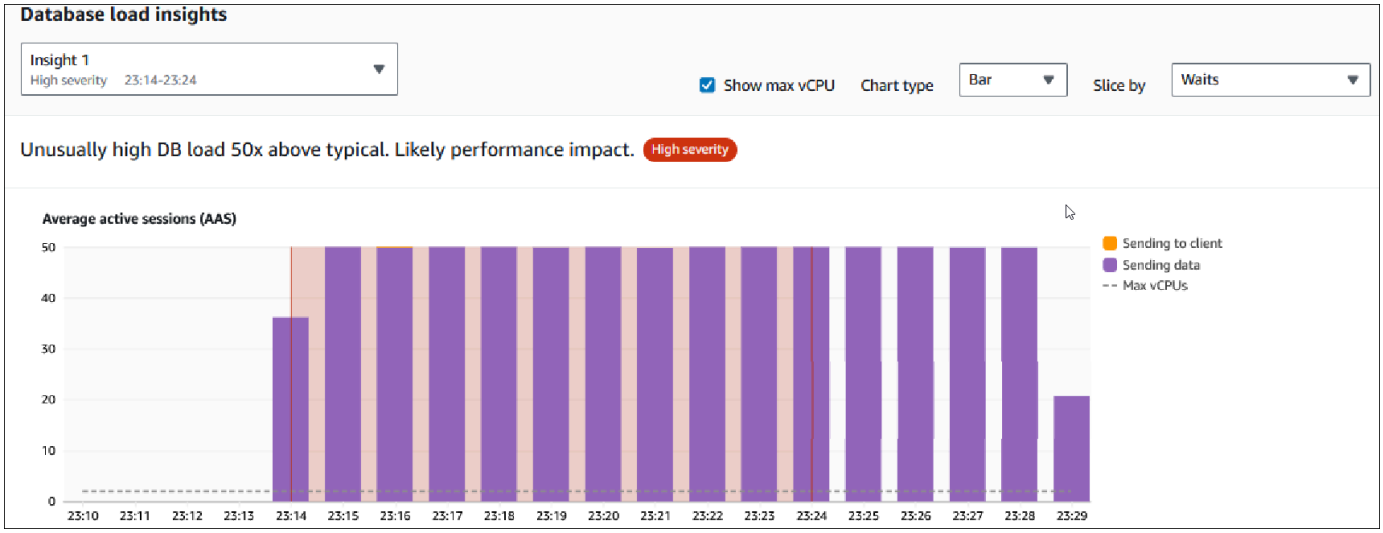
   L'esempio seguente mostra l'intero periodo di analisi per il report.

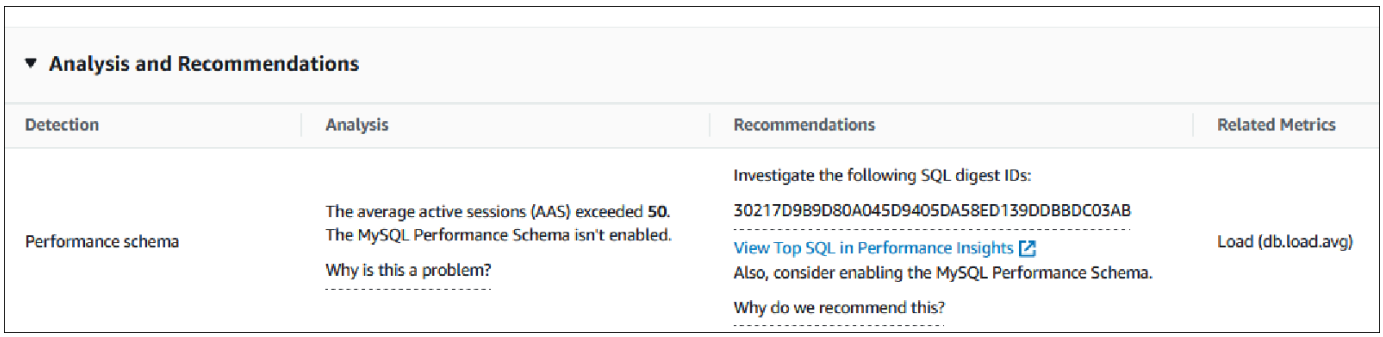


6. Scegli l'approfondimento nell'elenco Informazioni dettagliate sul carico del database che desideri visualizzare se nel report vengono identificati più approfondimenti.

   Il pannello di controllo mostra il messaggio di approfondimento, il grafico del carico del database evidenziando il periodo di tempo dell'approfondimento, l'analisi e i suggerimenti e l'elenco dei tag del report.

   Nell'esempio seguente viene mostrato l'approfondimento del carico del database nel report.





Aggiungere tag a un rapporto di analisi delle prestazioni in Performance Insights

È possibile aggiungere un tag quando si crea o visualizza un report. È possibile aggiungere fino a 50 tag per un report.

Per aggiungere i tag sono richieste autorizzazioni. Per ulteriori informazioni sulle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights

Per aggiungere uno o più tag durante la creazione di un report, consulta il passaggio 6 della procedura Creazione di un rapporto di analisi delle prestazioni in Performance Insights.

Per aggiungere uno o più tag durante la visualizzazione di un report

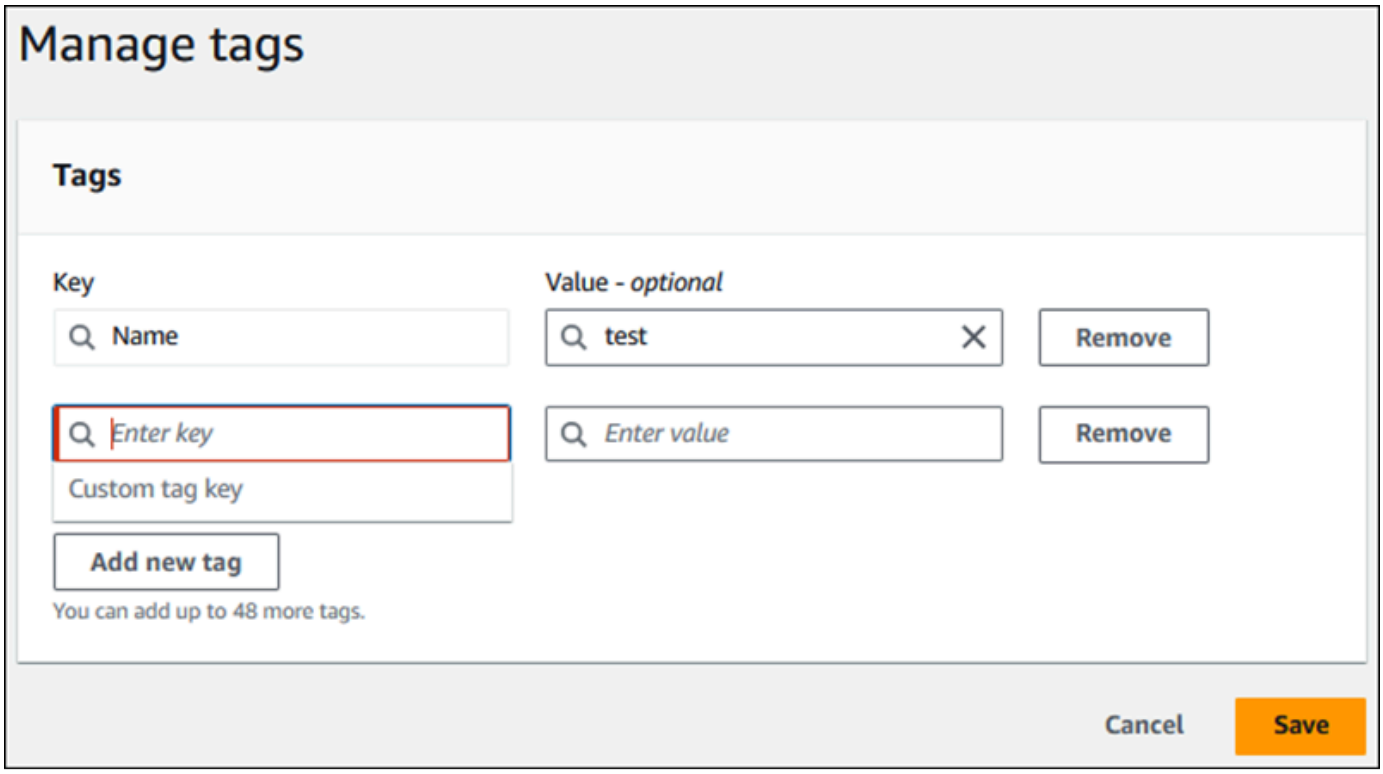
1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.

4. Scorri verso il basso e scegli la scheda Report di analisi delle prestazioni - nuovi.
5. Scegli il report per il quale desideri aggiungere i tag.

   Il pannello di controllo visualizza il report.
6. Scorri verso il basso fino a Tag e scegli Gestisci i tag.
7. Scegli Aggiungi nuovo tag.
8. Immetti Chiave e Valore-facoltativo e scegli Aggiungi nuovo tag.

   Nell'esempio seguente viene fornita la possibilità di aggiungere un nuovo tag per il report selezionato.



   Viene creato un nuovo tag per il report.

   L'elenco dei tag per il report viene visualizzato nella sezione Tag sul pannello di controllo. Se desideri rimuovere un tag dal report, scegli Rimuovi accanto al tag.

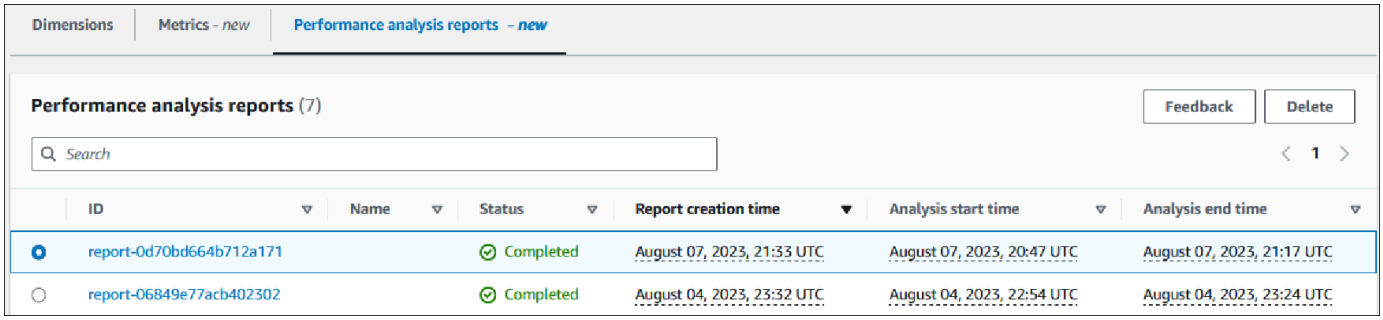
Eliminazione di un rapporto di analisi delle prestazioni in Performance Insights

È possibile eliminare un report dall'elenco dei report visualizzato nella scheda Report di analisi delle prestazioni o durante la visualizzazione di un report.

Per eliminare un report

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.
3. Scegli un'istanza database.

   Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Scorri verso il basso e scegli la scheda Report di analisi delle prestazioni - nuovi.
5. Seleziona il report che desideri eliminare e scegli Elimina nell'angolo in alto a destra.



   Viene visualizzata una finestra di conferma. Il report viene eliminato dopo aver scelto l'opzione di conferma.
6. (Facoltativo) Scegli ID del report che desideri eliminare.

   Nella pagina del report, scegli Elimina nell'angolo in alto a destra.

   Viene visualizzata una finestra di conferma. Il report viene eliminato dopo aver scelto l'opzione di conferma.

Analisi delle query con la SQL scheda Top di Performance Insights

Nella dashboard di Amazon RDS Performance Insights, puoi trovare informazioni sulle query in esecuzione e recenti nella SQL scheda Top della tabella Dimensioni principali. Queste informazioni possono essere utilizzate per ottimizzare le query.

Argomenti

- Panoramica della scheda Top SQL
- Accesso a una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights
- Visualizzazione delle SQL statistiche nella dashboard di Performance Insights

Panoramica della scheda Top SQL

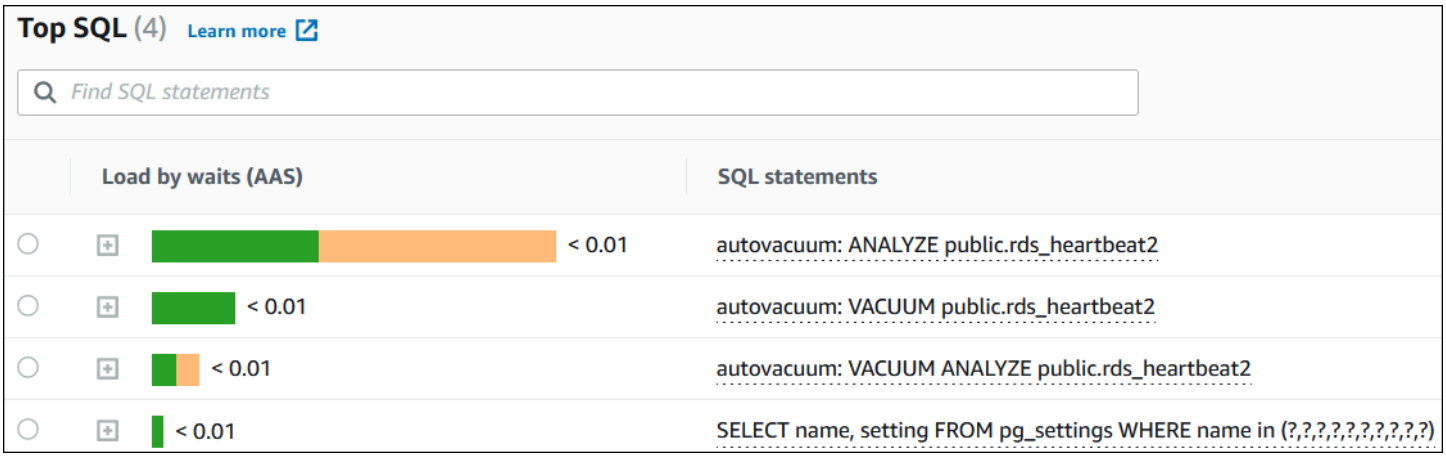
Per impostazione predefinita, la SQL scheda Top mostra le 25 query che contribuiscono maggiormente al caricamento del DB. Per ottimizzare le query, è possibile analizzare informazioni come il testo e SQL le statistiche della query. Puoi anche scegliere le statistiche che desideri visualizzare nella SQL scheda Superiore.

Argomenti

- SQLtesto
- Statistiche SQL
- Carica per attesa () AAS
- Visualizzazione delle informazioni su SQL
- Scegli le preferenze relative alle statistiche

SQLtesto

Per impostazione predefinita, ogni riga della SQL tabella Top mostra 500 byte di testo per ogni istruzione.

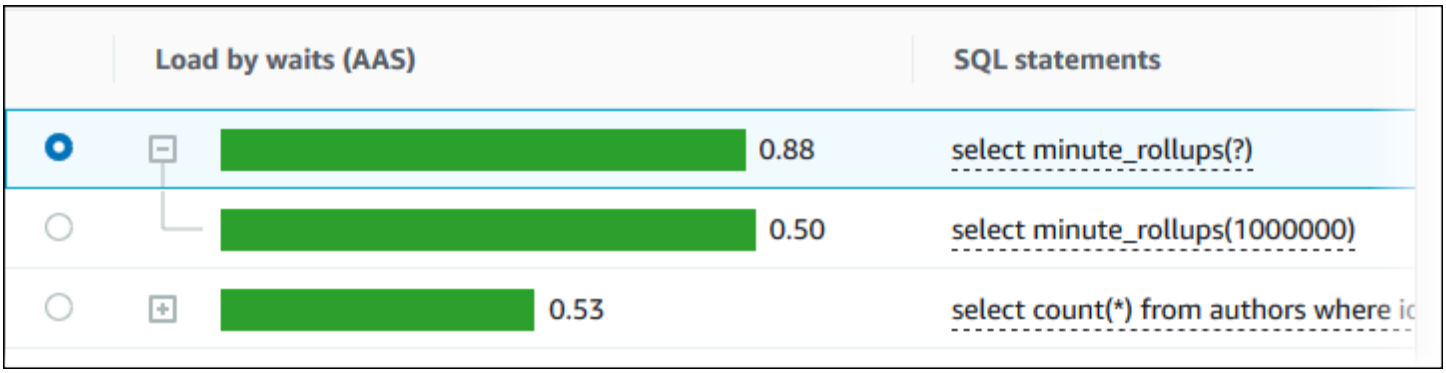


Per informazioni su come visualizzare più dei 500 byte di SQL testo predefiniti, consultaAccesso a una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights.

Un SQLdigest è un insieme di più interrogazioni effettive strutturalmente simili ma che potrebbero avere valori letterali diversi. Il digest sostituisce i valori codificati con un punto interrogativo. Ad esempio, un digest potrebbe essere `SELECT * FROM emp WHERE lname= ?`. Questo digest può includere le seguenti query figlio:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Per visualizzare le SQL istruzioni letterali in un digest, selezionate la query, quindi scegliete il simbolo più (+). Nell'esempio seguente, la query selezionata è un sunto.

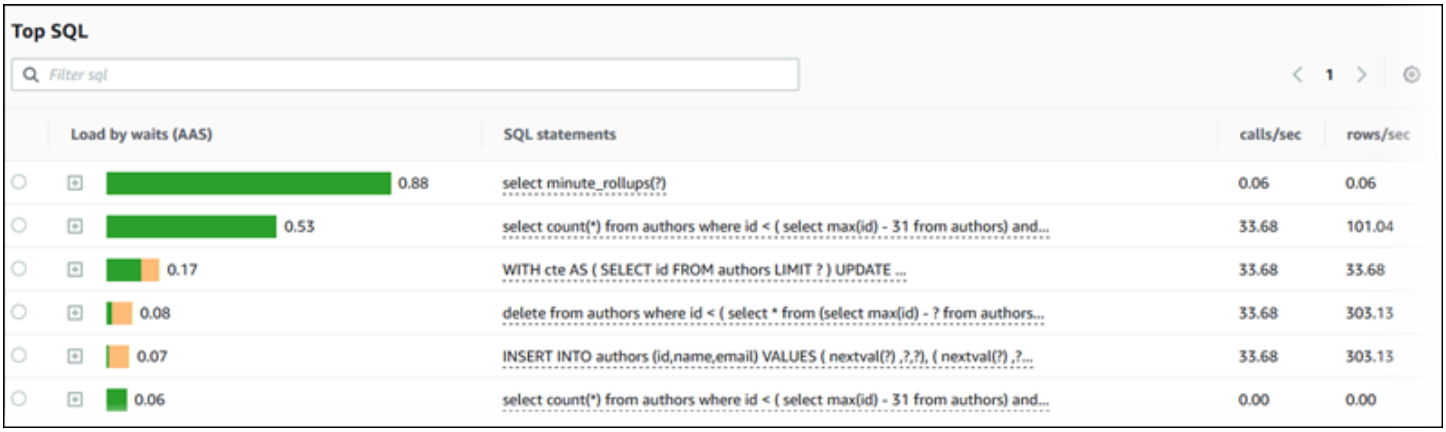


> **Note**
>
> Un SQL digest raggruppa SQL dichiarazioni simili, ma non oscura informazioni sensibili.

Statistiche SQL

SQLle statistiche sono metriche relative alle prestazioni relative alle query. SQL Ad esempio, Performance Insights potrebbe mostrare esecuzioni al secondo o righe elaborate al secondo. Performance Insights raccoglie statistiche solo per le query più comuni. In genere, queste query corrispondono alle prime query per carico mostrate nel dashboard di Performance Insights.

Ogni riga della SQL tabella Top mostra le statistiche pertinenti per l'SQListruzione o il riepilogo, come illustrato nell'esempio seguente.



Performance Insights può `0.00` generare report e SQL statistiche - (sconosciute). Questa situazione si verifica nelle seguenti condizioni:

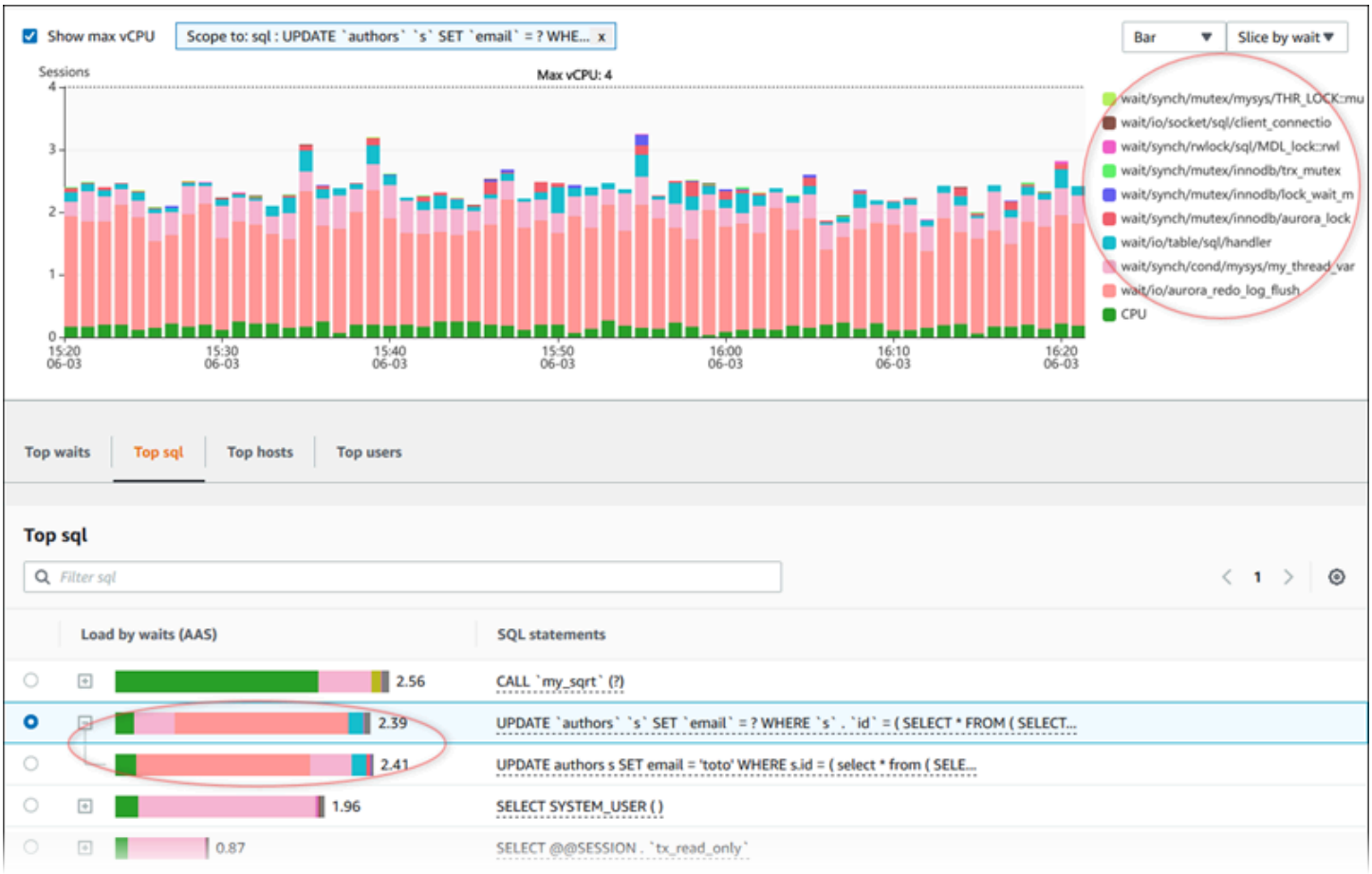
- Esiste un solo campione. Ad esempio, Performance Insights calcola i tassi di modifica per le query Aurora SQL Postgre sulla base di più campioni della vista. `pg_stat_statements` Quando un carico di lavoro viene eseguito per un breve periodo, Performance Insights potrebbe raccogliere solo un campione, il che significa che non è in grado di calcolare un tasso di varaizione. Il valore sconosciuto è rappresentato da un trattino (-).
- Due campioni hanno gli stessi valori. Performance Insights non è in grado di calcolare un tasso di variazione perché non si è verificata alcuna variazione, quindi riporta il tasso come `0.00`.
- Un'SQListruzione Aurora Postgre non dispone di un identificatore valido. Postgre SQL crea un identificatore per un'istruzione solo dopo l'analisi e l'analisi. Pertanto, un'istruzione può esistere nelle strutture SQL interne di memoria di Postgre senza identificatore. Poiché Performance Insights esegue il campionamento delle strutture interne in memoria una volta al secondo, le query a bassa latenza potrebbero apparire solo in un singolo campione. Se l'identificatore della query non è disponibile per questo campione, Performance Insights non può associare questa istruzione alle relative statistiche. Il valore sconosciuto è rappresentato da un trattino (-).

Per una descrizione delle SQL statistiche relative ai motori Aurora, consulta. Statistiche SQL per Performance Insights

Carica per attesa () AAS

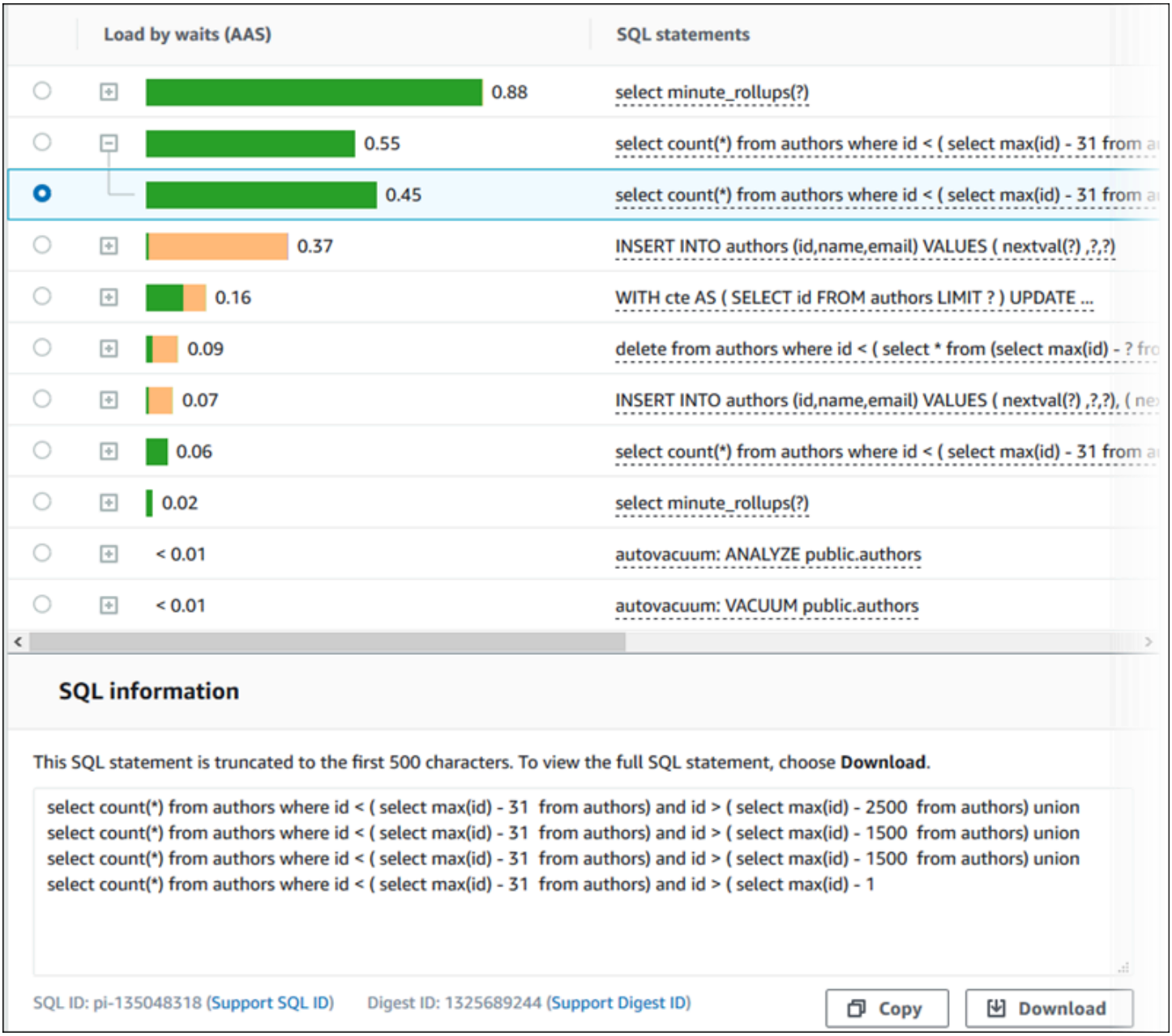
In Top SQL, la colonna Load by waits (AAS) illustra la percentuale di carico del database associata a ciascun elemento di primo caricamento. Questa colonna indica il carico per questo elemento in base a qualunque raggruppamento attualmente selezionato nel grafico DB Load. Per ulteriori informazioni sulla media delle sessioni attive (AAS), vedere. Media delle sessioni attive

Ad esempio, è possibile raggruppare il Carico DB in base agli stati di attesa. SQLLe interrogazioni vengono esaminate nella tabella degli elementi di caricamento più frequenti. In questo caso, la barra DB Load by Waits (Carico del database in base alle attese) è dimensionata, segmentata e rappresentata da un colore per mostrare qual è il contributo della query a un dato stato di attesa. Mostra anche quali stati di attesa stanno influenzando la query selezionata.



Visualizzazione delle informazioni su SQL

Nella SQL tabella Top, è possibile aprire una dichiarazione per visualizzarne le informazioni. Le informazioni vengono visualizzate nel riquadro inferiore.

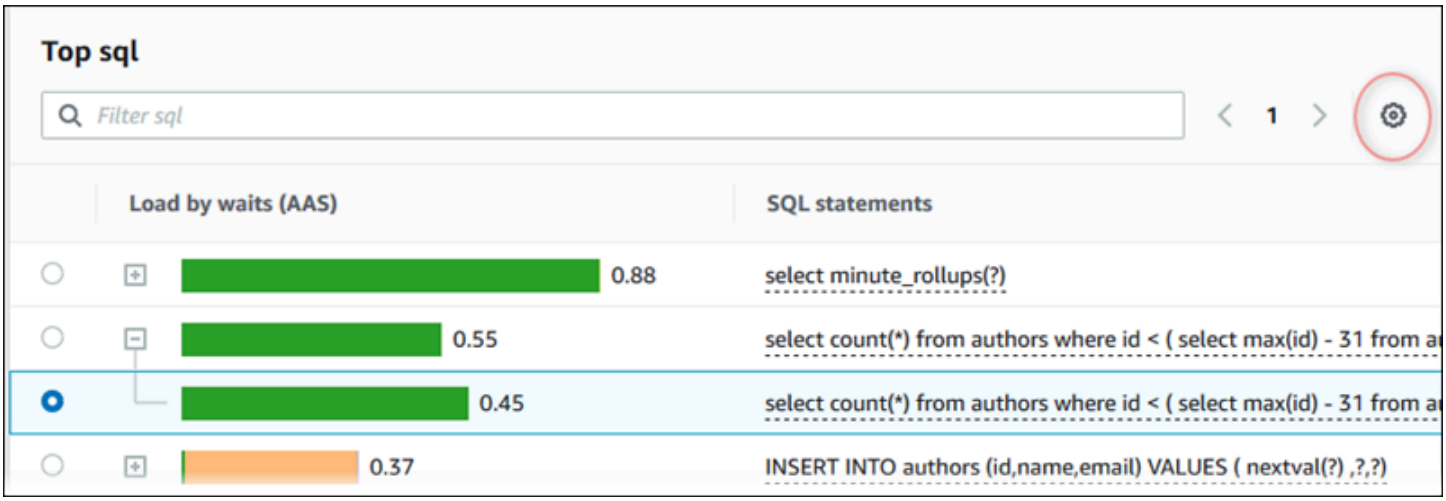


I seguenti tipi di identificatori (IDs) associati SQL alle istruzioni:

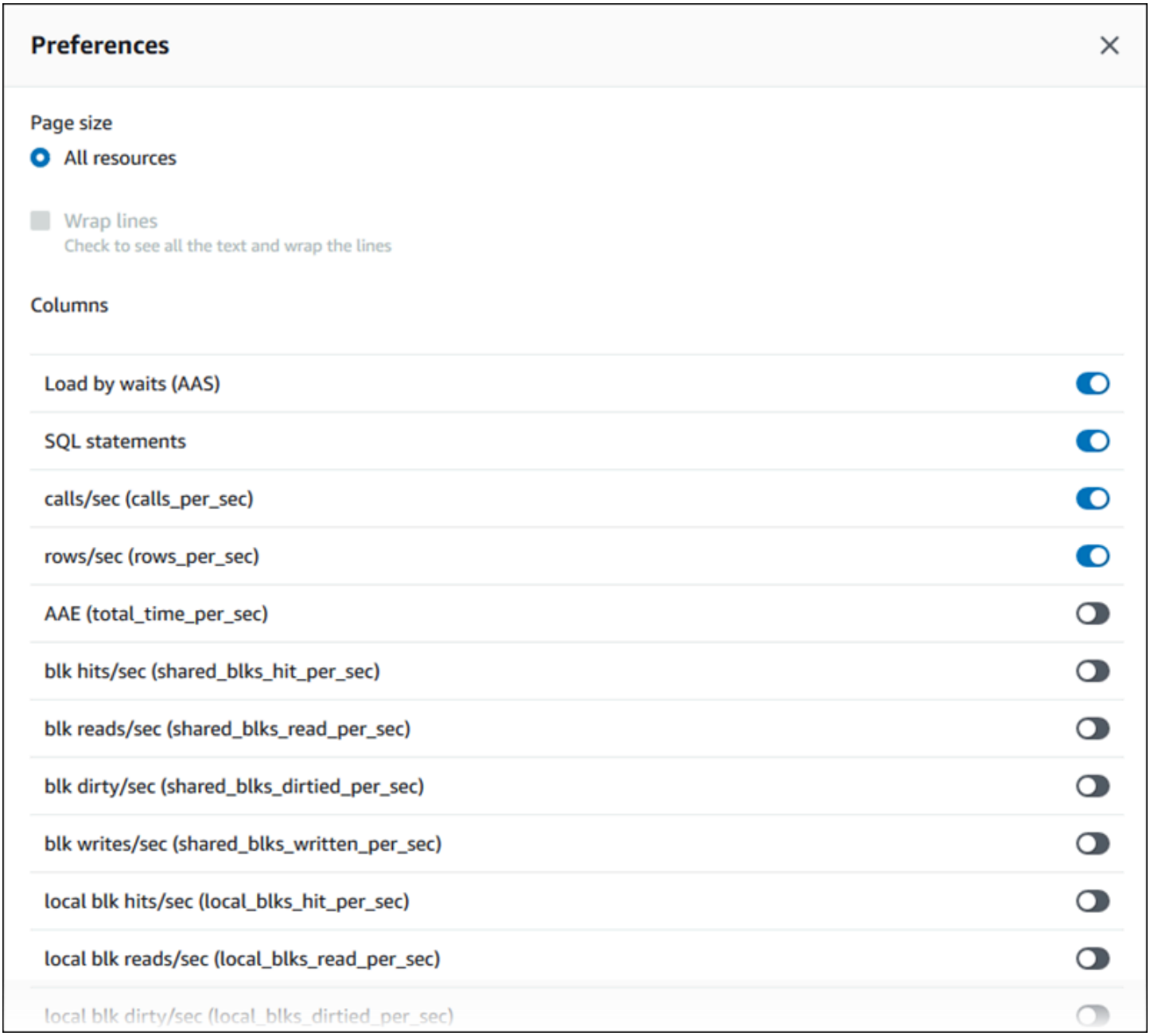
- Support SQL ID: un valore hash dell'SQLID. Questo valore serve solo per fare riferimento a un SQL ID quando si lavora con AWS Support. AWS Il supporto non ha accesso al SQL testo SQL IDs e al testo.
- ID Digest di supporto – Un valore hash dell'ID Digest. Questo valore serve solo per fare riferimento a un ID digest quando si lavora con Support AWS . AWS Il supporto non ha accesso al digest IDs e al SQL testo effettivi.

Scegli le preferenze relative alle statistiche

Puoi controllare le statistiche visualizzate nella SQL scheda Superiore scegliendo l'icona Preferenze.



Quando scegli l'icona Preferences (Preferenze), viene visualizzata la finestra Preferences (Preferenze). La schermata seguente è un esempio della finestra Preferences (Preferenze).



Per abilitare le statistiche che desideri vengano visualizzate nella SQL scheda Superiore, usa il mouse per scorrere fino alla parte inferiore della finestra, quindi scegli Continua.

Per ulteriori informazioni sulle statistiche al secondo o per chiamata per i motori Amazon , consulta la sezione relativa alle statistiche specifiche del motore in SQL Statistiche SQL per Performance Insights

Accesso a una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights

Per impostazione predefinita, ciascuna riga nella tabella Top SQL (Prime istruzioni SQL) mostra 500 byte di testo SQL per ciascuna istruzione SQL.

Quando un'istruzione SQL supera i 500 byte, puoi visualizzare più testo nella sezione SQL text (Testo SQL) sotto la tabella Top SQL (Prime istruzioni SQL). In questo caso, la lunghezza massima per il testo visualizzato in SQL text (Testo SQL) è 4 KB. Questo limite viene introdotto dalla console ed è soggetto ai limiti impostati dal motore del database. Per salvare il testo visualizzato in SQL text (Testo SQL), scegli Download (Scarica).

Argomenti

- Limiti delle dimensioni del testo per i Aurora MySQL
- Impostazione del limite di SQL testo per le istanze Postgre SQL
- Visualizzazione e download di SQL testo nella dashboard di Performance Insights

Limiti delle dimensioni del testo per i Aurora MySQL

Durante il download di testo SQL, il motore del database determina la sua lunghezza massima. Puoi scaricare il testo SQL fino ai seguenti limiti per motore.

Motore database	Lunghezza massima del testo scaricato
Aurora MySQL	La lunghezza è fissa a 4.096 byte.

La sezione SQL Text (Testo SQL) della console Performance Insights visualizza fino al massimo restituito dal motore. Ad esempio, se Aurora MySQL restituisce al massimo 1 KB a Performance Insights, può raccogliere e mostrare solo 1 KB, anche se la query originale è più grande. Pertanto, quando la query viene visualizzata in SQL text (Testo SQL) o scaricata, Performance Insights restituisce lo stesso numero di byte.

Se utilizzi l'API AWS CLI o, Performance Insights non ha il limite di 4 KB imposto dalla console. `DescribeDimensionKeyse GetResourceMetrics` restituiscono al massimo 500 byte.

> **Note**
>
> `GetDimensionKeyDetails`restituisce la query completa, ma la dimensione è soggetta al limite del motore.

Impostazione del limite di SQL testo per le istanze Postgre SQL

gestisce il testo in modo diverso. SQL È possibile impostare il limite delle dimensioni del testo con il parametro di istanza database `track_activity_query_size`. Questo parametro presenta le caratteristiche seguenti:

Dimensione di default del testo

Su Postgre SQL versione 9.6, l'impostazione predefinita per il parametro è 1.024 byte. `track_activity_query_size` Su versione 10 o successiva, il valore predefinito è 4.096 byte.

Dimensione massima del testo

Il limite `track_activity_query_size` è di 102.400 byte per SQL Il massimo è di 1 MB per la versione 13 e quelle successive.

Se il motore restituisce 1 MB a Performance Insights, la console visualizza solo i primi 4 KB. Se si scarica la query, si ottiene 1 MB per intero. In questo caso, la visualizzazione e il download restituiscono un numero diverso di byte. Per ulteriori informazioni sul parametro dell'istanza `track_activity_query_size` DB, consulta Runtime Statistics nella documentazione di Postgre. SQL

Per aumentare la dimensione del SQL testo, aumenta il limite. `track_activity_query_size` Per modificare il parametro, modifica l'impostazione del parametro nel gruppo di parametri associato all'istanza DB Postgre. SQL

Modifica dell'impostazione quando l'istanza utilizza il gruppo di parametri di default

1. Crea un nuovo gruppo di parametri dell'istanza database per il motore del database e la versione del motore del database appropriati.
2. Imposta il parametro nel nuovo gruppo di parametri.
3. Associa il nuovo gruppo di parametri all'istanza database.

Per ulteriori informazioni sull'impostazione di un parametro dell'istanza database, consulta Modifica dei parametri in un gruppo di parametri DB in .

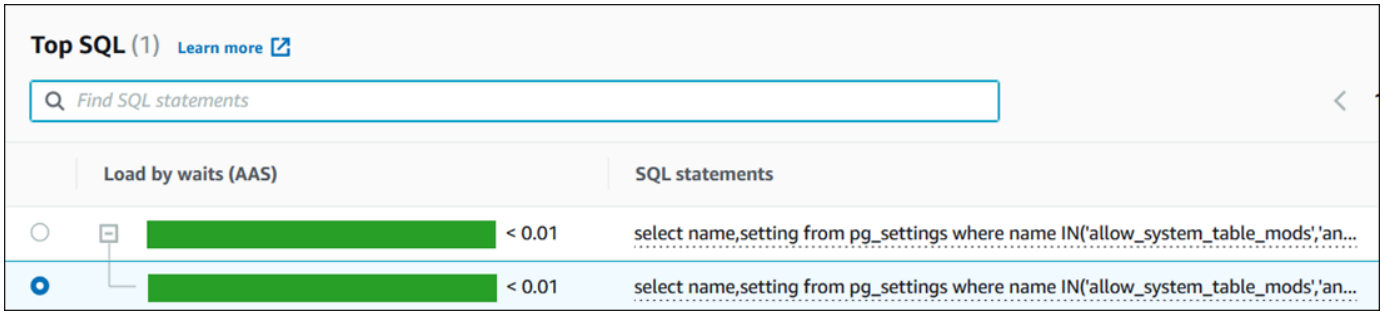
Visualizzazione e download di SQL testo nella dashboard di Performance Insights

Nella dashboard di Performance Insights, puoi visualizzare o scaricare SQL testo.

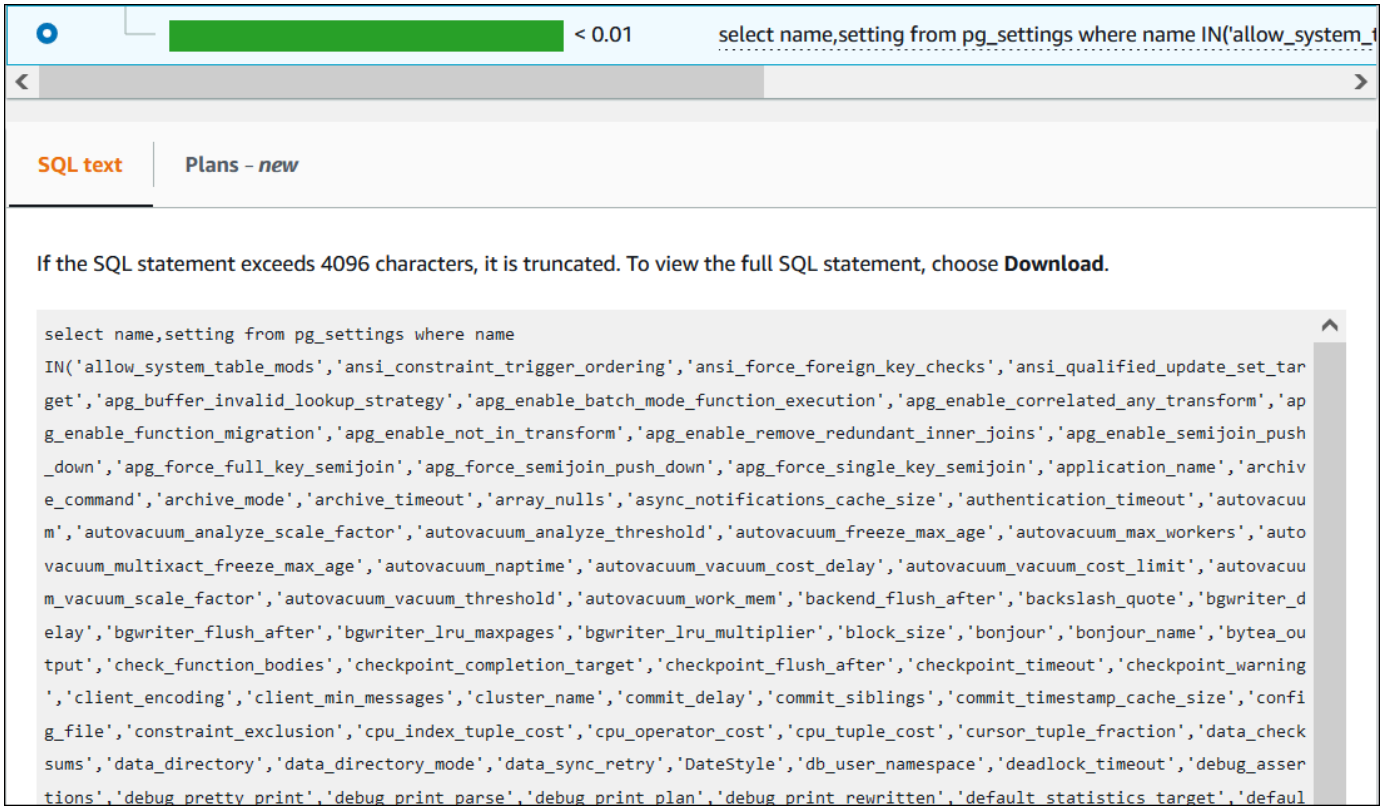
Per visualizzare altro SQL testo nella dashboard di Performance Insights

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database.
4. Scorri verso il basso fino alla SQL scheda Top della dashboard di Performance Insights.
5. Scegli il segno più per espandere un SQL digest e scegli una delle domande secondarie del digest.

   SQLIe istruzioni con testo di dimensioni superiori a 500 byte sono simili all'immagine seguente.



6. Scorri verso il basso fino alla scheda di SQLtesto.



Il pannello di controllo Performance Insights può visualizzare fino a 4.096 byte per ogni istruzione. SQL

7. (Facoltativo) Scegliete Copia per copiare l'SQListruzione visualizzata oppure scegliete Scarica per scaricare l'SQListruzione e visualizzare il SQL testo fino al limite del motore DB.

> **Note**
>
> Per copiare o scaricare l'SQListruzione, disattiva i blocchi popup.

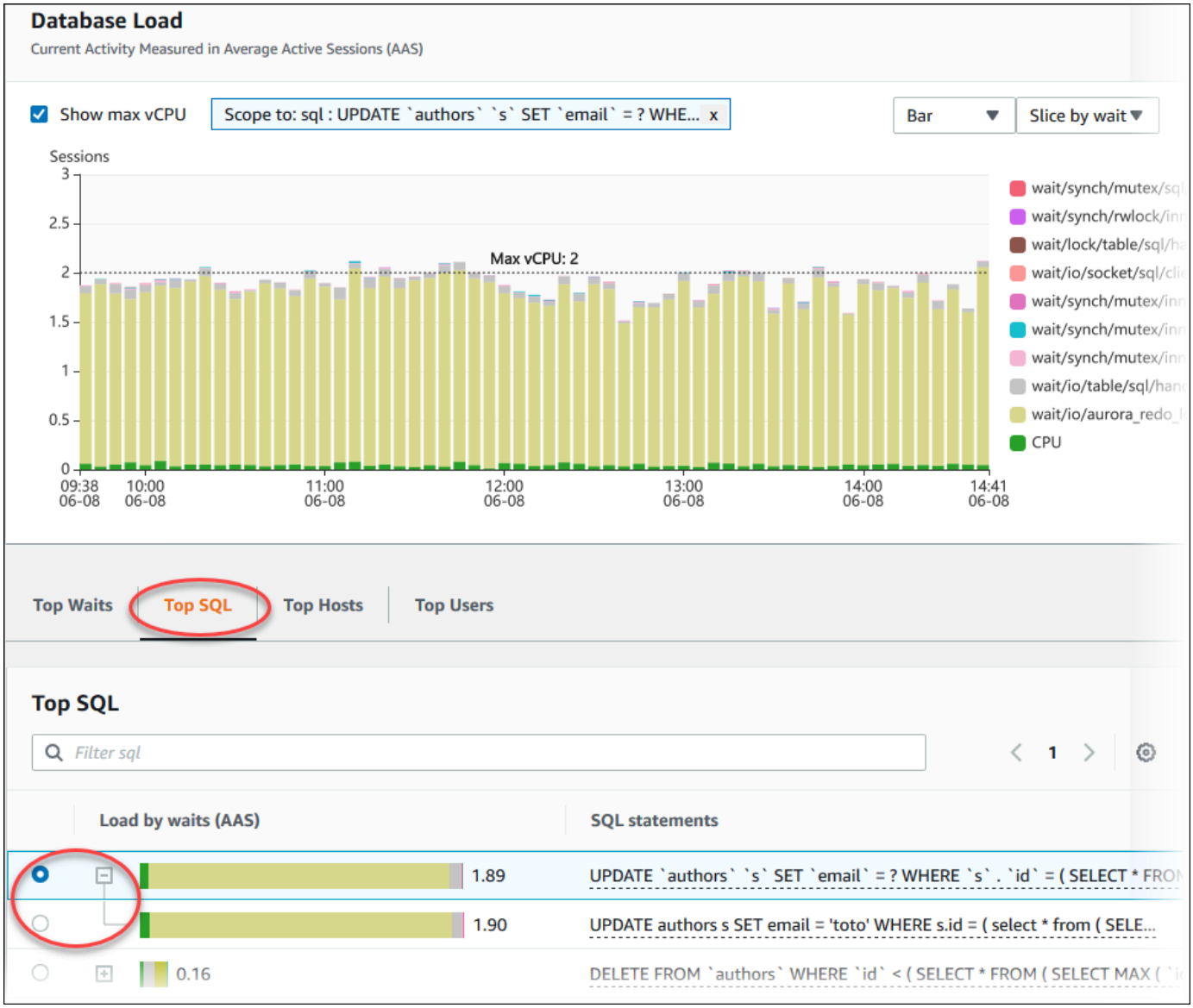
Visualizzazione delle SQL statistiche nella dashboard di Performance Insights

Nella dashboard di Performance Insights, SQL le statistiche sono disponibili nella SQL scheda Superiore del grafico di carico del database.

Per visualizzare le statistiche SQL

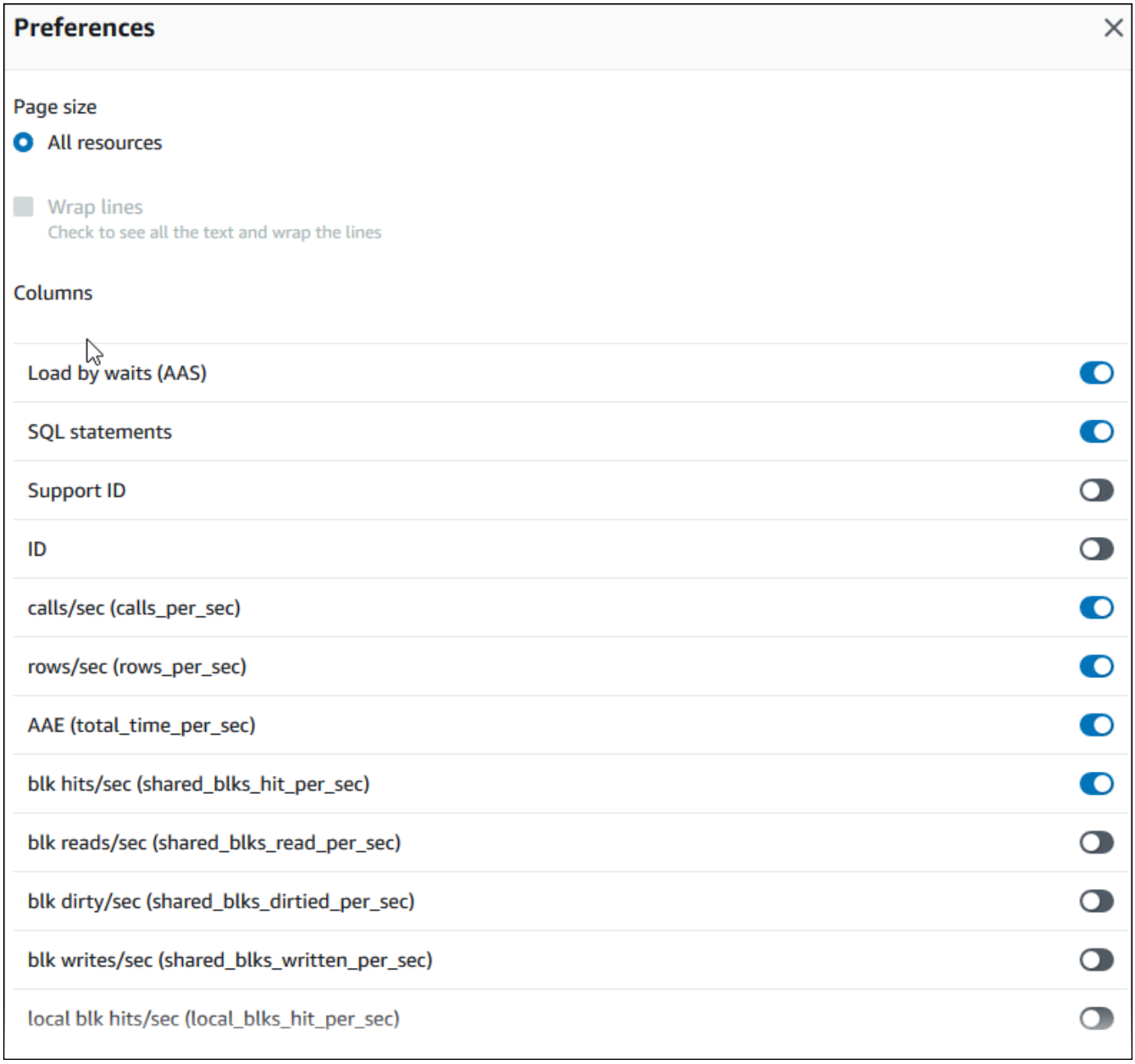
1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Performance Insights.

3. Nella parte superiore della pagina, scegli il database di cui desideri visualizzare SQL le statistiche.
4. Scorri fino alla fine della pagina e scegli la SQL scheda Superiore.
5. Scegli una singola dichiarazione (SQLsolo Aurora My) o una query digest.

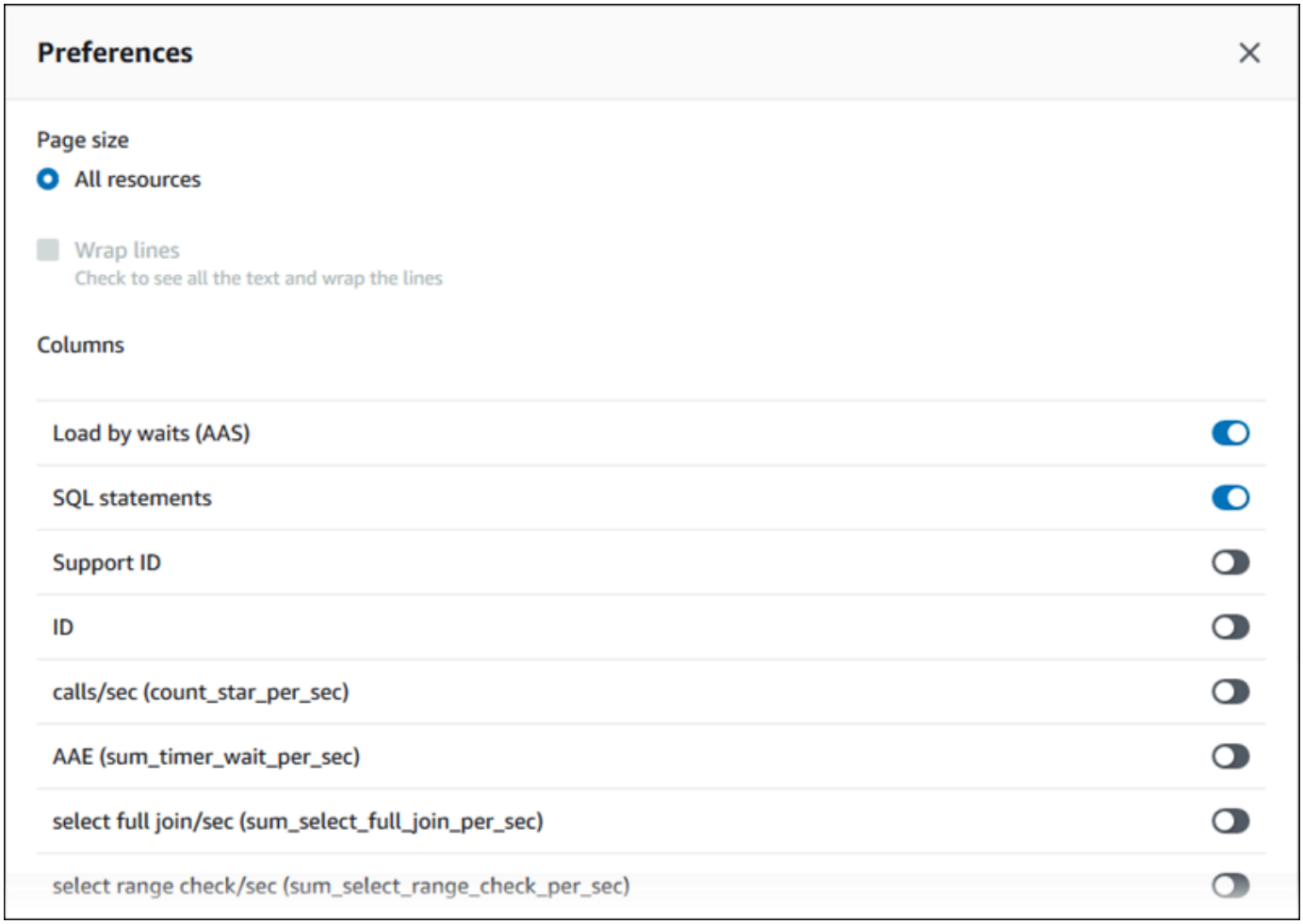


6. Scegliere le statistiche da visualizzare selezionando l'icona a forma di ingranaggio nell'angolo in alto a destra del grafico. Per le descrizioni delle SQL statistiche relative ai motori Amazon RDS Aurora, consulta. Statistiche SQL per Performance Insights

   L'esempio seguente mostra le preferenze per Aurora Postgre. SQL



L'esempio seguente mostra le preferenze per le istanze DB. SQL



7. Per salvare le preferenze, scegli Save (Salva).

   SQLLa tabella Top si aggiorna.

Visualizzazione dei consigli proattivi di Performance Insights

Amazon RDS Performance Insights monitora parametri specifici e crea automaticamente soglie analizzando quali livelli potrebbero essere potenzialmente problematici per una risorsa specifica. Quando i nuovi valori delle metriche superano una soglia predefinita in un determinato periodo di tempo, Performance Insights genera una raccomandazione proattiva. Questa raccomandazione aiuta a prevenire futuri impatti sulle prestazioni del database. Per ricevere questi consigli proattivi, devi attivare Performance Insights con un periodo di conservazione a pagamento.

Per ulteriori informazioni sull'attivazione di Performance Insights, consultare Attivazione e disattivazione di Performance Insights per Aurora. Per informazioni sui prezzi e sulla conservazione dei dati per Performance Insights, consultaPrezzi e conservazione dei dati per Performance Insights.

Per scoprire le regioni, i motori DB e le classi di istanze supportate per i consigli proattivi, consultaSupporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights.

È possibile visualizzare l'analisi dettagliata e le indagini consigliate sui consigli proattivi nella pagina dei dettagli dei consigli.

Per ulteriori informazioni sui consigli, vedere.

Per visualizzare l'analisi dettagliata di una raccomandazione proattiva

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, esegui una delle seguenti operazioni:
   - Scegliete Consigli.

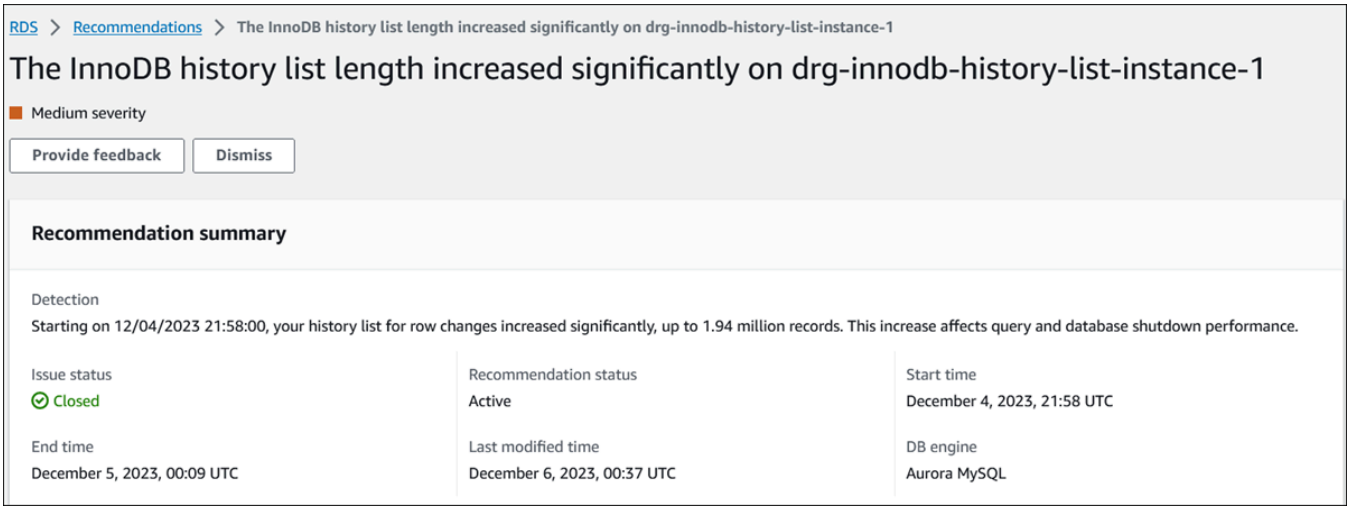
     La pagina Consigli mostra un elenco di consigli ordinati in base alla gravità per tutte le risorse del tuo account.
   - Scegli Database, quindi scegli Consigli per una risorsa nella pagina dei database.

     La scheda Consigli mostra i consigli e i relativi dettagli per la risorsa selezionata.
3. Trova un consiglio proattivo e scegli Visualizza dettagli.

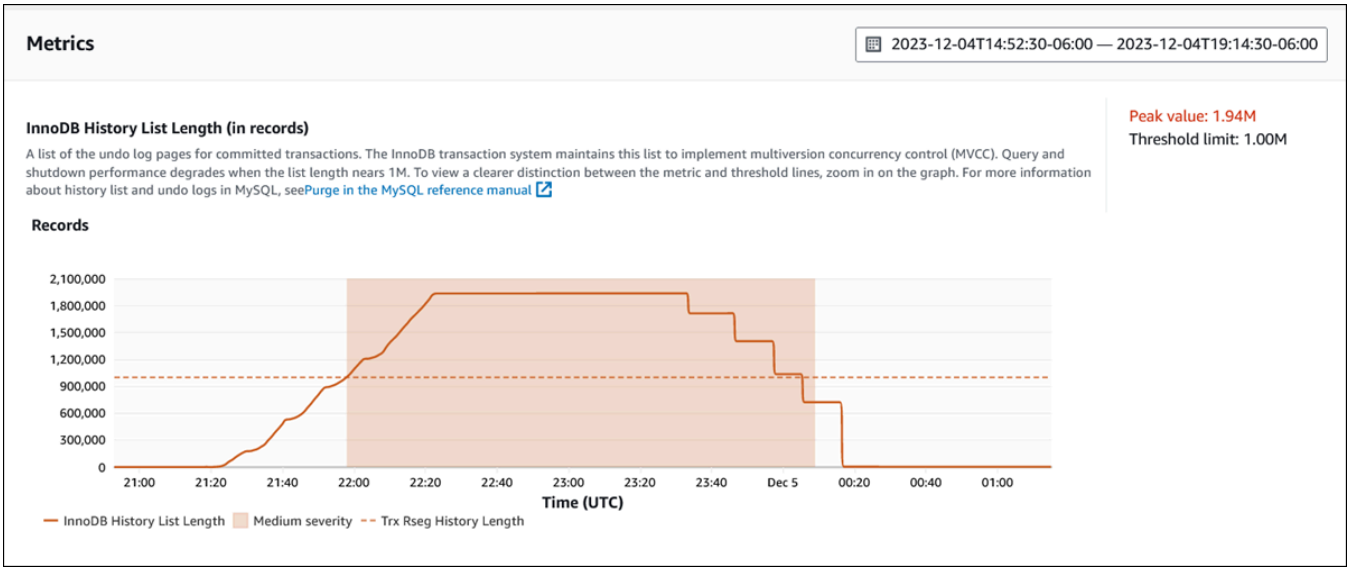
   Viene visualizzata la pagina dei dettagli del consiglio. Il titolo fornisce il nome della risorsa interessata con il problema rilevato e la gravità.

   Di seguito sono riportati i componenti della pagina dei dettagli dei consigli:

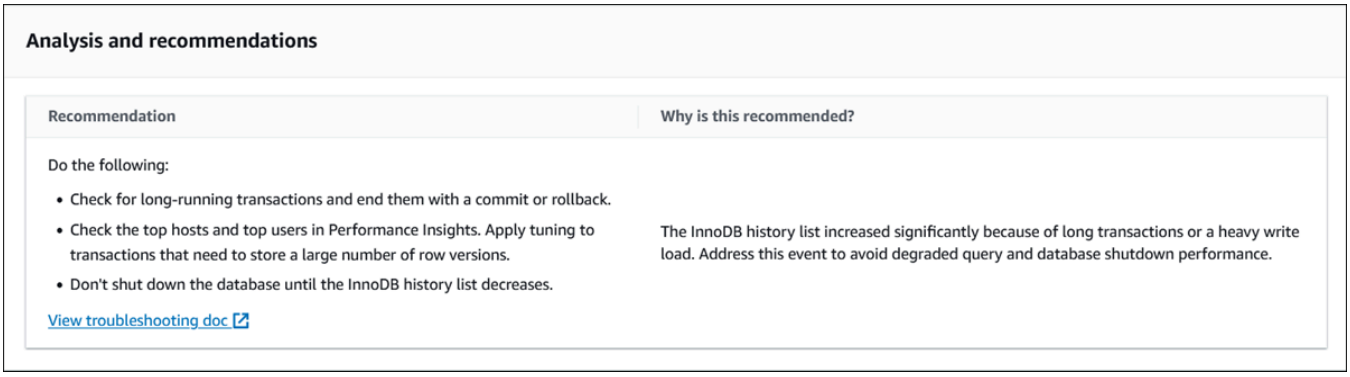
   - Riepilogo dei consigli: il problema rilevato, lo stato del suggerimento e del problema, l'ora di inizio e di fine del problema, l'ora di modifica del suggerimento e il tipo di motore.



- Metriche: i grafici del problema rilevato. Ogni grafico mostra una soglia determinata dal comportamento di base della risorsa e dai dati della metrica riportata dall'ora di inizio del problema.



- Analisi e raccomandazioni: la raccomandazione e il motivo della raccomandazione suggerita.



Puoi esaminare la causa del problema e quindi eseguire le azioni consigliate suggerite per risolvere il problema oppure scegliere Ignora in alto a destra per ignorare il consiglio.

Recupero delle metriche con Performance Insights for Aurora API

Quando Performance Insights è attivo, API fornisce visibilità sulle prestazioni dell'istanza. Amazon CloudWatch Logs fornisce la fonte autorevole per i parametri di monitoraggio dei servizi forniti. AWS

Performance Insights offre una visualizzazione specifica del dominio del carico del database misurato come sessioni attive medie (). AAS Questa metrica appare ai API consumatori come un set di dati bidimensionale di serie temporali. La dimensione temporale dei dati fornisce i dati relativi al carico del database per ogni momento dell'intervallo di tempo in cui è stata eseguita la query. Ogni punto temporale scompone il carico complessivo in relazione alle dimensioni richieste, come `SQL`, `Wait-event`, `User` o `Host`, misurato in corrispondenza di quel punto temporale.

Amazon RDS Performance Insights monitora il cluster per consentirti di analizzare e risolvere i problemi delle prestazioni del database. Un modo per visualizzare i dati di Performance Insights è disponibile nella AWS Management Console. Performance Insights fornisce anche un pubblico API in modo da poter interrogare i propri dati. È possibile utilizzare il API per effettuare le seguenti operazioni:

- Scaricamento dei dati in un database
- Aggiungi dati Performance Insights ai pannelli di controllo di monitoraggio esistenti
- Crea strumenti di monitoraggio

Per utilizzare Performance InsightsAPI, abilita Performance Insights su una delle tue istanze Amazon RDS DB. Per informazioni sull'abilitazione di Performance Insights, consulta Attivazione e disattivazione di Performance Insights per Aurora. Per ulteriori informazioni su Performance InsightsAPI, consulta Amazon RDS Performance Insights API Reference.

Performance Insights API fornisce le seguenti operazioni.

Operazione di Performance Insights	AWS CLI comando	Descrizione
CreatePerformanceAnalysisReport	aws pi create-performance-analysis-report	Crea un report di analisi delle prestazioni per un periodo di tempo specifico per l'istanza database. Il risultato è `AnalysisReportId` che è l'identificatore univoco del report.
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	Elimina un report di analisi delle prestazioni.
DescribeDimensionKeys	aws pi describe-dimension-keys	Recupera le prime N chiavi di dimensione per un parametro per un determinato periodo di tempo.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Recupera gli attributi del gruppo di dimensioni specificato per un'istanza database o un'origine dati. Ad esempio, se si specifica un SQL ID e se i dettagli della dimensione sono disponibili, `GetDimensionKeyDetails` recupera il testo completo della dimensione `db.sql.statement` associata a questo ID. Questa operazione è utile perché `GetResourceMetrics` `DescribeDimensionKeys` non supporta il recupero di testo di

Operazione di Performance Insights	AWS CLI comando	Descrizione
		istruzioni di grandi dimension iSQL.
GetPerformanceAnalysisReport	aws pi get-performance-analysis-report	Recupera il report, comprese le informazioni dettagliate relative al report. Il risultato include lo stato del report, l'ID del report, i dettagli sull'ora del report, le informazioni dettagliate e i suggerimenti.
GetResourceMetadata	aws pi get-resource-metadata	Recupera i metadati per diverse caratteristiche. Ad esempio, i metadati potrebbero indicare che una caratteristica è attivata o disattivata su un'istanza database specifica.
GetResourceMetrics	aws pi get-resource-metrics	Recupera parametri Performance Insights per un set di origini dati, su un periodo di tempo. Puoi fornire gruppi di dimensioni e dimensioni specifiche e fornire criteri di aggregazione e filtro per ogni gruppo.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Recupera le dimensioni su cui è possibile eseguire query per ogni tipo di parametro specificato su un'istanza specificata.

Operazione di Performance Insights	AWS CLI comando	Descrizione
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Recupera tutti i parametri disponibili dei tipi di parametro specificati su cui è possibile eseguire query per un'istanza database specificata.
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	Recupera tutti i report di analisi disponibili per l'istanza database. I report sono elencati in base all'ora di inizio di ciascun report.
ListTagsForResource	aws pi list-tags-for-resource	Elenca tutti i tag dei metadati aggiunti alla risorsa. L'elenco include il nome e il valore del tag.
TagResource	aws pi tag-resource	Aggiunge tag di metadati alla RDS risorsa Amazon. Il tag include un nome e un valore.
UntagResource	aws pi untag-resource	Rimuove tag dei metadati dalla risorsa.

Per ulteriori informazioni sul recupero delle metriche delle serie temporali ed esempi per AWS CLI Performance Insights, consulta i seguenti argomenti.

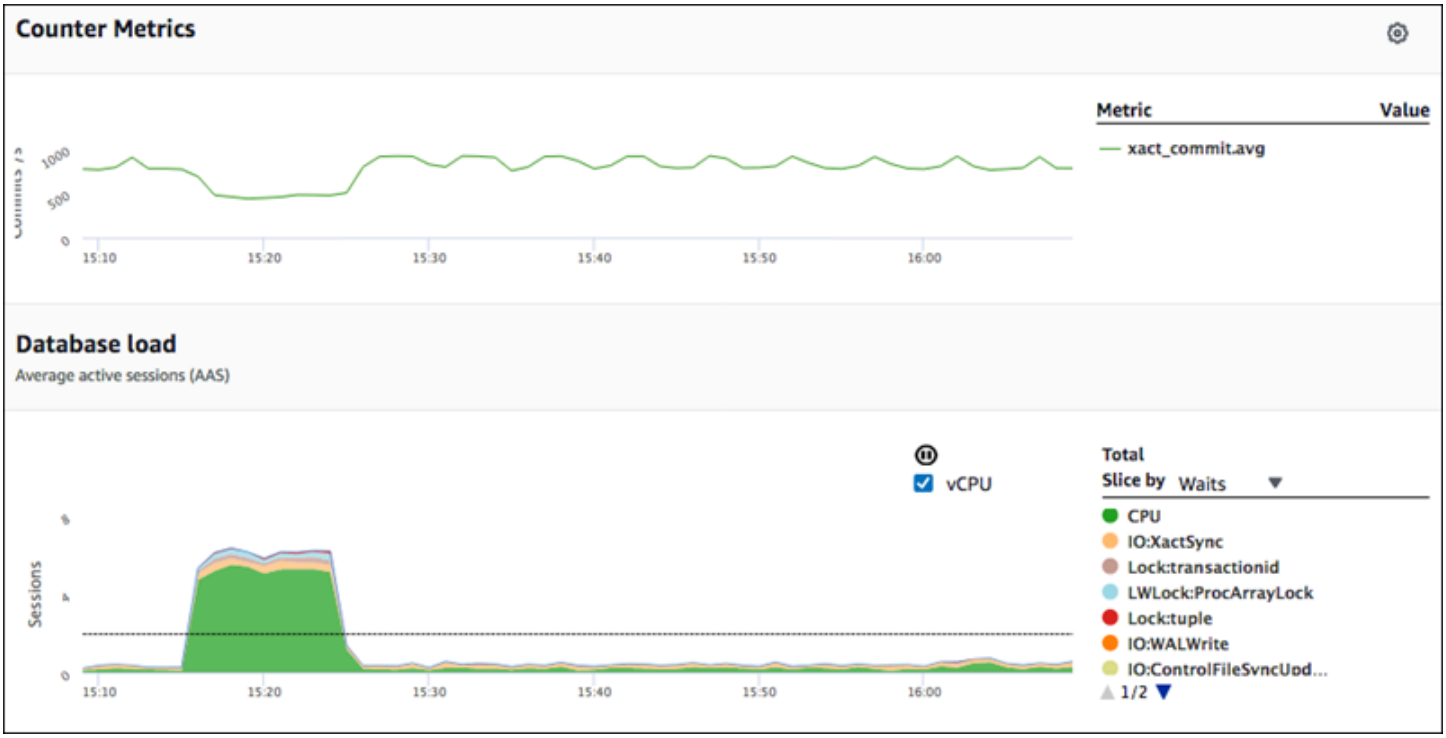
Argomenti

- Recupero delle metriche delle serie temporali per Performance Insights
- AWS CLI esempi di Performance Insights

Recupero delle metriche delle serie temporali per Performance Insights

L'operazione `GetResourceMetrics` recupera uno o più parametri di serie temporali dai dati di Performance Insights. `GetResourceMetrics` richiede un parametro e un periodo di tempo e restituisce una risposta con un elenco di punti di dati.

Ad esempio, gli AWS Management Console usi `GetResourceMetrics` per compilare il grafico Counter Metrics e il grafico Database Load, come illustrato nell'immagine seguente.



Tutti i parametri restituiti da `GetResourceMetrics` sono parametri di serie temporali standard ad eccezione di `db.load`. Questo parametro è visualizzato nel grafico Database Load (Carico del database). Il parametro `db.load` è diverso dagli altri parametri di serie temporali in quanto può essere suddiviso in sottocomponenti detti dimensioni. Nell'immagine precedente, `db.load` è suddiviso e raggruppato in base agli stati delle attese che formano il `db.load`.

> **Note**
>
> `GetResourceMetrics` può anche restituire il parametro `db.sampleload`, ma il parametro `db.load` è appropriato nella maggior parte dei casi.

Per informazioni sui parametri contatore restituiti da `GetResourceMetrics`, consulta Parametri contatore di Performance Insights.

I seguenti calcoli sono supportati per i parametri:

- Media: il valore medio per il parametro su un periodo di tempo. Aggiungi `.avg` al nome parametro.
- Minimo: il valore minimo per il parametro su un periodo di tempo. Aggiungi `.min` al nome parametro.
- Massimo: il valore massimo per il parametro su un periodo di tempo. Aggiungi `.max` al nome parametro.
- Somma: la somma dei valori dei parametri su un periodo di tempo. Aggiungi `.sum` al nome parametro.
- Conteggio di esempio: il numero di volte che il parametro è stato raccolto su un periodo di tempo. Aggiungi `.sample_count` al nome parametro.

Ad esempio, supponiamo che un parametro venga raccolto per 300 secondi (5 minuti) e che il parametro venga raccolto una volta al minuto. I valori per ogni minuto sono 1, 2, 3, 4 e 5. In questo caso. vengono restituiti i seguenti calcoli:

- Media: 3
- Minimo: 1
- Massimo: 5
- Somma: 15
- Conteggio del campione: 5

Per informazioni sull'utilizzo del `get-resource-metrics` AWS CLI comando, vedere. [get-resource-metrics]

Per l'opzione `--metric-queries`, specifica una o più query per cui ottenere risultati. Ciascuna query consiste di un parametro obbligatorio `Metric` e parametri facoltativi `GroupBy` e `Filter`. Di seguito è riportato un esempio della specifica di un'opzione `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
   },
```

```
  "Filter": {"string": "string"
    ...}
```

AWS CLI esempi di Performance Insights

Nelle sezioni seguenti, scopri di più su AWS Command Line Interface (AWS CLI) for Performance Insights ed AWS CLI esempi di utilizzo.

Argomenti

- Guida integrata AWS CLI per Performance Insights
- Recupero dei parametri contatore
- Recupero della media del carico del database per i principali eventi di attesa
- Recupero della media di carico del DB per top SQL
- Recupero della media di carico del DB filtrata per SQL
- Recupero del testo completo di una dichiarazione SQL
- Creazione di un report di analisi delle prestazioni per un periodo di tempo
- Recupero di un report di analisi delle prestazioni
- Elenco di tutti i report di analisi delle prestazioni per l'istanza database
- Eliminazione di un report di analisi delle prestazioni
- Aggiunta di un tag a un report di analisi delle prestazioni
- Elenco di tutti i tag per un report di analisi delle prestazioni
- Eliminazione di tag da un report di analisi delle prestazioni

Guida integrata AWS CLI per Performance Insights

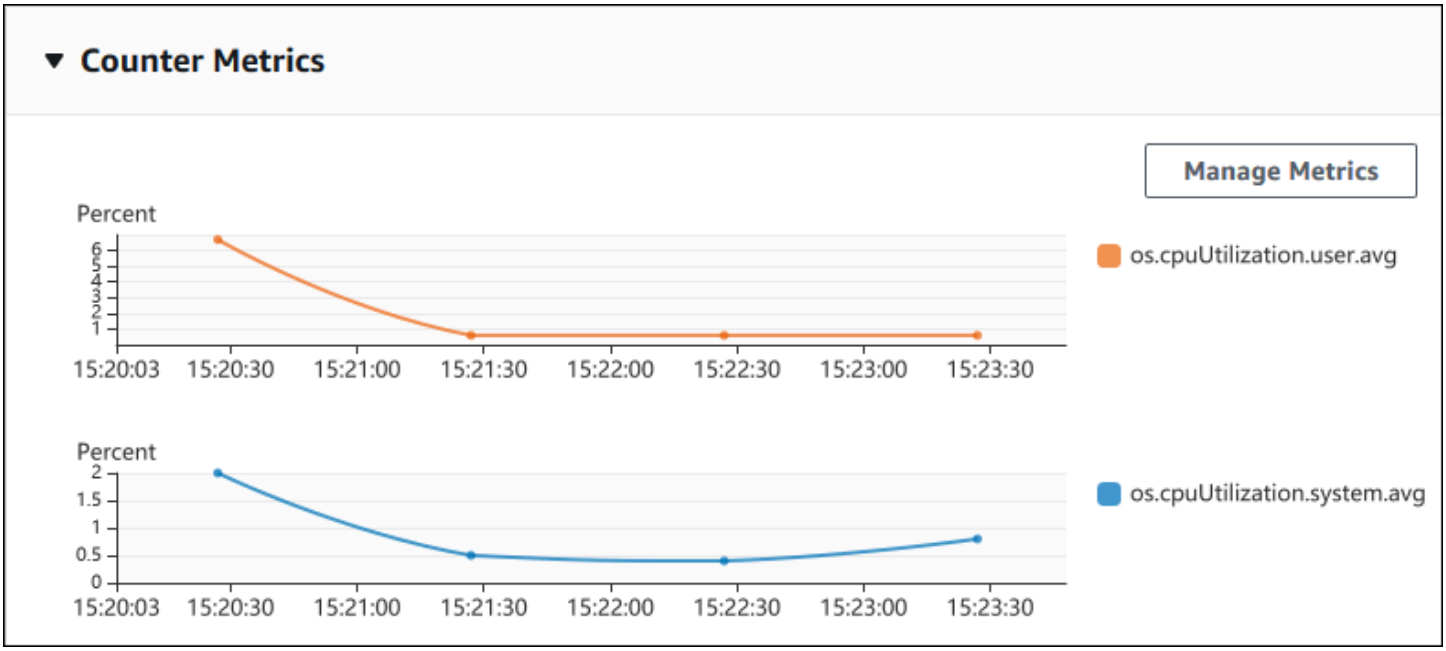
Puoi visualizzare i dati di Performance Insights utilizzando la AWS CLI. È possibile visualizzare la guida per i AWS CLI comandi di Performance Insights immettendo quanto segue nella riga di comando.

```
aws pi help
```

Se non lo hai AWS CLI installato, consulta Installazione di AWS CLI nella Guida per l'AWS CLI utente per informazioni sull'installazione.

Recupero dei parametri contatore

Lo screenshot seguente mostra due grafici dei parametri contatore nella AWS Management Console.



L'esempio seguente mostra come raccogliere gli stessi dati AWS Management Console utilizzati per generare i due grafici contimetrici.

In Linux, macOS, oppure Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

In Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
```

```
                    {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Puoi agevolare la lettura del comando specificando un file per l'opzione `--metrics-query`. Il seguente esempio utilizza un file denominato query.json per l'opzione. Il file presenta i seguenti contenuti.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Esegui il comando seguente per utilizzare il file.

In Linux, macOS, oppure Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

In Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

L'esempio precedente specifica i seguenti valori per le opzioni:

- `--service-type`— RDS per Amazon RDS
- `--identifier` – L'ID risorsa per l'istanza database

- `--start-timee--end-time`: i `DateTime` valori ISO 8601 per il periodo da interrogare, con più formati supportati

Esegue query per un intervallo di tempo di un'ora:

- `--period-in-seconds` – `60` per una query al minuto
- `--metric-queries` – Una serie di due query, ognuna solo per un parametro

  Il nome del parametro utilizza punti per classificare il parametro in una categoria utile, dove l'ultimo elemento è una funzione. Nell'esempio, la funzione è `avg` per ciascuna query. Come per Amazon CloudWatch, le funzioni supportate sono `min`,`max`,`total`, `avg`.

La risposta è simile a quella riportata di seguito.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
```

```
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

La risposta presenta un `Identifier`, un `AlignedStartTime` e un `AlignedEndTime`. Poiché il valore `--period-in-seconds` era `60`, l'ora di inizio e fine è stata allineata al minuto. Se `--period-in-seconds` fosse stato `3600`, l'ora di inizio e fine sarebbe stata allineata all'ora.

`MetricList` nella risposta ha una serie di voci, ciascuna con una voce `Key` e una voce `DataPoints`. Ciascun `DataPoint` ha un `Timestamp` e un `Value`. Ciascun elenco `Datapoints` ha 60 punti di dati in quanto le query sono per dati al minuto nell'arco di un'ora, con `Timestamp1/Minute1`, `Timestamp2/Minute2` e così via, fino a `Timestamp60/Minute60`.

Poiché la query è per due diversi parametri contatore, la risposta contiene due element `MetricList`.

Recupero della media del carico del database per i principali eventi di attesa

L'esempio seguente è la stessa query AWS Management Console utilizzata per generare un grafico a linee ad area impilata. L'esempio recupera `db.load.avg` per l'ultima ora con carico diviso in base ai sette principali eventi di attesa. Il comando è come quello in Recupero dei parametri contatore. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
```

```
]
```

Eseguire il comando riportato qui di seguito.

In Linux, macOS, oppure Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

In Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

L'esempio specifica il parametro di `db.load.avg` e un `GroupBy` dei sette principali eventi di attesa. Per informazioni dettagliate sui valori validi per questo esempio, consulta DimensionGroupil Performance Insights API Reference.

La risposta è simile a quella riportata di seguito.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
```

```
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
} //end of response
```

In questa risposta, ci sono otto voci in `MetricList`. C'è una voce per il `db.load.avg` totale e sette voci ciascuno per il `db.load.avg` suddivise secondo uno dei sette principali eventi di attesa. A differenza del primo esempio, poiché era presente una dimensione di raggruppamento, deve esserci una chiave per ciascun raggruppamento del parametro. Può esserci una sola chiave per ciascun parametro, come nel caso d'uso del parametro contatore di base.

Recupero della media di carico del DB per top SQL

L'esempio seguente raggruppa in `db.wait_events` base alle prime 10 SQL istruzioni. Esistono due diversi gruppi di SQL dichiarazioni:

- `db.sql`— La SQL dichiarazione completa, ad esempio `select * from customers where customer_id = 123`
- `db.sql_tokenized`— La SQL dichiarazione tokenizzata, ad esempio `select * from customers where customer_id = ?`

Quando si analizzano le prestazioni del database, può essere utile considerare come un unico elemento logico SQL le istruzioni che differiscono solo in base ai parametri. Pertanto, puoi utilizzare `db.sql_tokenized` durante le query. Tuttavia, specialmente quando si è interessati a spiegare i piani, a volte è più utile esaminare SQL le istruzioni complete con parametri e raggruppare le query per. `db.sql` Esiste una relazione padre-figlio tra tokenizzato e completoSQL, con più completi SQL (figli) raggruppati sotto lo stesso nome tokenizzato (genitore). SQL

Il comando in questo esempio è simile a quello in Recupero della media del carico del database per i principali eventi di attesa. Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
    }
]
```

Nell'esempio seguente viene utilizzato `db.sql_tokenized`.

In Linux, macOS, oppure Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
```

```
   --start-time 2018-10-29T00:00:00Z \
   --end-time   2018-10-30T00:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

In Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-29T00:00:00Z ^
   --end-time   2018-10-30T00:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

Questo esempio esegue interrogazioni nell'arco di 24 ore, di cui un'ora period-in-seconds.

L'esempio specifica il parametro di `db.load.avg` e un `GroupBy` dei sette principali eventi di attesa. Per informazioni dettagliate sui valori validi per questo esempio, consulta DimensionGroupil Performance Insights API Reference.

La risposta è simile a quella riportata di seguito.

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
                },
                //... 24 datapoints
            ]
        },
        {
```

```
            "Key": { //Next key is the top tokenized SQL
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

Questa risposta contiene 11 voci `MetricList` (1 in totale, 10 più tokenizzateSQL), con 24 voci all'ora per ogni voce. `DataPoints`

Per «tokenized»SQL, ci sono tre voci in ogni elenco di dimensioni:

- `db.sql_tokenized.statement`— La dichiarazione tokenizzataSQL.
- `db.sql_tokenized.db_id` — L'ID del database nativo usato per fare riferimento a o un ID sintetico che Performance Insights genera per te se l'ID del database nativo non è disponibile. SQL Questo esempio restituisce l'ID sintetico `pi-2372568224`.
- `db.sql_tokenized.id` – L'ID della query all'interno di Performance Insights.

  Nel AWS Management Console, questo ID è denominato Support ID. Si chiama così perché l'ID è costituito da dati che AWS Support può esaminare per aiutarti a risolvere un problema con il tuo database. AWS prende molto sul serio la sicurezza e la privacy dei dati e quasi tutti i dati vengono archiviati crittografati con la AWS KMS chiave dell'utente. Pertanto, nessuno all'interno AWS può guardare questi dati. Nell'esempio precedente, sia `tokenized.statement` che `tokenized.db_id` vengono archiviati crittografati. Se hai un problema con il tuo database, AWS Support può aiutarti facendo riferimento al Support ID.

Quando si eseguo query, potrebbe essere utile specificare un `Group` in `GroupBy`. Tuttavia, per un controllo più dettagliato dei dati restituiti, occorre specificare l'elenco delle dimensioni. Ad esempio, se tutto ciò di cui si necessita è `db.sql_tokenized.statement`, è possibile aggiungere l'attributo `Dimensions` al file query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Recupero della media di carico del DB filtrata per SQL

L'immagine precedente mostra che è stata selezionata una particolare query e che il grafico a linee ad area in pila con sessioni attive della media in alto è definito in base a tale query. Sebbene la query sia ancora per i sette principali eventi di attesa complessivi, il valore della risposta è filtrato. Il filtro fa sì che vengano prese in considerazione solo le sessioni che corrispondono al filtro specifico.

La API query corrispondente in questo esempio è simile al comando in[Recupero della media di carico del DB per top SQL](). Tuttavia, il file query.json presenta i seguenti contenuti.

```
[
```

```
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

In Linux, macOS, oppure Unix:

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

In Windows:

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

La risposta è simile a quella riportata di seguito.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
```

```
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
```

```
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
```

```
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

In questa risposta, tutti i valori vengono filtrati in base al contributo di tokenized SQL AKIAIOSFODNN7EXAMPLE specificato nel file query.json. Le chiavi potrebbero inoltre seguire un ordine diverso rispetto a una query senza filtro, poiché sono i primi cinque eventi di attesa che hanno influito sul filtrato. SQL

Recupero del testo completo di una dichiarazione SQL

L'esempio seguente recupera il testo completo di un'SQListruzione per un'istanza DB. `db-10BCD2EFGHIJ3KL4M5NO6PQRS5` `--group` è `db.sql`, e `--group-identifier` è `db.sql.id`. In questo esempio, *`my-sql-id`* rappresenta un SQL ID recuperato richiamando o. `pi get-resource-metrics` `pi describe-dimension-keys`

Esegui il comando seguente.

In Linux, macOS, oppure Unix:

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
   --group db.sql \
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

In Windows:

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
   --requested-dimensions statement
```

In questo esempio, sono disponibili i dettagli delle dimensioni. Pertanto, Performance Insights recupera il testo completo dell'SQListruzione, senza troncarlo.

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

Creazione di un report di analisi delle prestazioni per un periodo di tempo

L'esempio seguente crea un report di analisi delle prestazioni con l'ora di inizio `1682969503` e l'ora di fine `1682979503` per il database `db-loadtest-0`.

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

La risposta è l'identificatore univoco `report-0234d3ed98e28fb17` per il report.

```
{
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

Recupero di un report di analisi delle prestazioni

L'esempio seguente recupera i dettagli del report di analisi per il report `report-0d99cc91c4422ee61`.

```
aws pi get-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

La risposta fornisce lo stato del rapporto, l'ID, i dettagli temporali e gli approfondimenti.

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

Elenco di tutti i report di analisi delle prestazioni per l'istanza database

L'esempio seguente elenca tutti i report di analisi delle prestazioni disponibili per il database `db-loadtest-0`.

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
--region us-west-2
```

La risposta elenca tutti i report con i dettagli relativi all'ID, allo stato e al periodo di tempo del report.

```
{
```

```
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
            "StartTime": 1680583486.584,
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```

Eliminazione di un report di analisi delle prestazioni

L'esempio seguente elimina il report di analisi per il database `db-loadtest-0`.

```
aws pi delete-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

Aggiunta di un tag a un report di analisi delle prestazioni

L'esempio seguente aggiunge un tag con una chiave `name` e un valore `test-tag` al report `report-01ad15f9b88bcbd56`.

```
aws pi tag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

Elenco di tutti i tag per un report di analisi delle prestazioni

Nell'esempio seguente vengono elencati tutti i tag per il report `report-01ad15f9b88bcbd56`.

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

La risposta elenca il valore e la chiave per tutti i tag aggiunti al report:

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

Eliminazione di tag da un report di analisi delle prestazioni

Nell'esempio seguente viene eliminato il tag `name` dal report `report-01ad15f9b88bcbd56`.

```
aws pi untag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
```

```
--tag-keys name \
--region us-west-2
```

Dopo l'eliminazione del tag, la chiamata a `list-tags-for-resource` API non elenca questo tag.

Registrazione delle chiamate Performance Insights utilizzando AWS CloudTrail

Performance Insights funziona con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o AWS servizio in Performance Insights. CloudTrailacquisisce tutte le API chiamate a Performance Insights come eventi. Questa acquisizione include le chiamate dalla RDS console Amazon e le chiamate in codice alle API operazioni di Performance Insights.

Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Performance Insights. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando i dati raccolti da CloudTrail, puoi determinare determinate informazioni. Queste informazioni includono la richiesta effettuata a Performance Insights, l'indirizzo IP da cui è stata eseguita la richiesta, l'autore della richiesta e il momento in cui è stata eseguita. Include anche dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

Utilizzo delle informazioni di Performance Insights in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Performance Insights, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella CloudTrail console nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nel tuo AWS account. Per ulteriori informazioni, consulta Visualizzazione degli eventi con cronologia degli CloudTrail eventi nella Guida AWS CloudTrail per l'utente.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Performance Insights, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutte le AWS regioni. Il trail registra gli eventi di tutte le AWS regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS CloudTrail :

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione di Amazon SNS Notifications per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le operazioni di Performance Insights vengono registrate CloudTrail e documentate nel Performance Insights API Reference. Ad esempio, le chiamate alle `GetResourceMetrics` operazioni `DescribeDimensionKeys` and generano voci nei file di CloudTrail registro.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta l'CloudTrail userIdentityelemento.

Voci del file di registro Performance Insights

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta da un'origine. Ogni evento include informazioni sull'operazione richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. CloudTraili file di registro non sono una traccia stack ordinata delle API chiamate pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che dimostra l'`GetResourceMetrics`operazione.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
```

```
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Performance Insights API ed VPC endpoint di interfaccia ()AWS PrivateLink

Puoi utilizzarlo AWS PrivateLink per creare una connessione privata tra te VPC e Amazon RDS Performance Insights. Puoi accedere a Performance Insights come se fosse sul tuo computerVPC, senza utilizzare un gateway, un NAT dispositivo, una VPN connessione o una AWS Direct Connect connessione Internet. Le istanze del tuo VPC non hanno bisogno di indirizzi IP pubblici per accedere a Performance Insights.

Stabilisci questa connessione privata creando un endpoint di interfaccia attivato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di

interfaccia. Si tratta di interfacce di rete gestite dai richiedenti che fungono da punto di ingresso per il traffico destinato a Performance Insights.

Per ulteriori informazioni, consulta Access Servizi AWS through AWS PrivateLink nella Guida.AWS PrivateLink

Considerazioni relative a Performance Insights

Prima di configurare un endpoint di interfaccia per Performance Insights, consulta le considerazioni nella AWS PrivateLink Guida.

Performance Insights supporta l'esecuzione di chiamate a tutte API le sue azioni tramite l'endpoint dell'interfaccia.

Per impostazione predefinita, l'accesso completo a Performance Insights è consentito tramite l'endpoint dell'interfaccia. Per controllare il traffico verso Performance Insights attraverso l'endpoint dell'interfaccia, associa un gruppo di sicurezza alle interfacce di rete degli endpoint.

Disponibilità

Performance Insights API attualmente supporta VPC gli endpoint Regioni AWS che supportano Performance Insights. Per informazioni sulla disponibilità di Performance Insights, vedere Regioni supportate e motori Aurora DB per Performance Insights.

Crea un endpoint di interfaccia per Performance Insights

Puoi creare un endpoint di interfaccia per Performance Insights utilizzando la VPC console Amazon o AWS Command Line Interface (AWS CLI). Per ulteriori informazioni, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di AWS PrivateLink .

Crea un endpoint di interfaccia per Performance Insights utilizzando il seguente nome di servizio:

Se abiliti private DNS per l'endpoint dell'interfaccia, puoi effettuare API richieste a Performance Insights utilizzando il DNS nome regionale predefinito. Ad esempio `pi.us-east-1.amazonaws.com`.

Creazione di una policy VPC sugli endpoint per Performance Insights API

Una policy per gli endpoint è una IAM risorsa che è possibile collegare a un endpoint di interfaccia. La policy predefinita per gli endpoint consente l'accesso completo a Performance Insights tramite

l'endpoint dell'interfaccia. Per controllare l'accesso consentito a Performance Insights dal tuoVPC, allega una policy personalizzata per gli endpoint all'endpoint di interfaccia.

Una policy di endpoint specifica le informazioni riportate di seguito:

- I principali che possono eseguire azioni (IAMutenti Account AWS e IAM ruoli).
- Le azioni che possono essere eseguite.
- Le risorse in cui è possibile eseguire le operazioni.

Per ulteriori informazioni, consulta la sezione Controllo dell'accesso ai servizi con policy di endpoint nella Guida di AWS PrivateLink .

Esempio: policy sugli VPC endpoint per le azioni di Performance Insights

Di seguito è riportato l'esempio di una policy dell'endpoint personalizzata. Quando alleghi questa policy all'endpoint dell'interfaccia, concede l'accesso alle azioni Performance Insights elencate per tutti i principali su tutte le risorse.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreatePerformanceAnalysisReport",
            "rds:DeletePerformanceAnalysisReport",
            "rds:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

Esempio: policy VPC sugli endpoint che nega tutti gli accessi da un account specifico AWS

La seguente policy sugli VPC endpoint nega all' AWS account `123456789012` tutti gli accessi alle risorse che utilizzano l'endpoint. La policy consente tutte le operazioni da altri account.

```
{
  "Statement": [
```

```
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Indirizzamento IP per Performance Insights

Gli indirizzi IP consentono VPC alle risorse dell'utente di comunicare tra loro e con le risorse su Internet. Performance Insights supporta entrambi IPv4 i protocolli di IPv6 indirizzamento. Per impostazione predefinita, Performance Insights e Amazon VPC utilizzano il protocollo di IPv4 indirizzamento. Non puoi disattivare questo comportamento. Quando crei unVPC, assicurati di specificare un IPv4 CIDR blocco (un intervallo di IPv4 indirizzi privati).

Facoltativamente, puoi assegnare un IPv6 CIDR blocco alle tue VPC e alle sottoreti e assegnare IPv6 gli indirizzi di quel blocco alle RDS risorse della tua sottorete. Il supporto per il IPv6 protocollo amplia il numero di indirizzi IP supportati. Utilizzando il IPv6 protocollo, ti assicuri di disporre di indirizzi disponibili sufficienti per la crescita futura di Internet. Le RDS risorse nuove ed esistenti possono utilizzare IPv4 IPv6 gli indirizzi all'interno del tuoVPC. La configurazione, la protezione e la traduzione del traffico di rete tra i due protocolli utilizzati in diverse parti di un'applicazione può causare sovraccarico operativo. Puoi standardizzare il IPv6 protocollo per le RDS risorse Amazon per semplificare la configurazione della rete. Per ulteriori informazioni sugli endpoint e le quote del servizio, consulta Endpoint e quote di Amazon Relational Database Service.

Analisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS

Amazon DevOps Guru è un servizio operativo completamente gestito che aiuta sviluppatori e operatori a migliorare le prestazioni e la disponibilità delle loro applicazioni. DevOpsGuru delega le attività associate all'identificazione dei problemi operativi in modo da poter implementare rapidamente i consigli per migliorare la tua applicazione. Per ulteriori informazioni, consulta Cos'è Amazon DevOps Guru? nella Guida per l'utente di Amazon DevOps Guru.

DevOpsGuru rileva, analizza e fornisce raccomandazioni per i problemi operativi esistenti per tutti i motori Amazon RDS DB. DevOpsGuru for RDS estende questa funzionalità applicando l'apprendimento automatico alle metriche di Performance Insights per i database Amazon for Postgre. SQL Queste funzionalità di monitoraggio consentono a DevOps Guru di rilevare e diagnosticare i RDS rallentamenti delle prestazioni e consigliare azioni correttive specifiche. DevOpsGuru for RDS può anche rilevare condizioni problematiche nei database Aurora RDS per il database SQL prima che si verifichino.

Ora puoi visualizzare questi consigli nella console. RDS Per ulteriori informazioni, consulta .

> **Important**
>
> Amazon DevOps Guru non è disponibile per Aurora Serverless Cluster DB.

Il video seguente è una panoramica di DevOps Guru for. RDS

Per un approfondimento su questo argomento, consulta Amazon DevOps Guru for RDS under the hood.

Argomenti

- Vantaggi di DevOps Guru per RDS
- Come funziona DevOps Guru for RDS
- Configurare Guru per DevOps RDS

Vantaggi di DevOps Guru per RDS

Se sei responsabile di un SQL database Amazon Aurora RDS , potresti non sapere che si sta verificando un evento o una regressione che sta interessando quel database. Quando scopri il problema, potresti non sapere perché si sta verificando o cosa fare al riguardo. Invece di rivolgerti a un amministratore del database (DBA) per ricevere assistenza o affidarti a strumenti di terze parti, puoi seguire i consigli di Guru for. DevOps RDS

Dall'analisi dettagliata di DevOps Guru si ottengono i seguenti vantaggi per: RDS

Diagnosi rapida

DevOpsGuru for monitora e analizza RDS continuamente la telemetria del database. DevOpsGuru for RDS utilizza tecniche statistiche e di apprendimento automatico per estrarre questi dati e rilevare anomalie. Per ulteriori informazioni sui dati di telemetria, consulta Monitoraggio del carico del DB con Performance Insights su Amazon Aurora e Monitoring dei parametri del sistema operativo con Enhanced Monitoring nella Amazon Aurora User Guide Monitoring Monitoring del carico DB . RDS

Risoluzione rapida

Ogni anomalia identifica il problema delle prestazioni e suggerisce strade di indagine o azioni correttive. Ad esempio, DevOps Guru for RDS potrebbe consigliarti di esaminare specifici eventi di attesa. In alternativa, è consigliabile regolare le impostazioni del pool di applicazioni per limitare il numero di connessioni al database. Sulla base di questi consigli, è possibile risolvere i problemi di prestazioni più rapidamente rispetto alla risoluzione manuale dei problemi.

Approfondimenti proattivi

DevOpsGuru for RDS utilizza le metriche delle tue risorse per rilevare comportamenti potenzialmente problematici prima che diventino un problema più grave. Ad esempio, è in grado di rilevare quando il database utilizza un numero crescente di tabelle temporanee su disco, ovvero una situazione che potrebbe pregiudicare le prestazioni. DevOpsGuru fornisce quindi consigli per aiutarvi a risolvere i problemi prima che diventino problemi più gravi.

Conoscenza approfondita dei tecnici e del machine learning di Amazon

Per rilevare problemi di prestazioni e aiutarti a risolvere i problemi, DevOps Guru for RDS si affida all'apprendimento automatico (ML) e a formule matematiche avanzate. Gli ingegneri di database di Amazon hanno contribuito allo sviluppo di DevOps Guru for RDS findings, che racchiude molti anni di gestione di centinaia di migliaia di database. Attingendo a questa conoscenza collettiva, DevOps Guru for RDS può insegnarti le migliori pratiche.

Come funziona DevOps Guru for RDS

DevOpsGuru for RDS raccoglie dati sui tuoi database Aurora RDS da SQL Amazon Performance Insights. RDS La metrica più importante è. `DBLoad` DevOpsGuru for RDS utilizza le metriche di Performance Insights, le analizza con l'apprendimento automatico e pubblica approfondimenti sulla dashboard.

Un'analisi è una raccolta di anomalie correlate rilevate da Guru. DevOps

Approfondimenti proattivi

Un approfondimento proattivo consente di individuare i comportamenti problematici prima che si verifichino. Contiene anomalie con consigli e parametri correlati per aiutarti a risolvere i problemi nei database Amazon Aurora RDS prima che diventino problemi più gravi. Questi approfondimenti sono pubblicati nella dashboard Guru. DevOps

Ad esempio, DevOps Guru potrebbe rilevare che il database Aurora SQL Postgre sta creando molte tabelle temporanee su disco. Se non affrontata per tempo, questa tendenza può causare problemi di prestazioni. Ogni approfondimento proattivo include i suggerimenti per i comportamenti correttivi e i collegamenti ad argomenti pertinenti in Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru o Ottimizzazione con informazioni proattive su Amazon SQL Guru DevOps. Per ulteriori informazioni, consulta Working with Insights in DevOps Guru nella Amazon DevOps Guru User Guide.

Approfondimenti reattivi

Un approfondimento reattivo identifica un comportamento anomalo nel momento in cui si verifica. Se DevOps Guru for RDS rileva problemi di prestazioni nelle tue istanze DB di Amazon RDS Postgre, pubblica una visione reattiva nella dashboard di Guru. DevOps Per ulteriori informazioni, consulta Working with Insights in DevOps Guru nella Amazon DevOps Guru User Guide.

Anomalie causali

Un'anomalia causale è un'anomalia di livello superiore all'interno di un approfondimento reattivo. Il caricamento del database (caricamento del DB) è l'anomalia causale di Guru for. DevOps RDS

Un'anomalia misura l'impatto sulle prestazioni assegnando un livello di gravità di Elevato,Medio, oppureBasso. Per ulteriori informazioni, consulta Key concepts for DevOps Guru RDS nella Amazon DevOps Guru User Guide.

Se DevOps Guru rileva un'anomalia corrente sulla tua istanza DB, verrai avvisato nella pagina Databases della console. RDS La console ti avvisa anche delle anomalie che si sono verificate nelle ultime 24 ore. Per accedere alla pagina delle anomalie dalla RDS console, scegli il link nel messaggio di avviso. La RDS console ti avvisa anche nella pagina del tuo .

Anomalie contestuali

Un'anomalia contestuale è un risultato del carico del database correlato a un approfondimento reattivo. Ogni anomalia contestuale descrive uno specifico problema di prestazioni di Amazon RDS Aurora Postgre che richiede un'analisi. Ad esempio, DevOps Guru for RDS potrebbe consigliarti di prendere in considerazione l'aumento CPU della capacità o di analizzare gli eventi di attesa che contribuiscono al carico del DB.

> **Important**
>
> È consigliabile testare eventuali modifiche in un'istanza di test prima di modificare un'istanza di produzione. In questo modo, capisci l'impatto del cambiamento.

Per ulteriori informazioni, consulta la sezione Analyzing anomalies in Amazon nella Amazon RDS DevOps Guru User Guide.

Configurare Guru per DevOps RDS

Per consentire a DevOps Guru for Amazon RDS di pubblicare approfondimenti per un database Amazon Postgre, completa le seguenti attività.

Argomenti

- Configurazione delle politiche di accesso per Guru per IAM DevOps RDS
- Attivazione di Performance Insights per le istanze DB Aurora RDS
- Attivare DevOps Guru e specificare la copertura delle risorse

Configurazione delle politiche di accesso per Guru per IAM DevOps RDS

Per visualizzare gli avvisi di DevOps Guru nella console, RDS AWS Identity and Access Management (IAM) l'utente o il ruolo devono avere una delle seguenti politiche:

- Il AWS politica gestita `AmazonDevOpsGuruConsoleFullAccess`

- Il AWS politica gestita `AmazonDevOpsGuruConsoleReadOnlyAccess` e una delle seguenti politiche:
  - Il AWS politica gestita `AmazonRDSFullAccess`
  - Una policy gestita dal cliente che include `pi:GetResourceMetrics` e `pi:DescribeDimensionKeys`

Per ulteriori informazioni, consulta Configurazione delle policy di accesso per Performance Insights.

Attivazione di Performance Insights per le istanze DB Aurora RDS

DevOpsGuru for RDS si affida a Performance Insights per i suoi dati. Senza Performance Insights, DevOps Guru pubblica le anomalie, ma non include analisi e raccomandazioni dettagliate.

Quando crei un cluster database Aurora o modifichi un'istanza cluster, puoi attivare Performance Insights. Per ulteriori informazioni, consulta Attivazione e disattivazione di Performance Insights per Aurora.

Attivare DevOps Guru e specificare la copertura delle risorse

Puoi attivare DevOps Guru per fargli monitorare i tuoi database Amazon RDS Postgre in uno dei seguenti modi.

Argomenti

- Attivazione di Guru nella console DevOps RDS
- Aggiungere SQL risorse Aurora RDS nella console Guru DevOps
- Aggiungere risorse Aurora RDS utilizzando AWS CloudFormation

Attivazione di Guru nella console DevOps RDS

Puoi seguire più percorsi nella RDS console Amazon per attivare DevOps Guru.

Argomenti

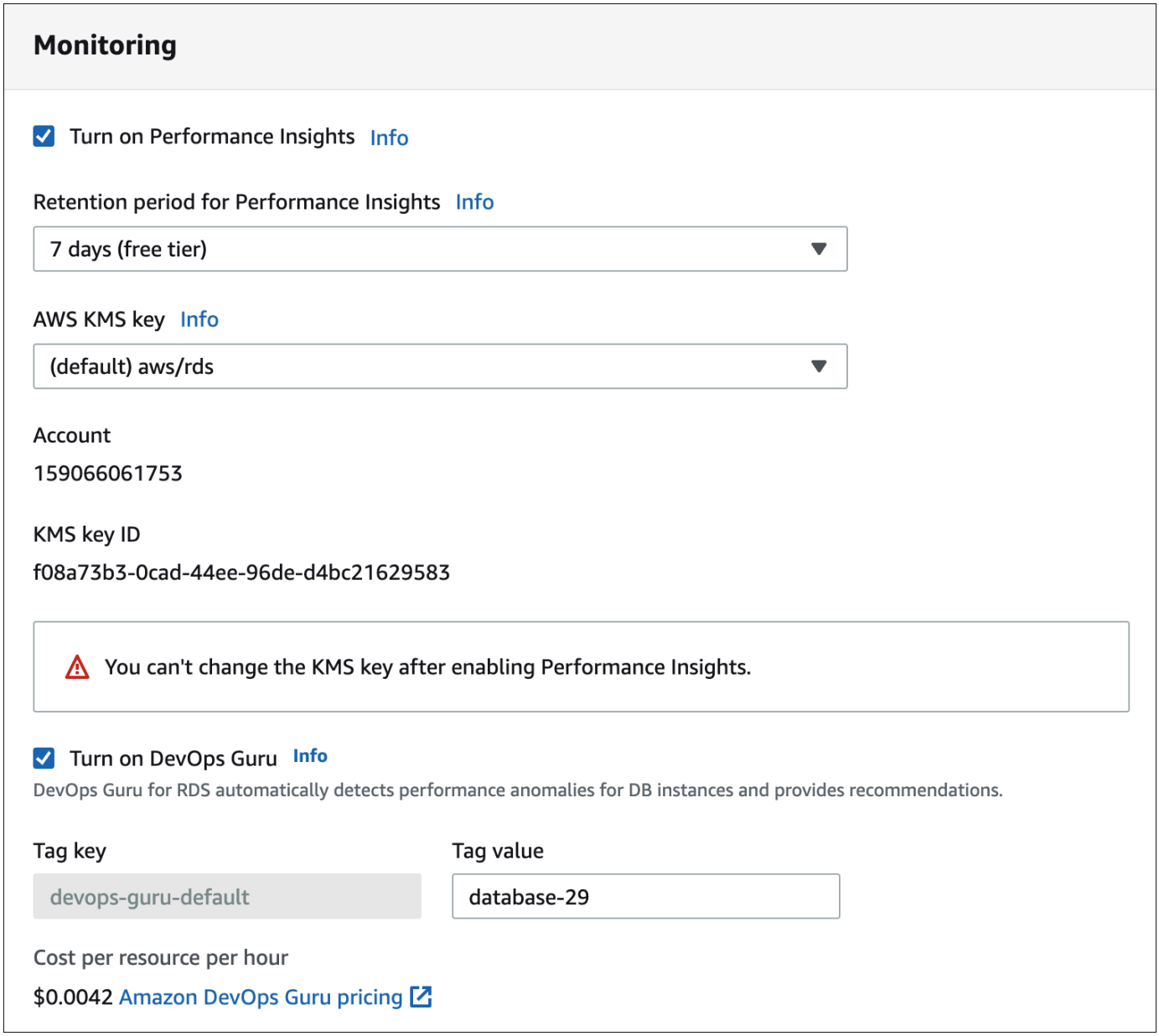
- Attivazione di DevOps Guru quando si crea un database per Postgre SQL
- Attivazione di DevOps Guru dal banner di notifica
- Risposta a un errore di autorizzazione quando attivi Guru DevOps

Attivazione di DevOps Guru quando si crea un database per Postgre SQL

Il flusso di lavoro di creazione include un'impostazione che attiva la copertura DevOps Guru per il database. Questa impostazione è abilitata per default quando scegli il modello Production (Produzione).

Per attivare DevOps Guru quando si crea un database per Postgre SQL

1. Accedi al AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Segui la procedura riportata in Creazione di un cluster di database, fino al passaggio, senza includerlo, in cui scegli le impostazioni di monitoraggio.
3. In Monitoring (Monitoraggio), scegli Turn on Performance Insights (Attiva Performance Insights). DevOpsAffinché Guru for RDS fornisca un'analisi dettagliata delle anomalie delle prestazioni, è necessario attivare Performance Insights.
4. Scegli Attiva Guru. DevOps



5. Crea un tag per il tuo database in modo che DevOps Guru possa monitorarlo. Esegui questa operazione:

   - Nel campo di testo per Tag key (Chiave tag), inserisci un nome che inizi con **`Devops-Guru-`**.
   - Nel campo di testo per Tag value (Valore tag), inserisci qualsiasi valore. Ad esempio, se si immette **`rds-database-1`** il nome del SQL database Aurora RDS , è possibile immettere anche **`rds-database-1`** come valore del tag.

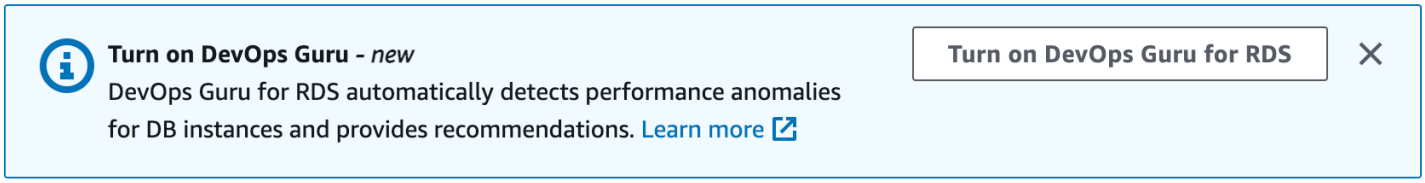
   Per ulteriori informazioni sui tag, consulta "Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru" nella Amazon DevOps Guru User Guide.

6. Completare le fasi restanti in Creazione di un cluster di database.

Attivazione di DevOps Guru dal banner di notifica

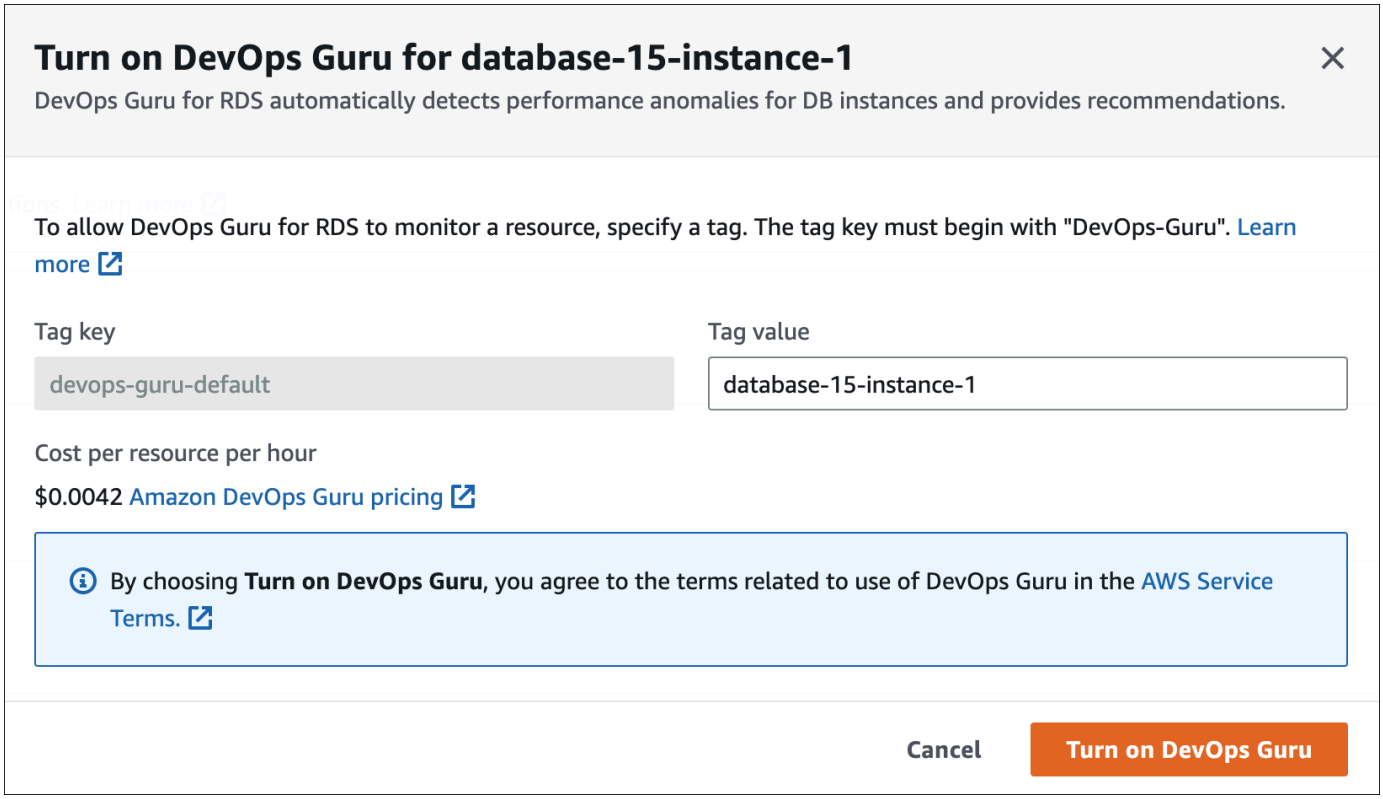
Se le tue risorse non sono coperte da DevOps Guru, Amazon ti RDS avvisa con un banner nelle seguenti posizioni:

- La scheda Monitoring (Monitoraggio) di un'istanza cluster database
- Pannello di controllo di Performance Insights



Per attivare DevOps Guru per il tuo database per Postgre SQL

1. Nel banner, scegli Turn on Guru per. DevOps RDS
2. Immetti un nome e un valore della chiave tag. Per ulteriori informazioni sui tag, consulta "Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru" nella Amazon DevOps Guru User Guide.



3. Scegli Attiva DevOps Guru.

Risposta a un errore di autorizzazione quando attivi Guru DevOps

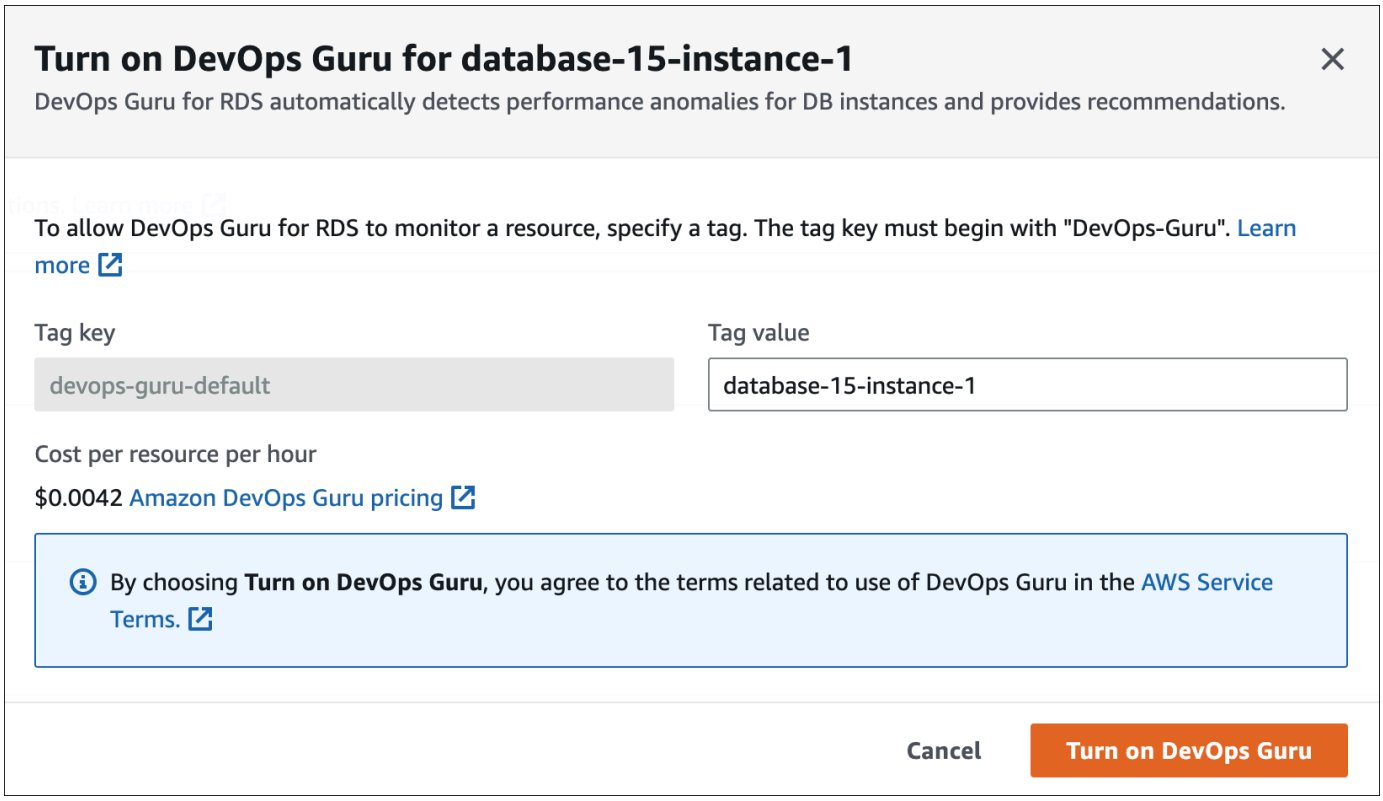
Se attivi DevOps Guru dalla RDS console quando crei un database, RDS potrebbe essere visualizzato il seguente banner sulle autorizzazioni mancanti.

Rispondere a un errore di autorizzazioni

1. Assegna al tuo IAM utente o ruolo il ruolo `AmazonDevOpsGuruConsoleFullAccess` gestito dagli utenti. Per ulteriori informazioni, consulta Configurazione delle politiche di accesso per Guru per IAM DevOps RDS.
2. Apri la RDS console.
3. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
4. Scegli un'istanza database nel cluster appena creato.
5. Scegli l'interruttore per cui attivare DevOpsGuru. RDS

DevOps Guru for RDS

6. Scegli un valore di tag. Per ulteriori informazioni, consulta "Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru" nella Amazon DevOps Guru User Guide.



7. Scegli Attiva DevOps Guru.

Aggiungere SQL risorse Aurora RDS nella console Guru DevOps

Puoi specificare la copertura delle risorse DevOps Guru sulla console Guru. DevOps Segui la procedura descritta in Specificare la copertura delle risorse DevOps Guru nella Amazon DevOps Guru User Guide. Quando modifichi le risorse analizzate, scegli una delle opzioni seguenti:

- Scegli Tutte le risorse dell'account per analizzare tutte le risorse supportate, inclusi i database Aurora RDS , nel tuo Account AWS e regione.
- Scegli CloudFormation gli stack per analizzare i database RDSAurora Postgre che si trovano negli stack che scegli. Per ulteriori informazioni, consulta Uso AWS CloudFormation pile per identificare le risorse nelle tue applicazioni DevOps Guru nella Amazon Guru DevOps User Guide.
- Scegli Tag per analizzare i SQL database Aurora RDS che hai taggato. Per ulteriori informazioni, consulta Usa i tag per identificare le risorse nelle tue applicazioni DevOps Guru nella Amazon DevOps Guru User Guide.

Per ulteriori informazioni, consulta Enable DevOps Guru nella Amazon DevOps Guru User Guide.

Aggiungere risorse Aurora RDS utilizzando AWS CloudFormation

Puoi usare i tag per aggiungere la copertura delle tue SQL risorse Aurora RDS ai tuoi modelli. CloudFormation La procedura seguente presuppone che tu disponga di un CloudFormation modello sia per l'istanza DB RDSAurora lo stack Guru. DevOps

Per specificare un'istanza SQL DB Aurora RDS tag CloudFormation

1. Nel CloudFormation modello per la tua istanza DB, definisci un tag utilizzando una coppia chiave/valore.

   L'esempio seguente assegna il valore `my-aurora-db-instance1` a `Devops-guru-cfn-default` per un'istanza database Aurora.

   ```
   MyAuroraDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBClusterIdentifier: my-aurora-db-cluster
       DBInstanceIdentifier: my-aurora-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-aurora-db-instance1
   ```

2. Nel CloudFormation modello per il tuo stack DevOps Guru, specifica lo stesso tag nel filtro di raccolta delle risorse.

   L'esempio seguente configura DevOps Guru per fornire una copertura alla risorsa con il valore del tag. `my-aurora-db-instance1`

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
         Tags:
           - AppBoundaryKey: "Devops-guru-cfn-default"
             TagValues:
             - "devopsguru-my-aurora-db-instance1"
   ```

   Nell'esempio seguente si fornisce la copertura per tutte le risorse all'interno di `Devops-guru-cfn-default` del limite dell'applicazione.

```
DevOpsGuruResourceCollection:
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

Per ulteriori informazioni, vedere AWS::DevOpsGuru: ResourceCollection e: AWS::RDS: DBInstance in AWS CloudFormation Guida per l'utente.

Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato

Con il monitoraggio avanzato, potete monitorare il sistema operativo del vostro database in tempo reale. I parametri di monitoraggio avanzato sono utili quando si desidera vedere come viene utilizzata la CPU in un'istanza database dai diversi processi o thread.

Argomenti

- Panoramica sul monitoraggio avanzato
- Configurare e abilitare il monitoraggio avanzato
- Visualizzazione delle metriche del sistema operativo nella console RDS
- Visualizzazione delle metriche del sistema operativo tramite i registri CloudWatch

Panoramica sul monitoraggio avanzato

Amazon RDS fornisce parametri in tempo reale per il sistema operativo su cui è in esecuzione l'istanza di database. È possibile visualizzare tutti i parametri di sistema e le informazioni sui processi per le istanze del database RDS sulla console. È possibile gestire quali parametri si desidera monitorare per ogni istanza e personalizzare il pannello di controllo in base alle proprie esigenze. Per le descrizioni dei parametri di monitoraggio avanzato, consulta Metriche OS nel monitoraggio avanzato.

RDS fornisce i parametri di Enhanced Monitoring al tuo account Amazon CloudWatch Logs. Puoi creare filtri per le metriche CloudWatch da CloudWatch Logs e visualizzare i grafici sulla dashboard. CloudWatch Puoi utilizzare l'output JSON di Enhanced Monitoring di CloudWatch Logs in un sistema di monitoraggio a tua scelta. Per ulteriori informazioni, consulta Enhanced Monitoring in Amazon RDS. FAQs

Argomenti

- Differenze tra le metriche di CloudWatch monitoraggio avanzato
- Conservazione delle metriche di monitoraggio avanzato
- Costo di Enhanced Monitoring (monitoraggio avanzato)

Differenze tra le metriche di CloudWatch monitoraggio avanzato

Un hypervisor crea ed esegue macchine virtuali (). VMs Utilizzando un hypervisor, un'istanza può supportare più guest VMs condividendo virtualmente memoria e CPU. CloudWatch raccoglie le metriche sull'utilizzo della CPU dall'hypervisor per un'istanza DB. Al contrario, Enhanced Monitoring raccoglie le metriche da un agente nell'istanza DB.

È possibile riscontrare differenze tra le misurazioni di Enhanced Monitoring CloudWatch e quelle di Enhanced Monitoring, poiché il livello dell'hypervisor esegue una piccola quantità di lavoro. Le differenze possono essere maggiori se le istanze DB utilizzano classi di istanza più piccole. In questo scenario, più macchine virtuali (VMs) sono probabilmente gestite dal livello di hypervisor su una singola istanza fisica.

Per le descrizioni dei parametri di monitoraggio avanzato, consulta Metriche OS nel monitoraggio avanzato. Per ulteriori informazioni sui CloudWatch parametri, consulta la Amazon CloudWatch User Guide.

Conservazione delle metriche di monitoraggio avanzato

Per impostazione predefinita, i parametri di Enhanced Monitoring vengono archiviati per 30 giorni nei CloudWatch log. Questo periodo di conservazione è diverso dalle metriche tipiche CloudWatch .

Per modificare la quantità di tempo in cui le metriche vengono archiviate nei CloudWatch log, modifica la conservazione per il gruppo di `RDSOSMetrics` log nella console. CloudWatch Per ulteriori informazioni, consulta Change log data retention in CloudWatch logs nella Amazon CloudWatch Logs User Guide.

Costo di Enhanced Monitoring (monitoraggio avanzato)

I parametri di monitoraggio avanzato vengono archiviati nei CloudWatch log anziché nelle metriche. CloudWatch Il costo del monitoraggio avanzato dipende dai seguenti fattori:

- L'Enhanced Monitoring ti verrà addebitato solo se superi il livello gratuito fornito da Amazon CloudWatch Logs. I costi si basano sulle CloudWatch tariffe di archiviazione e trasferimento dei dati dei log.
- La quantità di informazioni trasferite per un'istanza RDS è direttamente proporzionale alla granularità definita per la funzione di monitoraggio avanzato. Un intervallo di monitoraggio più piccolo comporta report più frequenti sui parametri del sistema operativo e aumenta i costi di monitoraggio. Per gestire i costi, imposta granularità diverse per istanze diverse nei tuoi account.

- I costi di utilizzo per il Monitoraggio avanzato vengono applicati a ciascuna istanza database per cui il monitoraggio avanzato è abilitato. Il monitoraggio di un numero elevato di istanze DB è più costoso rispetto al monitoraggio solo di pochi.
- Le istanze database che supportano un carico di lavoro ad alta intensità di elaborazione hanno più attività di processo del sistema operativo per generare report e costi più elevati per il monitoraggio avanzato.

Per ulteriori informazioni sui prezzi, consulta la pagina CloudWatch dei prezzi di Amazon.

Configurare e abilitare il monitoraggio avanzato

Per utilizzare il monitoraggio avanzato, è necessario creare un ruolo IAM e quindi abilitare il monitoraggio avanzato.

Argomenti

- Creazione di un ruolo IAM per Enhanced Monitoring
- Attivazione e disattivazione del monitoraggio avanzato
- Protezione dal problema del "confused deputy"

Creazione di un ruolo IAM per Enhanced Monitoring

Il monitoraggio avanzato richiede l'autorizzazione ad agire per conto dell'utente per inviare le informazioni sui parametri del sistema operativo ai CloudWatch registri. Concedi le autorizzazioni di Enhanced Monitoring utilizzando un ruolo AWS Identity and Access Management (IAM). È possibile creare questo ruolo quando si abilita il monitoraggio avanzato o lo si crea in anticipo.

Argomenti

- Creazione del ruolo IAM quando si attiva Enhanced Monitoring
- Creazione del ruolo IAM prima di abilitare Enhanced Monitoring

Creazione del ruolo IAM quando si attiva Enhanced Monitoring

Quando si attiva Enhanced Monitoring nella console RDS, Amazon RDS è possibile creare il ruolo IAM necessario. Il ruolo è denominato `rds-monitoring-role`. RDS utilizza questo ruolo per l'istanza database specificata, la replica di lettura o il cluster di database Multi-AZ.

Per creare il ruolo IAM quando si attiva Enhanced Monitoring

1. Segui la procedura riportata in Attivazione e disattivazione del monitoraggio avanzato.
2. Imposta Ruolo di monitoraggio su Predefinito nel passaggio in cui si sceglie un ruolo.

Creazione del ruolo IAM prima di abilitare Enhanced Monitoring

È possibile creare il ruolo richiesto prima di abilitare Enhanced Monitoring. Quando si abilita Enhanced Monitoring, specifica il nome del nuovo ruolo. Si deve creare questo ruolo necessario se si abilita il monitoraggio avanzato utilizzando AWS CLI oppure l'API RDS.

L'utente che abilita il monitoraggio avanzato deve ricevere l'autorizzazione `PassRole`. Per ulteriori informazioni, consulta l'Esempio 2 in Concessione a un utente delle autorizzazioni per il trasferimento di un ruolo a un AWS servizio nella Guida per l'utente IAM.

Per creare un ruolo IAM per Amazon RDS Enhanced Monitoring

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com
2. Nel pannello di navigazione, seleziona Roles (Ruoli).
3. Selezionare Create role (Crea ruolo).
4. Scegliere la scheda Servizio AWS quindi seleziona RDS dall’elenco di servizi.
5. Scegli RDS - Enhanced Monitoring (RDS - Monitoraggio avanzato), quindi seleziona Next (Avanti).
6. Assicurati che le politiche di autorizzazione mostrino Amazon RDSEnhanced MonitoringRole, quindi scegli Avanti.
7. In Nome ruolo, immetti un nome per il ruolo. Ad esempio, specifica **`emaccess`**.

   L'entità affidabile per il tuo ruolo è il AWS servizio monitoring.rds.amazonaws.com.
8. Scegliere Crea ruolo.

Attivazione e disattivazione del monitoraggio avanzato

Puoi gestire il monitoraggio avanzato utilizzando l'API, o RDS. AWS Management Console AWS CLI Puoi anche attivare il monitoraggio avanzato per un cluster DB esistente dalla console.

Console

È possibile attivare il monitoraggio avanzato quando si crea un cluster o una replica di lettura oppure quando si modifica un cluster. Se modifichi un'istanza DB o un cluster per attivare il monitoraggio avanzato, non è necessario riavviare l'istanza DB per rendere effettiva la modifica.

È possibile abilitare il monitoraggio avanzato nella console RDS quando si esegue una delle seguenti operazioni nella pagina Databases (Database):

- Creazione di un cluster: scegli Create database (Crea database).
- Creazione di una replica di lettura: scegli Actions (Operazioni), quindi Create read replica (Crea replica di lettura).
- Modifica un'istanza DB o un cluster DB o un cluster : scegli Modifica.

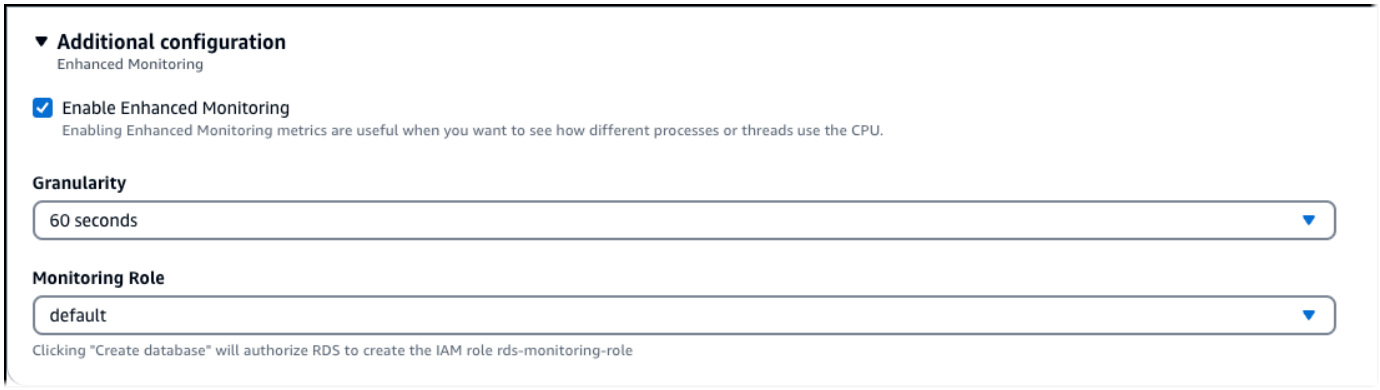
> **Note**
>
> Quando abiliti il monitoraggio avanzato per un cluster DB, non puoi gestire il monitoraggio avanzato per singole istanze DB all'interno del cluster.

Se gestisci il monitoraggio avanzato per singole istanze DB in un cluster DB e la granularità è impostata su valori diversi per istanze diverse, il cluster DB è eterogeneo rispetto al monitoraggio avanzato. In questi casi, non è possibile modificare il cluster DB per gestire il monitoraggio avanzato a livello di cluster.

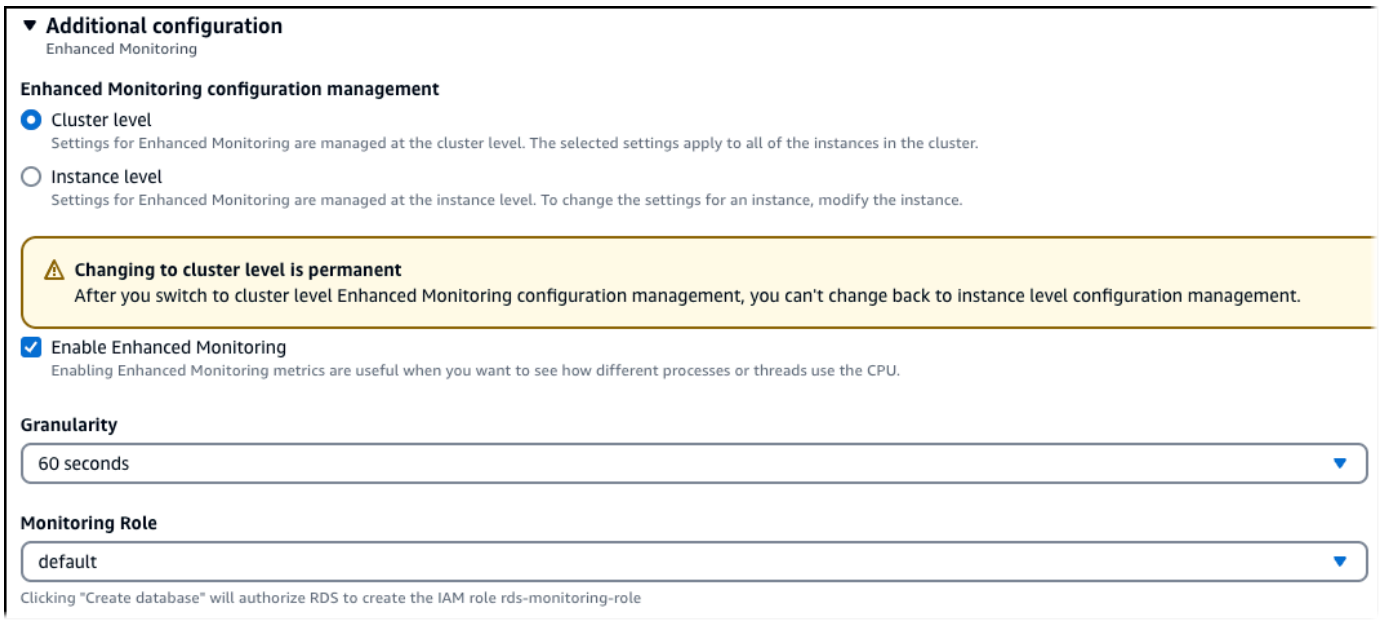
Per attivare o disattivare il monitoraggio avanzato nella console RDS

1. Scorri fino a Additional configuration (Configurazione aggiuntiva).
2. In Monitoring, scegli Enhanced Monitoring per l' DB, il cluster o la replica di lettura. L'abilitazione del monitoraggio avanzato a livello di cluster consente di gestire le impostazioni e le opzioni di monitoraggio avanzato a livello di cluster. Le impostazioni a livello di cluster si applicano a tutte le istanze DB del cluster.Deseleziona l'opzione per disabilitare il monitoraggio avanzato a livello di cluster. Successivamente è possibile modificare le impostazioni di Enhanced Monitoring per le singole istanze DB del cluster.

   Nella pagina Crea database, puoi scegliere di attivare il monitoraggio avanzato a livello di cluster.



Se non abiliti il monitoraggio avanzato durante la creazione di un cluster, puoi modificare il cluster nella pagina Modifica cluster DB.



> **Note**
>
> Non è possibile gestire Enhanced Monitoring per una singola istanza DB in un cluster DB che ha già Enhanced Monitoring gestito a livello di cluster.

3. Non è possibile scegliere di gestire il monitoraggio avanzato a livello di cluster se il cluster è eterogeneo rispetto a Enhanced Monitoring. Per gestire il monitoraggio avanzato a livello di cluster, modifica le impostazioni di monitoraggio avanzato per ogni istanza in modo che corrispondano. Ora puoi scegliere di gestire il monitoraggio avanzato a livello di cluster quando modifichi il cluster.

4. Imposta la proprietà Monitoring Role sul ruolo IAM che hai creato per consentire ad Amazon RDS di comunicare con Amazon CloudWatch Logs per te, oppure scegli Default per fare in modo che RDS crei un ruolo per te denominato. `rds-monitoring-role`
5. Imposta la proprietà Granularity sull'intervallo, in secondi, tra i punti in cui vengono raccolte le metriche per l'istanza DB, il cluster DB o la replica di lettura. La proprietà Granularity (Granularità) può essere impostata su uno dei valori seguenti: `1`, `5`, `10`, `15`, `30` oppure `60`.

   L'intervallo più veloce in cui la console RDS si aggiorna è ogni 5 secondi. Se si imposta la granularità su 1 secondo nella console RDS, vengono comunque visualizzati parametri aggiornati solo ogni 5 secondi. Puoi recuperare gli aggiornamenti delle metriche di 1 secondo utilizzando Logs. CloudWatch

AWS CLI

Per attivare il monitoraggio avanzato utilizzando i comandi seguenti AWS CLI, impostate l'`--monitoring-interval`opzione su un valore diverso da `0` e impostate l'`--monitoring-role-arn`opzione sul ruolo in cui avete creato. Creazione di un ruolo IAM per Enhanced Monitoring

- create-db-cluster
- modify-db-cluster
- create-db-instance
- create-db-instance-read-replica
- modify-db-instance

L'opzione `--monitoring-interval` specifica l'intervallo, in secondi, tra i punti quando vengono raccolti i parametri di monitoraggio avanzato. I valori validi per l'opzione sono `0`, `1`, `5`, `10`, `15`, `30` e `60`.

Per disattivare il monitoraggio avanzato utilizzando il AWS CLI, imposta l'`--monitoring-interval`opzione su `0` in questi comandi.

Example

Nell'esempio seguente viene attivato il monitoraggio avanzato per un'istanza database:

Per LinuxmacOS, oUnix:

```
aws rds modify-db-instance \
```

```
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

L'esempio seguente attiva il monitoraggio avanzato per un cluster DB:

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

Nell'esempio seguente viene attivato il monitoraggio avanzato per un cluster di database Multi-AZ:

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

API RDS

Per attivare il monitoraggio avanzato utilizzando l'API RDS, imposta il parametro `MonitoringInterval` su un valore diverso da `0` e imposta il parametro `MonitoringRoleArn` sul ruolo creato in Creazione di un ruolo IAM per Enhanced Monitoring. Imposta questi parametri nelle seguenti operazioni:

- CreaDBCluster
- ModificaDBCluster
- CreaDBInstance
- CreaDBInstanceReadReplica
- ModificaDBInstance

Il parametro `MonitoringInterval` specifica l'intervallo, in secondi, tra i punti quando vengono raccolti i parametri di monitoraggio avanzato. I valori validi sono `0`, `1`, `5`, `10`, `15`, `30` e `60`.

Per disattivare il monitoraggio avanzato utilizzando l'API RDS, imposta `MonitoringInterval` su `0`.

Protezione dal problema del "confused deputy"

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. Nel AWS, l'impersonificazione tra servizi può portare alla confusione del vicesceriffo. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse del tuo account. Per ulteriori informazioni, consulta Problema del "confused deputy".

Per limitare le autorizzazioni relative alle risorse che Amazon RDS può fornire a un altro servizio, si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e

`aws:SourceAccount` in una policy di attendibilità per il tuo ruolo di monitoraggio avanzato. Se si utilizzano entrambe le chiavi di contesto delle condizioni globali, devono utilizzare lo stesso ID account.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Per Amazon RDS, imposta `aws:SourceArn` su `arn:aws:rds:`*`Region`*`:`*`my-account-id`*`:db:`*`dbname`*.

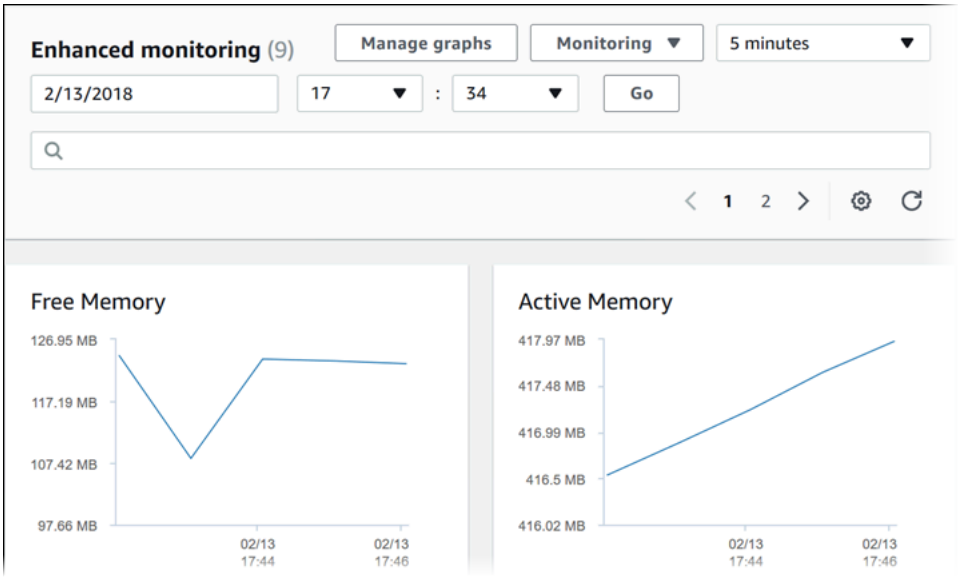
L'esempio seguente usa le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` in una policy di affidabilità per prevenire il problema del "confused deputy".

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-account-id:db:dbname"
        },
        "StringEquals": {
          "aws:SourceAccount": "my-account-id"
        }
      }
    }
  ]
}
```

Visualizzazione delle metriche del sistema operativo nella console RDS

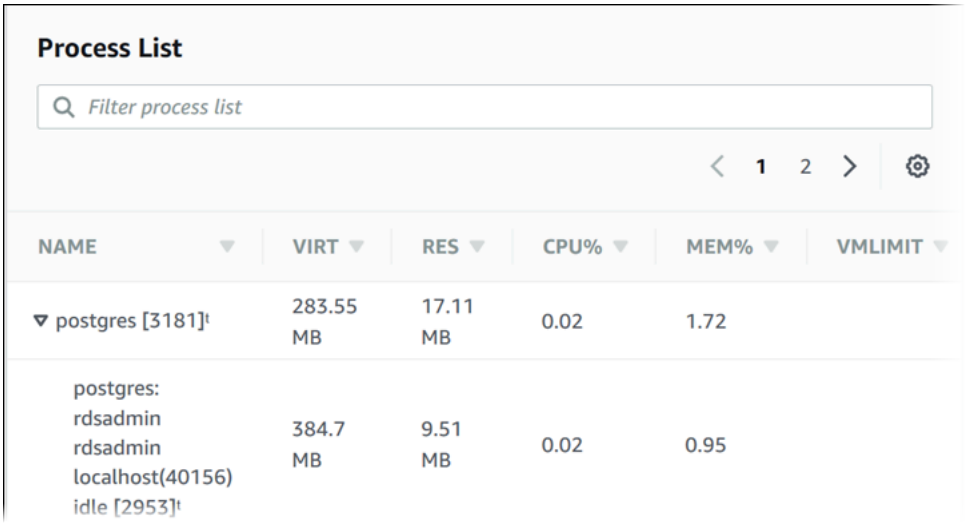
È possibile visualizzare le metriche del sistema operativo riportate da Enhanced Monitoring nella RDS console scegliendo Enhanced monitoring for Monitoring.

L'esempio seguente mostra la pagina Enhanced Monitoring (Monitoraggio avanzato). Per le descrizioni dei parametri di monitoraggio avanzato, consulta Metriche OS nel monitoraggio avanzato.



Se desideri vedere i dettagli per i processi in esecuzione nell'istanza database, scegli OS process list (Elenco processi sistema operativo) per Monitoring (Monitoraggio).

La vista Process List (Elenco processi) è mostrata di seguito.



I parametri del monitoraggio avanzato mostrati nella vista Process List (Elenco processi) sono organizzati nel seguente modo:

- RDSprocessi secondari: mostra un riepilogo dei RDS processi che supportano l'istanza DB, ad esempio `auroraper` i cluster Amazon Aurora DB `mysqld` DB. I thread del processo appaiono nidificati sotto il processo genitore. I thread di processo mostrano CPU l'utilizzo solo perché le altre metriche sono le stesse per tutti i thread del processo. La console visualizza un massimo di 100 processi e thread. I risultati sono una combinazione dei processi e dei thread che CPU consumano

più e più memoria. Se ci sono più di 50 processi e più di 50 thread, la console visualizza i primi 50 consumatori in ciascuna categoria. Questo display aiuta a identificare quali processi stanno avendo il maggiore impatto sulle prestazioni.

- RDSprocessi: mostra un riepilogo delle risorse utilizzate dall'agente di RDS gestione, dei processi di monitoraggio della diagnostica e di altri AWS processi necessari per supportare le istanze RDS DB.
- OS processes (Processi del sistema operativo) – Viene visualizzato un riepilogo dei processi del kernel e di sistema, che generalmente hanno un impatto minimo sulle prestazioni.

Gli elementi elencati per ogni processo sono:

- VIRT— Visualizza la dimensione virtuale del processo.
- RES— Visualizza la memoria fisica effettiva utilizzata dal processo.
- CPU%: visualizza la percentuale della CPU larghezza di banda totale utilizzata dal processo.
- MEM%: visualizza la percentuale della memoria totale utilizzata dal processo.

I dati di monitoraggio visualizzati nella RDS console vengono recuperati da Amazon CloudWatch Logs. Puoi anche recuperare le metriche per un'istanza DB come flusso di log da Logs. CloudWatch Per ulteriori informazioni, consulta Visualizzazione delle metriche del sistema operativo tramite i registri CloudWatch .

I parametri di monitoraggio avanzato non vengono restituiti durante:

- Un failover dell'istanza database.
- Modifica della classe di istanza dell'istanza database (dimensionamento del calcolo).

I parametri del monitoraggio avanzato vengono restituiti durante un riavvio di un'istanza DB perché viene riavviato solo il motore del database. I parametri per il sistema operativo vengono ancora segnalati.

Visualizzazione delle metriche del sistema operativo tramite i registri CloudWatch

Dopo aver abilitato il monitoraggio avanzato per l', è possibile visualizzarne le relative metriche utilizzando CloudWatch Logs, in cui ogni flusso di log rappresenta una singola istanza DB o cluster

DB monitorato. L'identificatore del flusso di log è l'identificativo della risorsa (`DbiResourceId`) per l'istanza database o il cluster di database.

Per visualizzare i dati di log del Monitoraggio avanzato

1. Apri la console all' CloudWatch indirizzo. https://console.aws.amazon.com/cloudwatch/
2. Se necessario, scegli Regione AWS quello in cui si trova l'. Per ulteriori informazioni, consulta la pagina relativa a regioni ed endpoint nei Riferimenti generali di Amazon Web Services.
3. Selezionare Logs (Log) nel riquadro di navigazione.
4. Scegli RDSOSMetricsdall'elenco dei gruppi di log.
5. Scegliere il flusso di log che si desidera visualizzare dall´elenco dei flussi di log.

Riferimento alle metriche per

In questo riferimento, puoi trovare le descrizioni dei parametri di Aurora per Amazon, Performance CloudWatch Insights ed Enhanced Monitoring.

Argomenti

- CloudWatch Parametri Amazon per Amazon Aurora
- CloudWatch Dimensioni Amazon per Aurora
- Disponibilità delle metriche Aurora nella console Amazon RDS
- CloudWatch Parametri Amazon per Amazon RDS Performance Insights
- Parametri contatore di Performance Insights
- Statistiche SQL per Performance Insights
- Metriche OS nel monitoraggio avanzato

CloudWatch Parametri Amazon per Amazon Aurora

Lo spazio dei nomi `AWS/RDS` include i seguenti parametri che si applicano alle entità di database in esecuzione su Amazon Aurora. Alcuni parametri si applicano a Aurora MySQL, Aurora PostgreSQL o a entrambi. Inoltre, alcuni parametri sono specifici per un cluster di database, un'istanza database primaria, un'istanza database di replica o per tutte le istanze database.

Per i parametri del database globale Aurora, consulta CloudWatch Parametri Amazon per l'inoltro della scrittura in Aurora MySQL e CloudWatch Parametri Amazon per l'inoltro delle scritture in Aurora PostgreSQL. Per i parametri delle query parallele Aurora, consulta Monitoraggio delle interrogazioni parallele per Aurora My SQL.

Argomenti

- Parametri a livello di cluster per Amazon Aurora
- Parametri a livello di istanza per Amazon Aurora
- Metriche CloudWatch di utilizzo di

Parametri a livello di cluster per Amazon Aurora

Nella tabella seguente vengono descritti parametri specifici per i cluster Aurora.

Parametro	Descrizione	Si applica a	Unità
`AuroraGlobalDBDataTransferBytes`	In un database globale Aurora, la quantità di dati di redo log trasferiti dalla regione di origine a una AWS regione secondaria. AWS **Note** Questa metrica è disponibile solo in versione secondaria. Regioni AWS	Aurora MySQL e Aurora PostgreSQL	Byte
`AuroraGlobalDBProgressLag`	In un Database globale Aurora, la misura di quanto il cluster secondario si trova dietro il cluster primario sia per le transazioni utente che per le transazioni di sistema. **Note** Questa metrica è disponibile solo nella versione secondaria. Regioni AWS	Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraGlobalDBReplicatedWriteIO`	In un database globale Aurora, il numero di I/O operazioni di scrittura replicate dalla AWS regione primaria al volume del cluster in una regione	Aurora MySQL e Aurora PostgreSQL	Conteggio

Parametro	Descrizione	Si applica a	Unità
	secondaria. AWS I calcoli di fatturazione per le AWS regioni secondarie in un database globale vengono utilizzati `VolumeWriteIOPs` per tenere conto delle scritture eseguite all'interno del cluster. I calcoli di fatturazione per la AWS regione principale in un database globale vengono utilizzati per `VolumeWriteIOPs` tenere conto dell'attività di scrittura all'interno di quel cluster e `AuroraGlobalDBReplicatedWriteIO` per tenere conto della replica tra regioni all'interno del database globale. **Note** Questa metrica è disponibile solo nella versione secondaria. Regioni AWS		

Parametro	Descrizione	Si applica a	Unità
`AuroraGlobalDBReplicationLag`	Per un database globale Aurora, il ritardo durante la replica degli aggiornamenti dalla regione AWS primaria. **Note** Questa metrica è disponibile solo nella versione secondaria. Regioni AWS	Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraGlobalDBRPOLag`	In un Database globale Aurora, il tempo di ritardo dell'obiettivo del punto di ripristino (RPO). Questo parametro misura la distanza del cluster secondario dietro il cluster primario per le transazioni utente. **Note** Questa metrica è disponibile solo nella versione secondaria. Regioni AWS	Aurora MySQL e Aurora PostgreSQL	Millisecondi

Parametro	Descrizione	Si applica a	Unità
`AuroraVolumeBytesLeftTotal`	Spazio disponibile rimanente per il volume del cluster. All'aumentare del volume del cluster, questo valore diminuisce. Se raggiunge lo zero, il cluster segnala un out-of-space errore. Per rilevare se il cluster Aurora MySQL si sta avvicinando al limite di dimensione 128 tebibytes (TiB), questo valore è più semplice e più affidabile da monitorare rispetto a `VolumeBytesUsed`. `AuroraVolumeBytesLeftTotal` tiene conto dello spazio di archiviazione utilizzato per le la manutenzione interna e altre allocazioni che non influiscono sulla fatturazione dell'archiviazione.	Aurora MySQL	Byte
`BacktrackChangeRecordsCreationRate`	Il numero di record delle modifiche di backtrack create in cinque minuti per il cluster DB.	Aurora MySQL	Conteggio per 5 minuti
`BacktrackChangeRecordsStored`	Il numero di record delle modifiche di backtrack utilizzate per il cluster DB.	Aurora MySQL	Conteggio

Parametro	Descrizione	Si applica a	Unità
`BackupRetentionPeriodStorageUsed`	La quantità totale di storage di backup utilizzata per supportare la funzionalità di point-in-time ripristino all'interno della finestra di conservazione dei backup del cluster Aurora DB. Questa quantità è inclusa nel totale riportato dal parametro `TotalBackupStorageBilled` . Calcolato separatamente per ogni cluster Aurora. Per istruzioni, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte
`ServerlessDatabaseCapacity`	Capacità corrente di un cluster di database Aurora Serverless.	Aurora MySQL e Aurora PostgreSQL	Conteggio

Parametro	Descrizione	Si applica a	Unità
`SnapshotStorageUsed`	Il totale dello storage di backup utilizzato da tutti gli snapshot Aurora per un cluster DB Aurora al di fuori della finestra di retention dei backup. Questa quantità è inclusa nel totale riportato dal parametro `TotalBackupStorageBilled`. Calcolato separatamente per ogni cluster Aurora. Per istruzioni, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte
`TotalBackupStorageBilled`	Il totale, espresso in byte, dello storage di backup addebitato per un determinato cluster DB Aurora. Include lo storage di backup misurato dai parametri `BackupRetentionPeriodStorageUsed` e `SnapshotStorageUsed`. Questo parametro viene calcolato separatamente per ogni cluster Aurora. Per istruzioni, consulta Informazioni sull'utilizzo dello storage di backup Amazon Aurora.	Aurora MySQL e Aurora PostgreSQL	Byte

Parametro	Descrizione	Si applica a	Unità
`VolumeBytesUsed`	La quantità di storage utilizzata dal cluster Aurora DB. Questo valore influisce sul costo del cluster database Aurora (per informazioni relative ai prezzi, consulta la pagina Prezzi di Amazon RDS). Questo valore non riflette alcune allocazioni di storage interne che non influenza no la fatturazione dello storage. Per Aurora MySQL è possibile prevedere i out-of-space problemi in modo più accurato verificando se si avvicina allo zero anziché `VolumeBytesUsed` confrontarli con `AuroraVolumeBytesLeftTotal` il limite di archiviazione di 128 TiB. Per i cluster che sono cloni, il valore di questa metrica dipende dalla quantità di dati aggiunti o modificati nel clone. La metrica può anche aumentare o diminuire quando il cluster originale viene eliminato o quando vengono aggiunti o eliminati	Aurora MySQL e Aurora PostgreSQL	Byte

Parametro	Descrizione	Si applica a	Unità
	nuovi cloni. Per maggiori dettagli, consulta Eliminazione di un volume cluster di origine. Tieni presente che non ha senso scegliere un `--period` valore piccolo, perché Amazon RDS raccoglie questi parametri a intervalli, non in modo continuo.		

Parametro	Descrizione	Si applica a	Unità
`VolumeReadIOPs`	Il numero di I/O operazioni di lettura fatturate da un volume del cluster in un intervallo di 5 minuti. Le operazioni di lettura fatturate sono calcolate a livello del volume del cluster, aggregate da tutte le istanze nel cluster di database di Aurora e, in seguito, indicate a intervalli di 5 minuti. Il valore è calcolato prendendo il valore del parametro Read operations per un periodo che supera i 5 minuti. Puoi determinare la quantità delle operazioni di lettura fatturate al secondo prendendo il valore del parametro Billed read operations e dividendo per 300 secondi. Ad esempio, se le Billed read operations restituiscono 13.686, allora le operazioni di lettura fatturate al secondo saranno 45 (13.686 / 300 = 45,62). Le operazioni di lettura fatturate si accumulano per query che richiedono pagine del database che non si trovano nella cache del buffer e, per questo, è	Aurora MySQL e Aurora PostgreSQL	Conteggio per 5 minuti

Parametro	Descrizione	Si applica a	Unità
	necessario caricarle dallo storage. Potresti vedere dei picchi nelle operazioni di lettura fatturate poiché i risultati della query vengono letti dallo storage e, in seguito, caricati nella cache del buffer. Tieni presente che non ha senso scegliere un `--period` valore piccolo, perché Amazon RDS raccoglie questi parametri a intervalli, non in modo continuo. **Tip** Se il tuo cluster Aurora MySQL utilizza una query parallela , potresti vedere un aumento dei valori di `VolumeReadIOPS` . Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare		

Parametro	Descrizione	Si applica a	Unità
	un aumento delle operazioni di lettura e degli addebiti associati.		
`VolumeWriteIOPs`	Il numero di I/O operazioni di scrittura del disco sul volume del cluster, riportate a intervalli di 5 minuti. Per una descrizione dettagliata del modo in cui vengono calcolate le operazioni di scrittura fatturate, consulta `VolumeReadIOPs` . Tieni presente che non ha senso scegliere un `--period` valore piccolo, perché Amazon RDS raccoglie questi parametri a intervalli, non in modo continuo.	Aurora MySQL e Aurora PostgreSQL	Conteggio per 5 minuti

Parametri a livello di istanza per Amazon Aurora

Le seguenti metriche CloudWatch Amazon specifiche dell'istanza si applicano a tutte le istanze Aurora MySQL e Aurora PostgreSQL, salvo diversa indicazione.

Parametro	Descrizione	Si applica a	unità
`AbortedClients`	Numero di connessioni client che non sono state chiuse correttamente.	Aurora MySQL	Conteggio

Parametro	Descrizione	Si applica a	unità
`ActiveTransactions`	Il numero medio delle attuali transazioni in esecuzione su un'istanza database di Aurora al secondo. Per impostazione predefinita, Aurora non abilita questo parametro. Per iniziare a misurare questo valore, imposta `innodb_monitor_enable='all'` nel gruppo di parametri DB per una specifica istanza database.	Aurora MySQL	Conteggio al secondo
`ACUUtilization`	Valore del parametro `ServerlessDatabaseCapacity` diviso per il valore ACU massimo del cluster database. Questa metrica è applicabile solo per Aurora Serverless v2.	Aurora MySQL e Aurora PostgreSQL	Percentuale

Parametro	Descrizione	Si applica a	unità
`AuroraBinlogReplicaLag`	Il periodo di tempo in cui un cluster DB di replica di log binario in esecuzione su Aurora MySQL è in ritardo rispetto alla fonte di replica dei log binari. Un ritardo indica che l'origine genera record più velocemente di quanto la replica possa applicarli. Questo parametro riporta valori diversi a seconda della versione del motore: Aurora MySQL versione 2 Il campo `Seconds_Behind_Master` del `SHOW SLAVE STATUS` MySQL Aurora MySQL versione 3 `SHOW REPLICA STATUS` È possibile utilizzare questo parametro per monitorare gli errori e il ritardo di replica in un cluster che funge da replica di log binario. Il valore del parametro indica quanto segue:	Primario per Aurora MySQL	Secondi

Parametro	Descrizione	Si applica a	unità
	Un valore elevato La replica sta ritardando l'origine della replica. **`0`** o un valore vicino a **`0`** Il processo di replica è attivo e attuale. **`-1`** Aurora non è in grado di determinare il ritardo, che può verificarsi durante la configurazione della replica o quando la replica si trova in uno stato di errore. Poiché la replica dei log binari si verifica solo sull'istanza di scrittura del cluster, consigliamo di utilizzare la versione del parametro associata al ruolo WRITER. Per ulteriori informazioni sull'amministrazione della replica, consulta Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS. Per ulteriori informazioni sulla risoluzione dei problemi, consulta Problemi di replica relativi a Amazon Aurora MySQL.		

Parametro	Descrizione	Si applica a	unità
`AuroraDMLRejectedMasterFull`	Il numero di query inoltrate che vengono rifiutate perché la sessione è piena sull'istanza DB di Writer.	Primario per Aurora MySQL versione 2	Conteggio
`AuroraDMLRejectedWriterFull`	Il numero di query inoltrate che vengono rifiutate perché la sessione è piena sull'istanza DB di Writer.	Primario per Aurora MySQL versione 3	Conteggio
`AuroraEstimatedSharedMemoryBytes`	La quantità stimata di memoria condivisa del buffer o del pool di buffer che è stata utilizzata attivamente durante l'ultimo intervallo di polling configurato.	Aurora PostgreSQL	Byte
`AuroraMemoryHealthState`	Indica lo stato di salute della memoria. Un valore `0` uguale a`NORMAL`. Un valore `10` uguale a`RESERVED`, che significa che il server si sta avvicinando a un livello critico di utilizzo della memoria. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.06.1 e successive	Gauge

Parametro	Descrizione	Si applica a	unità
`AuroraMemoryNumDeclinedSqlTotal`	Il numero incrementale di query è diminuito come parte dell'eliminazione di (OOM). out-of-memory Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.06.1 e successive	Conteggio
`AuroraMemoryNumKillConnTotal`	Il numero incrementale di connessioni chiuse come parte dell'eliminazione dell'OOM. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.06.1 e successive	Conteggio
`AuroraMemoryNumKillQueryTotal`	Il numero incrementale di query è terminato come parte dell'elusione dell'OOM. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.06.1 e successive	Conteggio

Parametro	Descrizione	Si applica a	unità
`AuroraMilliseconds SpentInOomRecovery`	Il periodo di tempo trascorso dallo stato di salute della memoria è sceso al di sotto dello stato normale. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.08.0 e successive	Millisecondi
`AuroraNumOomRecove rySuccessful`	Il numero di volte in cui lo stato di integrità della memoria è stato ripristinato allo stato normale. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.08.0 e successive	Conteggio
`AuroraNumOomRecove ryTriggered`	Il numero di volte in cui lo stato della memoria è sceso al di sotto dello stato normale. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.	Aurora MySQL versione 3.08.0 e successive	Conteggio

Parametro	Descrizione	Si applica a	unità
`AuroraOptimizedReadsCacheHitRatio`	La percentuale di richieste servite dalla cache di lettura ottimizzata. Il valore viene calcolato utilizzando la formula seguente: `orcache_blks_hit/(orcache_blks_hit + storage_blks_read)` Quando `AuroraOptimizedReadsCacheHitRatio` è al 100%, significa che tutte le pagine sono state lette dalla cache di lettura ottimizzata. Se `AuroraOptimizedReadsCacheHitRatio` è`0`, significa che nessuna pagina è stata letta dalla cache di lettura ottimizzata.	Primario per Aurora PostgreSQL	Percentuale
`AuroraReplicaLag`	Per una replica Aurora, la quantità di ritardo durante la replica degli aggiornamenti dall'istanza principale.	Replica per Aurora MySQL e Aurora PostgreSQL	Millisecondi

Parametro	Descrizione	Si applica a	unità
`AuroraReplicaLagMaximum`	Il ritardo massimo tra l'istanza primaria e ogni istanza database Aurora nel cluster di database. Quando le repliche di lettura vengono eliminate o rinominate, può verificarsi un picco temporaneo del ritardo di replica dovuto al processo di riciclaggio della vecchia risorsa. Per ottenere una rappresentazione accurata del ritardo di replica durante quel periodo, si consiglia di monitorare la metrica su ogni istanza di replica letta. `AuroraReplicaLag`	Primario per Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraReplicaLagMinimum`	Il ritardo minimo tra l'istanza primaria e ogni istanza database Aurora nel cluster di database.	Primario per Aurora MySQL e Aurora PostgreSQL	Millisecondi
`AuroraSlowConnectionHandleCount`	Il numero di connessioni che hanno atteso due o più secondi per avviare l'handshake. Questa metrica si applica solo ad Aurora MySQL versione 3.	Aurora MySQL	Conteggio

Parametro	Descrizione	Si applica a	unità
`AuroraSlowHandshakeCount`	Il numero di connessioni che hanno impiegato 50 o più millisecondi per completare l'handshake. Questa metrica si applica solo ad Aurora MySQL versione 3.	Aurora MySQL	Conteggio
`BacktrackWindowActual`	La differenza tra la finestra target di backtrack e la finestra attuale di backtrack.	Primario per Aurora MySQL	Minutes (Minuti)
`BacktrackWindowAlert`	Il numero di volte in cui la finestra attuale di backtrack è più piccola della finestra di backtrack target in un intervallo temporale specifico.	Primario per Aurora MySQL	Conteggio
`BlockedTransactions`	Il numero medio di transazioni nel database bloccate ogni secondo.	Aurora MySQL	Conteggio al secondo
`BufferCacheHitRatio`	La percentuale di richieste gestite dalla cache del buffer.	Aurora MySQL e Aurora PostgreSQL	Percentuale
`CommitLatency`	La durata media impiegata dal motore e dall'archiviazione per completare le operazioni di commit.	Aurora MySQL e Aurora PostgreSQL	Millisecondi

Parametro	Descrizione	Si applica a	unità
`CommitThroughput`	Il numero medio di operazioni di conferma al secondo.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`ConnectionAttempts`	Il numero di tentativi di connessione a un'istanza, a prescindere che vadano a buon fine.	Aurora MySQL	Conteggio

Parametro	Descrizione	Si applica a	unità
`CPUCreditBalance`	Il numero di crediti CPU accumulati da un'istanza, segnalati a intervalli di 5 minuti. Viene utilizzato per determinare per quanto tempo un'istanza database può superare il proprio livello di prestazioni di base a una determinata velocità. Questa metrica si applica solo a queste classi di istanze: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` e `db.t4g` • Aurora PostgreSQL: `db.t3` e `db.t4g` **Note** Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di	Aurora MySQL e Aurora PostgreSQL	Conteggio

Parametro	Descrizione	Si applica a	unità
	classi di istanza database. I crediti di lancio funzionano allo stesso modo in Amazon RDS e in Amazon EC2. Per ulteriori informazioni, consulta Crediti di lancio nella Guida per l'utente di Amazon Elastic Compute Cloud per istanze Linux.		

Parametro	Descrizione	Si applica a	unità
`CPUCreditUsage`	Il numero di crediti CPU consumati durante il periodo specificato, segnalato a intervalli di 5 minuti. Questa metrica misura la quantità di tempo durante la quale i dati fisici CPUs sono stati utilizzati per l'elaborazione delle istruzioni mediante CPUs allocazioni virtuali all'istanza DB. Questa metrica si applica solo a queste classi di istanze: • Aurora MySQL: `db.t2.small` , `db.t2.medium` , `db.t3` e `db.t4g` • Aurora PostgreSQL: `db.t3` e `db.t4g` **Note** Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di	Aurora MySQL e Aurora PostgreSQL	Conteggio

Parametro	Descrizione	Si applica a	unità
	classi di istanza database.		
`CPUSurplusCreditBalance`	Il numero di crediti extra spesi da un'istanza illimitata quando il rispettivo valore `CPUCreditBalance` è pari a zero. Il valore `CPUSurplusCreditBalance` viene saldato con i crediti CPU ottenuti. Se il numero dei crediti extra va oltre il numero massimo di crediti che un'istanza può ottenere in un periodo di 24 ore, i crediti extra spesi, eccedenti il limite, incorreranno in costi aggiuntivi. I parametri di credito CPU sono disponibili solo con una frequenza di 5 minuti.	Aurora MySQL e Aurora PostgreSQL	Crediti (vCPU/minuti)

Parametro	Descrizione	Si applica a	unità
`CPUSurplusCreditsCharged`	Il numero di crediti extra spesi da un'istanza, che non sono saldati con i crediti CPU ottenuti e che pertanto incorrono in costi aggiuntivi. I crediti extra spesi subiscono costi aggiuntivi quando si verifica uno dei seguenti casi: • I crediti extra spesi vanno oltre il numero massimo di crediti che un'istanza può ottenere in un periodo di 24 ore. I crediti extra spesi, che eccedono il limite, subiscono costi aggiuntivi alla fine dell'ora; • l'istanza viene arrestata o terminata; • l'istanza passa da `unlimited` a `standard`. I parametri di credito CPU sono disponibili solo con una frequenza di 5 minuti.	Aurora MySQL e Aurora PostgreSQL	Crediti (vCPU/minuti)
`CPUUtilization`	La percentuale di CPU utilizzata da un'istanza database Aurora.	Aurora MySQL e Aurora PostgreSQL	Percentuale

Parametro	Descrizione	Si applica a	unità
`DatabaseConnections`	Il numero di connessioni di rete client all'istanza del database. Il numero di sessioni del database può essere superiore al valore del parametro perché il valore del parametro non include quanto segue: • Sessioni che non hanno più una connessione di rete ma che il database non ha ripulito • Sessioni create dal motore del database per i propri scopi • Sessioni create dalle funzionalità di esecuzione parallela del motore del database • Sessioni create dal pianificatore dei processi del motore del database • Connessioni Amazon Aurora	Aurora MySQL e Aurora PostgreSQL	Conteggio
`DDLLatency`	La durata media delle richieste, ad esempio, richieste di creazione, modifica e eliminazione.	Aurora MySQL	Millisecondi
`DDLThroughput`	Il numero medio di richieste DDL al secondo.	Aurora MySQL	Conteggio al secondo

Parametro	Descrizione	Si applica a	unità
`Deadlocks`	Il numero medio di deadlock nel database al secondo.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`DeleteLatency`	La durata media delle operazioni di eliminazione.	Aurora MySQL	Millisecondi
`DeleteThroughput`	Il numero medio di query di eliminazione al secondo.	Aurora MySQL	Conteggio al secondo
`DiskQueueDepth`	Il numero di read/write richieste in sospeso in attesa di accesso al disco.	Aurora MySQL e Aurora PostgreSQL	Conteggio
`DMLLatency`	La durata media di inserimenti, aggiornamenti ed eliminazioni.	Aurora MySQL	Millisecondi
`DMLThroughput`	Il numero medio delle inserzioni, degli aggiornamenti e delle eliminazioni al secondo.	Aurora MySQL	Conteggio al secondo
`EngineUptime`	La durata di esecuzione dell'istanza.	Aurora MySQL e Aurora PostgreSQL	Secondi

Parametro	Descrizione	Si applica a	unità
`FreeableMemory`	La quantità di memoria RAM disponibile. Per i database Aurora MySQL e Aurora PostgreSQL, questa metrica riporta il valore del campo di. `MemAvailable /proc/meminfo`	Aurora MySQL e Aurora PostgreSQL	Byte
`FreeEphemeralStorage`	La quantità di NVMe spazio di archiviazione effimero disponibile.	Aurora PostgreSQL	Byte
`FreeLocalStorage`	La quantità di storage locale disponibile. A differenza di altri motori di database, per le istanze database Aurora questo parametro indica la quantità di storage disponibile per ogni istanza database. Questo valore dipende dalla classe dell'istanza database (per informazioni relative ai prezzi, consulta la pagina Prezzi di Amazon RDS). Puoi aumentare la quantità di spazio di storage gratuito per un'istanza scegliendo una classe di istanza database più ampia per l'istanza. (Non valido per Aurora Serverless v2).	Aurora MySQL e Aurora PostgreSQL	Byte

Parametro	Descrizione	Si applica a	unità
`InsertLatency`	La durata media delle operazioni di inserimento.	Aurora MySQL	Millisecondi
`InsertThroughput`	Il numero medio di operazioni di inserimento al secondo.	Aurora MySQL	Conteggio al secondo
`LoginFailures`	Il numero medio di tentativi di login non riusciti al secondo.	Aurora MySQL	Conteggio al secondo
`MaximumUsedTransactionIDs`	L'età dell'ID transazione non sottoposto al comando VACUUM meno recente, nelle transazioni. Se questo valore raggiunge 2.146.483 .648 (2^31 - 1.000.000), il database viene forzato in modalità di sola lettura, per evitare il wraparoun d dell'ID transazione. Per ulteriori informazioni, consulta Impedire gli errori wraparound dell'ID transazione nella documentazione di PostgreSQL.	Aurora PostgreSQL	Conteggio
`NetworkReceiveThroughput`	La quantità di throughput di rete ricevuta dai client da ciascuna istanza nel cluster Aurora DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database Aurora e il volume del cluster.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo (la console mostra Megabyte al secondo)

Parametro	Descrizione	Si applica a	unità
`NetworkThroughput`	La quantità di throughput di rete ricevuta e trasmessa ai client da ciascuna istanza del cluster Aurora DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database Aurora e il volume del cluster.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`NetworkTransmitThroughput`	La quantità di throughput della rete inviato ai client da ogni istanza nel cluster DB Aurora. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo (la console mostra Megabyte al secondo)
`NumBinaryLogFiles`	Il numero di file binlog generati.	Aurora MySQL	Conteggio
`OldestReplicationSlotLag`	L'entità del ritardo della replica più in ritardo in termini di dati WAL ricevuti.	Aurora PostgreSQL	Byte

Parametro	Descrizione	Si applica a	unità
`PurgeBoundary`	Numero di transazione fino al quale è consentito l'elimina zione da InnoDB. Se questa metrica non avanza per lunghi periodi di tempo, è una buona indicazione che l'eliminazione di InnoDB è bloccata da transazioni di lunga durata. Per indagare, controlla le transazioni attive sul tuo cluster Aurora MySQL DB.	Aurora MySQL versione 2, versioni 2.11 e successive Aurora MySQL versione 3, versioni 3.08 e successive	Conteggio
`PurgeFinishedPoint`	Numero di transazione fino al quale viene eseguita l'eliminazione da InnoDB. Questa metrica può aiutarti a esaminare la velocità con cui procede l'eliminazione da InnoDB.	Aurora MySQL versione 2, versioni 2.11 e successive Aurora MySQL versione 3, versioni 3.08 e successive	Conteggio
`Queries`	Numero medio di query eseguite al secondo.	Aurora MySQL	Conteggio al secondo
`RDSToAuroraPostgreSQLReplicaLag`	La quantità di ritardo durante la replica degli aggiornamenti dall'istanza RDS PostgreSQL principale agli altri nodi presenti nel cluster.	Replica per Aurora PostgreSQL	Secondi

Parametro	Descrizione	Si applica a	unità
`ReadIOPS`	Il numero medio di I/O operazioni su disco al secondo, esclusi i report letti e scritti separatamente, a intervalli di 1 minuto.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`ReadIOPSEphemeralStorage`	Il numero medio di I/O operazioni di lettura del disco su NVMe Ephemeral Storage.	Aurora PostgreSQL	Conteggio al secondo
`ReadLatency`	Il tempo medio impiegato per operazione su disco. I/O	Aurora MySQL e Aurora PostgreSQL	Secondi
`ReadLatencyEphemeralStorage`	Il tempo medio impiegato per ogni I/O operazione di lettura del disco per NVMe Ephemeral Storage.	Aurora PostgreSQL	Millisecondi
`ReadThroughput`	Il numero medio di byte letti dal disco al secondo.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`ReadThroughputEphemeralStorage`	Il numero medio di byte letti dal disco al secondo per Ephemeral Storage. NVMe	Aurora PostgreSQL	Byte al secondo
`ReplicationSlotDiskUsage`	La quantità di spazio su disco occupata dai file degli slot di replica.	Aurora PostgreSQL	Byte

Parametro	Descrizione	Si applica a	unità
`ResultSetCacheHitRatio`	La percentuale di richieste servite dalla cache Resultset.	Aurora MySQL versione 2	Percentuale
`RollbackSegmentHistoryListLength`	I log di annullamento che registrano le transazioni sottoposte a commit con record contrassegnati per l'eliminazione. Questi record sono pianificati per essere elaborati dall'operazione di rimozione InnoDB.	Aurora MySQL	Conteggio
`RowLockTime`	Il tempo totale impiegato per l'acquisizione di blocchi di riga per le tabelle InnoDB.	Aurora MySQL	Millisecondi
`SelectLatency`	La durata media per le operazioni di selezione.	Aurora MySQL	Millisecondi
`SelectThroughput`	Numero medio di query di selezione al secondo.	Aurora MySQL	Conteggio al secondo
`ServerlessDatabaseCapacity`	Capacità corrente di un cluster di database Aurora Serverless.	Aurora MySQL e Aurora PostgreSQL	Conteggio
`StorageNetworkReceiveThroughput`	La quantità di velocità effettiva di rete ricevuta dal sottosistema di archiviazione Aurora mediante ogni istanza nel cluster di database.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo

Parametro	Descrizione	Si applica a	unità
`StorageNetworkThroughput`	La quantità di throughput di rete ricevuta e inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`StorageNetworkTransmitThroughput`	La quantità di throughput di rete inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`SumBinaryLogSize`	La dimensione totale dei file binlog.	Aurora MySQL	Byte
`SwapUsage`	La quantità di spazio di scambio utilizzato. Questo parametro non è disponibile per le seguenti classi di istanza database: • db.r3.*, db.r4.* e db.r7g.* (Aurora MySQL) • db.r7g.* (Aurora PostgreSQL)	Aurora MySQL e Aurora PostgreSQL	Byte

Parametro	Descrizione	Si applica a	unità
`TempStorageIOPS`	Numero di IOPS per le operazioni sia di lettura che di scrittura eseguite sull'archiviazione locale collegata all'istanza database. Questo parametro rappresenta un conteggio e viene misurato una volta al secondo. Questa metrica è applicabile solo per Aurora Serverless v2.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`TempStorageThroughput`	Quantità di dati trasferiti da e verso l'archiviazione locale associata all'istanza database. Questo parametro rappresenta i byte e viene misurato una volta al secondo. Questa metrica è applicabile solo per Aurora Serverless v2.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`TransactionAgeMaximum`	L'età della transazione attiva più vecchia in esecuzione.	Aurora MySQL versione 3, versioni 3.08 e successive	Secondi

Parametro	Descrizione	Si applica a	unità
`TransactionLogsDiskUsage`	La quantità di spazio su disco occupata dai log delle transazioni nell'istanza database di Aurora PostgreSQL. Questa metrica viene generata solo quando Aurora PostgreSQL utilizza la replica logica o. AWS Database Migration Service Per impostazione predefinita, Aurora PostgreSQL utilizza i record di log, non i log delle transazioni. Quando i log delle transazioni non sono in uso, il valore per questo parametro è `-1`.	Primario per Aurora PostgreSQL	Byte
`TruncateFinishedPoint`	L'identificatore della transazione fino al quale viene eseguito il troncamento dell'annullamento.	Aurora MySQL versione 2, versioni 2.11 e successive Aurora MySQL versione 3, versioni 3.08 e successive	Conteggio
`UpdateLatency`	La durata media per le operazioni di aggiornamento.	Aurora MySQL	Millisecondi
`UpdateThroughput`	Il numero medio di aggiornamenti al secondo.	Aurora MySQL	Conteggio al secondo

Parametro	Descrizione	Si applica a	unità
`WriteIOPS`	Il numero di registri di scrittura di memoria Aurora generati al secondo. Questo è più o meno il numero di record di registro generati dal database. Questi non corrispondono a 8K pagine di scrittura e non corrispondono ai pacchetti di rete inviati.	Aurora MySQL e Aurora PostgreSQL	Conteggio al secondo
`WriteIOPSEphemeralStorage`	Il numero medio di I/O operazioni di scrittura su disco su Ephemeral Storage. NVMe	Aurora PostgreSQL	Conteggio al secondo
`WriteLatency`	Il tempo medio impiegato per operazione su disco. I/O	Aurora MySQL e Aurora PostgreSQL	Secondi
`WriteLatencyEphemeralStorage`	Il tempo medio impiegato per ogni I/O operazione di scrittura su disco per NVMe Ephemeral Storage.	Aurora PostgreSQL	Millisecondi
`WriteThroughput`	Il numero medio di byte scritti nello storage persistente ogni secondo.	Aurora MySQL e Aurora PostgreSQL	Byte al secondo
`WriteThroughputEphemeralStorage`	Il numero medio di byte scritti su disco al secondo per lo storage Ephemeral. NVMe	Aurora PostgreSQL	Byte al secondo

Metriche CloudWatch di utilizzo di

Il `AWS/Usage` namespace in Amazon CloudWatch include parametri di utilizzo a livello di account per le quote dei servizi Amazon RDS. CloudWatch raccoglie automaticamente i parametri di utilizzo per tutti. Regioni AWS

Per ulteriori informazioni, consulta i parametri di CloudWatch utilizzo nella Amazon CloudWatch User Guide. Per ulteriori informazioni sulle quote, consulta Quote e vincoli per Amazon Aurora e Requesting a quota increase nella Guida per l'utente di Service Quotas.

Parametro	Descrizione	Unità*
`AuthorizationsPerDBSecurityGroup`	Il numero di regole di ingresso per gruppo di sicurezza DB nel tuo. Account AWS Il valore utilizzato è il numero massimo di regole di ingresso in un gruppo di sicurezza DB dell'account. Altri gruppi di sicurezza DB nell'account potrebbero avere un numero inferiore di regole di ingresso.	Conteggio
`CustomEndpointsPerDBCluster`	Il numero di endpoint personalizzati per cluster DB nel tuo. Account AWS Il valore utilizzato è il numero massimo di endpoint personalizzati in un cluster DB dell'account. Altri cluster DB nell'account potrebbero avere un numero inferiore di endpoint personalizzati.	Conteggio
`DBClusterParameterGroups`	Il numero di gruppi di parametri del cluster di database nel tuo Account AWS. I gruppi di parametri di default non vengono conteggiati.	Conteggio
`DBClusterRoles`	Il numero di ruoli associati AWS Identity and Access Management (IAM) per cluster DB nel tuo Account AWS. Il valore utilizzato è il numero massimo di ruoli IAM associati per un cluster DB nell'account. Altri cluster DB nell'account potrebbero avere un numero inferiore di ruoli IAM associati.	Conteggio
`DBClusters`	Il numero di cluster di database Amazon Aurora nel tuo Account AWS.	Conteggio
`DBInstanceRoles`	Il numero di ruoli associati AWS Identity and Access Management (IAM) per istanza DB nel tuo Account AWS. Il	Conteggio

Parametro	Descrizione	Unità*
	valore utilizzato è il numero massimo di ruoli IAM associati per un'istanza DB nell'account. Altre istanze DB nell'acco unt potrebbero avere un numero inferiore di ruoli IAM associati.	
`DBInstances`	Il numero di istanze database nel tuo Account AWS.	Conteggio
`DBParamet erGroups`	Il numero di gruppi di parametri database nel tuo Account AWS. I gruppi di parametri database di default non vengono conteggiati.	Conteggio
`DBSubnetGroups`	Il numero di gruppi di sottoreti nel tuo Account AWS. Il gruppo di sottoreti predefinito non viene conteggiato.	Conteggio
`EventSubs criptions`	Il numero di abbonamenti per la notifica degli eventi presenti nel tuo. Account AWS	Conteggio
`Integrations`	Il numero di integrazioni zero-ETL con Amazon Redshift nel tuo. Account AWS	Conteggio
`ManualClu sterSnapshots`	Il numero di snapshot del cluster di database creati manualmente nel tuo Account AWS. Gli snapshot non validi non vengono conteggiati.	Conteggio
`OptionGroups`	Il numero di gruppi di opzioni nel tuo Account AWS. I gruppi di opzioni di default non vengono conteggiati.	Conteggio
`Proxies`	Il numero di proxy RDS presenti nel tuo account. AWS	Conteggio
`ReadRepli casPerMaster`	Il numero di repliche di lettura per istanza DB nel tuo account. Il valore utilizzato è il numero massimo di repliche di lettura per un'istanza DB nell'account. Altre istanze DB nell'account potrebbero avere un numero inferiore di repliche di lettura.	Conteggio

Parametro	Descrizione	Unità*
`ReservedDBInstances`	Il numero di istanze database riservate nel tuo Account AWS. Le istanze ritirate o rifiutate non vengono conteggiate.	Conteggio
`SubnetsPerDBSubnetGroup`	Il numero di sottoreti per gruppo di sottoreti DB nel tuo. Account AWS Il numero massimo di sottoreti per un gruppo di sottoreti DB nell'account. Altri gruppi di sottoreti DB nell'account potrebbero avere un numero inferiore di sottoreti.	Conteggio

> **Note**
>
> Amazon RDS non pubblica unità per i parametri di utilizzo. CloudWatch Le unità sono presenti solo nella documentazione.

CloudWatch Dimensioni Amazon per Aurora

Puoi filtrare i dati delle metriche di Aurora utilizzando qualsiasi dimensione nella tabella seguente.

Dimensione	Filtra i dati richiesti per...
`DBInstanceIdentifier`	Un'istanza database specifica.
`DBClusterIdentifier`	Un cluster di database Aurora specifico.
`DBClusterIdentifier, Role`	Un cluster Aurora DB specifico, che aggrega la metrica per ruolo di istanza (/). WRITER READER Ad esempio, puoi aggregare le metriche per tutte le READER istanze che appartengono a un cluster.
`DbClusterIdentifier, EngineName`	Una combinazione di nome specifica applicabile al cluster database Aurora e al motore Aurora. Ad esempio, è possibile visualizzare il parametro `VolumeReadIOPs` per il cluster `ams1` e il motore `aurora`.

Dimensione	Filtra i dati richiesti per...
`DatabaseClass`	Tutte le istanze di una classe di database. Ad esempio, puoi aggregare i parametri per tutte le istanze che appartengono alla classe database `db.r5.large` .
`EngineName`	Solo il nome del motore identificato. Ad esempio, puoi aggregare i parametri per tutte le istanze con nome del motore `aurora-postgresql` .
`SourceRegion`	Usa solo la regione specificata. Ad esempio, puoi aggregare i parametri per tutte le istanze database nella regione `us-east-1` .

Disponibilità delle metriche Aurora nella console Amazon RDS

Non tutte le metriche fornite da Amazon Aurora sono disponibili nella console AmazonRDS. Puoi visualizzare queste metriche utilizzando strumenti come and. AWS CLI CloudWatch API Inoltre, alcune metriche nella RDS console Amazon vengono mostrate solo per classi di istanze specifiche o con nomi e unità di misura diversi.

Argomenti

- Parametri Aurora disponibili nella vista Last Hour (Ultima ora)
- Aurora metriche disponibili in casi specifici
- Parametri Aurora non disponibili nella console

Parametri Aurora disponibili nella vista Last Hour (Ultima ora)

Puoi visualizzare un sottoinsieme di metriche Aurora categorizzate nella visualizzazione Last Hour predefinita nella console Amazon. RDS La tabella seguente elenca le categorie e i parametri associati visualizzati nella RDS console Amazon per un'istanza Aurora.

Categoria	Parametri
SQL	`ActiveTransactions` `BlockedTransactions`

Categoria	Parametri
	BufferCacheHitRatio CommitLatency CommitThroughput DatabaseConnections DDLLatency DDLThroughput Deadlocks DMLLatency DMLThroughput LoginFailures ResultSetCacheHitRatio SelectLatency SelectThroughput

Categoria	Parametri

System (Sistema)	`AuroraReplicaLag`
	`AuroraReplicaLagMaximum`
	`AuroraReplicaLagMinimum`
	`CPUCreditBalance`
	`CPUCreditUsage`
	`CPUUtilization`
	`FreeableMemory`
	`FreeLocalStorage` (Non valido per Aurora Serverless v2).
	`NetworkReceiveThroughput`

Distribuzione	`AuroraReplicaLag`
	`BufferCacheHitRatio`
	`ResultSetCacheHitRatio`
	`SelectThroughput`

Aurora metriche disponibili in casi specifici

Inoltre, alcuni dei parametri Aurora sono visualizzati solo per classi di istanze specifiche, solo per istanze database o con nomi e unità di misura diversi:

- Le `CPUCreditUsage` metriche `CPUCreditBalance` e vengono visualizzate solo per le classi di istanze Aurora SQL `db.t2` My e per le classi di istanze SQL `db.t3` Aurora Postgre.
- I seguenti parametri che sono visualizzati con nomi diversi, come elencato:

Parametro	Display name (Nome visualizzato)
`AuroraReplicaLagMaximum`	Ritardo replica massimo

Parametro	Display name (Nome visualizzato)
AuroraReplicaLagMinimum	Ritardo replica minimo
DDLThroughput	DDL
NetworkReceiveThroughput	Throughput di rete
VolumeBytesUsed	[Fatturati] Byte di volume utilizzati
VolumeReadIOPs	[Fatturato] Volume letto IOPS
VolumeWriteIOPs	[Fatturato] Scrittura in volume IOPS

- Le seguenti metriche si applicano a un intero cluster Aurora DB, ma vengono visualizzate solo quando si visualizzano le istanze DB per un cluster Aurora DB nella console Amazon: RDS
  - VolumeBytesUsed
  - VolumeReadIOPs
  - VolumeWriteIOPs
- Le seguenti metriche vengono visualizzate in megabyte, anziché in byte, nella console Amazon: RDS
  - FreeableMemory
  - FreeLocalStorage
  - NetworkReceiveThroughput
  - NetworkTransmitThroughput
- Le seguenti metriche si applicano a un cluster Aurora SQL Postgre DB con Aurora Optimized Reads:
  - AuroraOptimizedReadsCacheHitRatio
  - FreeEphemeralStorage
  - ReadIOPSEphemeralStorage
  - ReadLatencyEphemeralStorage
  - ReadThroughputEphemeralStorage
  - WriteIOPSEphemeralStorage
  - WriteLatencyEphemeralStorage
  - WriteThroughputEphemeralStorage

Parametri Aurora non disponibili nella console

Le seguenti metriche Aurora non sono disponibili nella console Amazon: RDS

- `AuroraBinlogReplicaLag`
- `DeleteLatency`
- `DeleteThroughput`
- `EngineUptime`
- `InsertLatency`
- `InsertThroughput`
- `NetworkThroughput`
- `Queries`
- `UpdateLatency`
- `UpdateThroughput`

CloudWatch Parametri Amazon per Amazon RDS Performance Insights

Performance Insights pubblica automaticamente alcune metriche su Amazon. CloudWatch Gli stessi dati possono essere interrogati da Performance Insights, ma l'inserimento delle metriche CloudWatch semplifica l'aggiunta di allarmi. CloudWatch Inoltre, semplifica l'aggiunta delle metriche alle dashboard esistenti. CloudWatch

Parametro	Descrizione
DBLoad	Il numero di sessioni attive per il database. Generalmente, si richiedono i dati per il numero medio di sessioni attive. In Performance Insights, questi dati sono oggetto di query come `db.load.avg` .
DBLoadCPU	Il numero di sessioni attive in cui il tipo evento di attesa è CPU. In Performance Insights, questi dati sono oggetto di query come `db.load.avg` , filtrate per tipo di evento di attesa `CPU`.

Parametro	Descrizione
DBLoadNon CPU	Il numero di sessioni attive in cui il tipo evento di attesa non è CPU.
DBLoadRelativeToNumVCPUs	Il rapporto tra il carico del DB e il numero di dati virtuali CPUs per il database.

> **Note**
>
> Queste metriche vengono pubblicate su CloudWatch solo in caso di carico sull'istanza DB.

Puoi esaminare queste metriche utilizzando la CloudWatch console AWS CLI, l'o l' CloudWatch API. Puoi anche esaminare altre metriche dei contatori di Performance Insights utilizzando una speciale funzione matematica metrica. Per ulteriori informazioni, consulta Interrogazione di altre metriche dei contatori di Performance Insights in CloudWatch.

Ad esempio, è possibile ottenere le statistiche per la `DBLoad` metrica eseguendo il comando. get-metric-statistics

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

Questo esempio genera un output simile a quello riportato di seguito.

```
{
  "Datapoints": [
  {
  "Timestamp": "2021-07-19T21:30:00Z",
  "Unit": "None",
  "Average": 2.1
```

```
},
{
"Timestamp": "2021-07-19T21:34:00Z",
"Unit": "None",
"Average": 1.7
},
{
"Timestamp": "2021-07-19T21:35:00Z",
"Unit": "None",
"Average": 2.8
},
{
"Timestamp": "2021-07-19T21:31:00Z",
"Unit": "None",
"Average": 1.5
},
{
"Timestamp": "2021-07-19T21:32:00Z",
"Unit": "None",
"Average": 1.8
},
{
"Timestamp": "2021-07-19T21:29:00Z",
"Unit": "None",
"Average": 3.0
},
{
"Timestamp": "2021-07-19T21:33:00Z",
"Unit": "None",
"Average": 2.4
}
],
"Label": "DBLoad"
}
```


Per ulteriori informazioni su CloudWatch, consulta What is Amazon CloudWatch? nella Amazon CloudWatch User Guide.

Interrogazione di altre metriche dei contatori di Performance Insights in CloudWatch

> **Note**
>
> Se abiliti la modalità avanzata di Database Insights, Amazon RDS pubblica i parametri dei contatori di Performance Insights su Amazon. CloudWatch Con Database Insights, non è necessario utilizzare la funzione matematica delle `DB_PERF_INSIGHTS` metriche. Puoi utilizzare la dashboard di CloudWatch Database Insights per cercare, interrogare e impostare allarmi per le metriche dei contatori di Performance Insights.

È possibile eseguire interrogazioni, avvisi e creare grafici sulle metriche di RDS Performance Insights da. CloudWatch È possibile accedere alle informazioni sull' utilizzando la funzione matematica `DB_PERF_INSIGHTS` metrica for. CloudWatch Questa funzione consente di utilizzare le metriche di Performance Insights che non vengono riportate direttamente per CloudWatch creare una nuova serie temporale.

È possibile utilizzare la nuova funzione Metric Math facendo clic sul menu a discesa Aggiungi matematica nella schermata Seleziona metrica nella console. CloudWatch Puoi usarlo per creare allarmi e grafici sulle metriche di Performance Insights o su combinazioni di metriche di Performance CloudWatch Insights, inclusi allarmi ad alta risoluzione per metriche inferiori al minuto. È inoltre possibile utilizzare la funzione a livello di codice includendo l'espressione Metric Math in una richiesta. `get-metric-data` Per ulteriori informazioni, vedere Sintassi e funzioni matematiche delle metriche e Creare un allarme sulle metriche dei contatori di Performance Insights da un database. AWS

Parametri contatore di Performance Insights

I parametri contatore sono parametri prestazionali di sistema operativo e database nel pannello di controllo di Performance Insights. Per agevolare l'individuazione e l'analisi di problemi legati alle prestazioni, è possibile correlare i parametri contatore ai carichi dei database. È necessario aggiungere una funzione statistica alla metrica per ottenere i valori della metrica. Ad esempio, le funzioni supportate per la metrica `os.memory.active` sono `.avg`, `.min`,`.max`, `.sum` e `.sample_count`.

Le metriche dei contatori vengono raccolte una volta al minuto. La raccolta delle metriche del sistema operativo dipende dall'attivazione o dalla disattivazione della funzionalità Monitoraggio avanzato. Se la funzionalità Monitoraggio avanzato è disattivata, le metriche del sistema operativo vengono

raccolte una volta al minuto. Se la funzionalità Monitoraggio avanzato è attivata, le metriche del sistema operativo vengono raccolte per il periodo di tempo selezionato. Per ulteriori informazioni sull'attivazione o sulla disattivazione della funzionalità Monitoraggio avanzato, consulta Attivazione e disattivazione del monitoraggio avanzato.

Argomenti

- Contatori del sistema operativo in Performance Insights
- Contatori di Performance Insights per Aurora MySQL
- Contatori di Performance Insights per Aurora PostgreSQL

Contatori del sistema operativo in Performance Insights

I seguenti contatori del sistema operativo, con prefisso `os`, sono disponibili in Approfondimenti sulle prestazioni per Aurora PostgreSQL e Aurora MySQL.

Puoi utilizzare l'API `ListAvailableResourceMetrics` per l'elenco delle metriche dei contatori disponibili per l'istanza database. Per ulteriori informazioni, consulta la guida ListAvailableResourceMetricsdi riferimento dell'API Amazon RDS Performance Insights.

Contatore	Tipo	Unità	Parametro	Descrizione
Attivo	Memoria	Kilobyte	os.memory .active	La quantità di memoria assegnata, in kilobyte.
Buffer	Memoria	Kilobyte	os.memory .buffers	La quantità di memoria utilizzata per il buffering delle richieste di I/O prima della scrittura sul dispositivo di storage, in kilobyte.

Contatore	Tipo	Unità	Parametro	Descrizione
Cached	Memoria	Kilobyte	os.memory .cached	La quantità di memoria utilizzata per la memorizzazione nella cache dell'I/O basato sul file system, in kilobyte.
DB Cache	Memoria	Byte	os.memory .db.cache	La quantità di memoria utilizzata per la cache della pagina in base al processo del database, incluso tmpfs (shmem), in byte.
DB Resident Set Size	Memoria	Byte	os.memory.db. residentSetSize	La quantità di memoria utilizzata per la cache anonima e di swap in base al processo del database, escluso tmpfs (shmem), in byte.

Contatore	Tipo	Unità	Parametro	Descrizione
DB Swap	Memoria	Byte	os.memory .db.swap	La quantità di memoria utilizzat a per lo scambio dal processo del database, in byte.
Dirty	Memoria	Kilobyte	os.memory.dirty	La quantità di pagine di memoria nella RAM che sono state modificat e ma non scritte nel relativo blocco di dati nello storage, in kilobyte.
Gratuito	Memoria	Kilobyte	os.memory.free	La quantità di memoria non assegnata, in kilobyte.
Huge Pages libere	Memoria	Pagine	os.memoria. hugePagesFree	Il numero di pagine di grandi dimensioni gratuite. Le pagine di grandi dimensioni sono una caratteri stica del kernel di Linux.

Contatore	Tipo	Unità	Parametro	Descrizione
Huge Pages Rsvd	Memoria	Pagine	os.memoria.hugePagesRsvd	Il numero di pagine di grandi dimensioni impegnate.
Dimensioni Huge Pages	Memoria	Kilobyte	os.memoria.hugePagesSize	La dimensione per ogni unità delle pagine di grandi dimensioni, in kilobyte.
Huge Pages Surp	Memoria	Pagine	os.memoria.hugePagesSurp	Il numero di pagine di grandi dimensioni in eccesso disponibili sul totale.
Totale Huge Pages	Memoria	Pagine	os.memoria.hugePagesTotal	Il numero totale di Huge Pages.
Inattivo	Memoria	Kilobyte	os.memory.inactive	La quantità di pagine di memoria utilizzate meno frequentemente, in kilobyte.

Contatore	Tipo	Unità	Parametro	Descrizione
Mapped	Memoria	Kilobyte	os.memory .mapped	La quantità totale di contenuti del sistema di file che è mappata in memoria all'interno di uno spazio di indirizzamento del processo, in kilobyte.
Out of Memory Kill Count	Memoria	Uccide	memoria del sistema operativo. outOfMemoryKillCount	Il numero di interruzioni OOM avvenute nell'ultimo intervallo di raccolta.
Tabelle delle pagine	Memoria	Kilobyte	os.memory .pageTables	La quantità di memoria utilizzata dalle tabelle della pagina, in kilobyte.
Slab	Memoria	Kilobyte	os.memory.slab	La quantità di strutture dati del kernel riutilizzabili, in kilobyte.
Totale	Memoria	Kilobyte	os.memory.total	La quantità totale di memoria, in kilobyte.

Contatore	Tipo	Unità	Parametro	Descrizione
Writeback	Memoria	Kilobyte	os.memory .writeback	La quantità di pagine sporche nella RAM che sono ancora scritte nello storage di backup, in kilobyte.
Guest	Utilizzo CPU	Percentuale	os.cpuUtilization. guest	La percentuale di CPU utilizzat a dai programmi guest.
Idle	Utilizzo CPU	Percentuale	os.cpuUtilization. idle	La percentuale di tempo CPU che è inattiva.
Irq	Utilizzo CPU	Percentuale	os.cpuUtilization. irq	La percentuale di CPU utilizzata dalle interruzioni dei software.
Nice	Utilizzo CPU	Percentuale	os.cpuUtilization. nice	La percentuale di CPU utilizzat a dai programmi in esecuzione con priorità più bassa.
Steal	Utilizzo CPU	Percentuale	os.cpuUtilization. steal	La percentuale di CPU utilizzata da altre macchine virtuali.

Contatore	Tipo	Unità	Parametro	Descrizione
System (Sistema)	Utilizzo CPU	Percentuale	os.cpuUtilization.system	La percentuale di CPU utilizzata dal kernel.
Totale	Utilizzo CPU	Percentuale	os.cpuUtilization.total	La percentuale totale del CPU utilizzata. Questo valore include il valore nice.
Utente	Utilizzo CPU	Percentuale	os.cpuUtilization.user	La percentuale di CPU utilizzata dai programmi utente.
Attendi	Utilizzo CPU	Percentuale	os.cpuUtilization.wait	La percentuale di CPU non utilizzata durante l'attesa per l'accesso I/O.
Aurora Storage Aurora Storage Bytes Rx	I/O del disco	Byte al secondo	OS.DiskIO.AuroraStorage.auroraStorageBytesRx	Il numero di byte ricevuti per l'archiviazione Aurora al secondo.
Aurora Storage Aurora Storage Bytes Tx	I/O del disco	Byte al secondo	os.diskio.Aurora Storage.auroraStorageBytesTx	Il numero di byte caricati per archiviazione Aurora al secondo.
Aurora Storage Disk Queue Depth	I/O del disco	Richieste	os.diskio.Aurora Storage.diskQueueDepth	La lunghezza della coda del disco di archiviazione Aurora.

Contatore	Tipo	Unità	Parametro	Descrizione
Aurora Storage Leggi PS IOs	I/O del disco	Richieste al secondo	os.diskio .AuroraStorage.Leggi PS IOs	Il numero di operazioni di lettura al secondo.
Aurora Storage Read Latency	I/O del disco	Millisecondi	os.diskIO .auroraStorage.readLatency	La latenza media di una richiesta di I/O di lettura verso lo storage Aurora, in millisecondi.
Throughput di lettura di Aurora Storage	I/O del disco	Byte al secondo	os.diskIO .auroraStorage.readThroughput	La quantità di velocità effettiva di rete utilizzata dalle richieste al cluster di database, espressa in byte al secondo.
Aurora Storage Write PS IOs	I/O del disco	Richieste al secondo	os.diskio .AuroraStorage.Scrivi PS IOs	Il numero di operazioni di scrittura al secondo.
Aurora Storage Write Latency	I/O del disco	Millisecondi	os.diskIO .auroraStorage.writeLatency	La latenza media di una richiesta di I/O di scrittura verso lo storage Aurora, in millisecondi.

Contatore	Tipo	Unità	Parametro	Descrizione
Aurora Storage Write Throughput	I/O del disco	Byte al secondo	os.diskIO.auroraStorage.writeThroughput	La quantità di throughput di rete effettivo utilizzato dalle risposte del cluster di database, espressa in byte al secondo.
Rdstemp Avg Queue Len	I/O del disco	Richieste	os.diskio.RDSTemp.avgQueueLen	Il numero di richieste in attesa nella coda del dispositivo I/O.
Restemp Avg Req Sz	I/O del disco	Richieste	OS.DiskIO.RSTemp.avgReqSz	Il numero di richieste in attesa nella coda del dispositivo I/O.
Restemp Await	I/O del disco	Millisecondi	os.diskIO.rdstemp.await	Il numero di millisecondi necessari per rispondere alle richieste, compreso il tempo della coda e il tempo del servizio.

Contatore	Tipo	Unità	Parametro	Descrizione
Sistema operativo Read Temp IOs	I/O del disco	Richieste	os.diskio .rdstemp.LeggiPS IOs	Il numero di operazioni di lettura al secondo.
Rdstemp Read KB	I/O del disco	Kilobyte	os.diskIO .rdstemp.readKb	Il numero totale di kilobyte letti.
Rdstemp Read KB PS	I/O del disco	Kilobyte al secondo	os.diskIO .rdstemp. readKbPS	Il numero di kilobytes letti al secondo.
Rdstemp Rrqm PS	I/O del disco	Richieste al secondo	os.diskIO .rdstemp.rrqmPS	Il numero di richieste di lettura unite in coda al secondo.
Rdstemp TPS	I/O del disco	Transazioni al secondo	os.diskIO .rdstemp.tps	Il numero di transazioni I/O al secondo.
Restemp Util	I/O del disco	Percentuale	os.diskIO .rdstemp.util	La percentuale di tempo della CPU durante il quale sono state emesse le richieste.
Sistema operativo di scrittura Restemp IOs	I/O del disco	Richieste al secondo	os.diskio .rdstemp.WritePS IOs	Il numero di operazioni di scrittura al secondo.
Rdstemp Write KB	I/O del disco	Kilobyte	os.diskIO .rdstemp.writeKb	Il numero totale di kilobyte scritti.

Contatore	Tipo	Unità	Parametro	Descrizione
Rdstemp Write KB PS	I/O del disco	Kilobyte al secondo	os.diskIO .rdstemp. writeKbPS	Il numero di kilobytes scritti al secondo.
Rdstemp Wrqm PS	I/O del disco	Richieste al secondo	os.diskIO .rdstemp. wrqmPS	Il numero di richieste di scrittura unite in coda al secondo.
Bloccato	Attività	Attività	os.tasks.blocked	Il numero di attività che sono bloccate.
In esecuzione	Attività	Attività	os.tasks.running	Il numero di attività che sono in esecuzione.
Sleeping	Attività	Attività	os.tasks.sleeping	Il numero di attività che sono a riposo.
Arrestato	Attività	Attività	os.tasks.stopped	Il numero di attività che sono arrestate.
Totale	Attività	Attività	os.tasks.total	Il numero totale di attività.
Zombie	Attività	Attività	os.tasks.zombie	Il numero di attività secondarie che sono inattive con un'attività genitore attiva.

Contatore	Tipo	Unità	Parametro	Descrizione
One	Media carico al minuto	Processes	così. loadAverageMinute.uno	Il numero di processi che richiedono l'ora della CPU nell'ultimo minuto.
Fifteen	Media carico al minuto	Processes	così. loadAverageMinute. quindici	Il numero di processi che richiedono l'ora della CPU negli ultimi 15 minuti.
Cinque	Media carico al minuto	Processes	così. loadAverageMinute.cinque	Il numero di processi che richiedono l'ora della CPU negli ultimi 5 minuti.
Cached	Swap	Kilobyte	os.swap.cached	La quantità di memoria di scambio, in kilobyte, utilizzata come memoria cache.
Gratuito	Swap	Kilobyte	os.swap.free	La quantità di memoria di scambio libera, in kilobyte.

Contatore	Tipo	Unità	Parametro	Descrizione
In	Swap	Kilobyte	os.swap.in	Quantità di memoria, in kilobyte, scambiata in ingresso nel disco.
Out	Swap	Kilobyte	os.swap.out	Quantità di memoria, in kilobyte, scambiata in uscita dal disco.
Totale	Swap	Kilobyte	os.swap.total	La quantità totale di memoria di scambio disponibile, in kilobyte.
Max Files	File system	File	os.fileSys.maxFiles	Il numero massimo di file che è possibile creare per il sistema di file.
Used Files	File system	File	os.fileSys.usedFiles	Il numero di file audio nel sistema di file.
Used File Percent	File system	File	sistema operativo . Filesys. usedFilePercent	La percentuale di file disponibili in uso.

Contatore	Tipo	Unità	Parametro	Descrizione
Used Percent	File system	Percentuale	os.fileSys.usedPercent	La percentuale dello spazio su disco del sistema di file in uso.
Used	File system	Kilobyte	os.fileSys.used	La quantità totale di spazio su disco utilizzato dai file nel sistema di file, in kilobyte.
Totale	File system	Kilobyte	os.fileSys.total	Il numero totale di spazio su disco disponibile per il sistema di file, in kilobyte.
Rx	Rete	Byte al secondo	os.network.rx	Il numero di bytes ricevuti al secondo.
Tx	Rete	Byte al secondo	os.network.tx	Il numero di bytes caricati al secondo.
Acu Utilization	Generali	Percentuale	os.general.acuUtilization	La percentuale di capacità attuale rispetto alla capacità massima configurata.

Contatore	Tipo	Unità	Parametro	Descrizione
Max Configured Acu	Generali	ACUs	sistema operativo generale. maxConfig uredAcu	La capacità massima configurata dall'utente, in unità di capacità Aurora ()ACUs.
Min Configured Acu	Generali	ACUs	sistema operativo generale. minConfig uredAcu	La capacità minima configura ta dall'utente, in ACUs.
Num VCPUs	Generali	v CPUs	os.general.num VCPUs	Il numero di virtual CPUs (vCPUs) per l'istanza DB.
Serverles s Database Capacity	Generali	ACUs	sistema operativo generale. serverles sDatabase Capacity	La capacità attuale dell'ista nza, in ACUs.

Contatori di Performance Insights per Aurora MySQL

I seguenti contatori del database sono disponibili in Performance Insights per Aurora MySQL.

Argomenti

- Contatori nativi per Aurora MySQL
- Contatori non nativi per Aurora MySQL

Contatori nativi per Aurora MySQL

I parametri nativi sono definiti dal motore del database e non da Amazon Aurora. Le definizioni di questi parametri nativi sono disponibili in Variabili dello stato del server nella documentazione di MySQL.

Contatore	Tipo	Unità	Parametro
Com_analyze	SQL	Query al secondo	db.SQL.Com_analyze
Com_optimize	SQL	Query al secondo	db.SQL.Com_optimize
Com_select	SQL	Query al secondo	db.SQL.Com_select
Innodb_rows_deleted	SQL	Righe al secondo	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	Righe al secondo	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	Righe al secondo	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	Righe al secondo	db.SQL.Innodb_rows_updated
Query	SQL	Query al secondo	db.SQL.Queries
Questions	SQL	Query al secondo	db.SQL.Questions
Select_full_join	SQL	Query al secondo	db.SQL.Select_full_join
Select_full_range_join	SQL	Query al secondo	db.SQL.Select_full_range_join

Contatore	Tipo	Unità	Parametro
Select_range	SQL	Query al secondo	db.SQL.Select_range
Select_range_check	SQL	Query al secondo	db.SQL.Select_range_check
Select_scan	SQL	Query al secondo	db.SQL.Select_scan
Slow_queries	SQL	Query al secondo	db.SQL.Slow_queries
Sort_merge_passes	SQL	Query al secondo	db.SQL.Sort_merge_passes
Sort_range	SQL	Query al secondo	db.SQL.Sort_range
Sort_rows	SQL	Query al secondo	db.SQL.Sort_rows
Sort_scan	SQL	Query al secondo	db.SQL.Sort_scan
Total_query_time	SQL	Millisecondi	db.SQL.Total_query_time
Table_locks_immediate	Locks	Richieste al secondo	db.Locks.Table_locks_immediate
Table_locks_waited	Locks	Richieste al secondo	db.Locks.Table_locks_waited

Contatore	Tipo	Unità	Parametro
Innodb_row_lock_time	Locks	Millisecondi (media)	db.Locks.Innodb_row_lock_time
Aborted_clients	Utenti	Connessioni	db.Users.Aborted_clients
Aborted_connects	Utenti	Connessioni	db.Users.Aborted_connects
Connessioni	Utenti	Connessioni	db.Users.Connections
External_threads_connected	Utenti	Connessioni	db.Users.External_threads_connected
max_connections	Utenti	Connessioni	db.users.max_connections
Threads_connected	Utenti	Connessioni	db.Users.Threads_connected
Threads_created	Utenti	Connessioni	db.Users.Threads_created
Threads_running	Utenti	Connessioni	db.Users.Threads_running
Created_tmp_disk_tables	Temp	Tabelle al secondo	db.Temp.Created_tmp_disk_tables
Created_tmp_tables	Temp	Tabelle al secondo	db.Temp.Created_tmp_tables
Innodb_buffer_pool_pages_data	Cache	Pagine	db.Cache.Innodb_buffer_pool_pages_data

Contatore	Tipo	Unità	Parametro
Innodb_buffer_pool_pages_total	Cache	Pagine	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Cache	Pagine al secondo	db.Cache.Innodb_buffer_pool_read_requests
Innodb_buffer_pool_reads	Cache	Pagine al secondo	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Cache	Tabelle	db.Cache.Opened_tables
Opened_table_definitions	Cache	Tabelle	db.Cache.Opened_table_definitions
Qcache_hits	Cache	Query	db.Cache.Qcache_hits

Contatori non nativi per Aurora MySQL

I parametri contatore non nativi sono contatori definiti da Amazon RDS. Un parametro non nativo può essere un parametro che si ottiene con una query specifica. Un parametro non nativo può essere un parametro derivato, dove vengono utilizzati due o più contatori nativi nei calcoli di rapporti, percentuali di riscontri o latenze.

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
active_transactions	Transa: ni	db.Trar ctions.a tive_tra saction	Le transazio ni attive totali.	`SELECT COUNT(1) AS active_tr ansactions FROM INFOR MATION_SC HEMA.INNO DB_TRX`	
innodb_buffer_pool_hit_rate	Cache	db.Cac innoDB ffer_poo _hit_rat	La percentua le di letture	`100 * innodb_bu ffer_pool _read_req uests /`	

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
			che InnoDB potrebbe soddisfare dal pool di buffer.	(innodb_buffer_pool_read_requests + innodb_buffer_pool_reads)	
innodb_buffer_pool_hits	Cache	Pagine al second	db.Cache.innoDB_buffer_pool_hits	Il numero di letture che InnoDB potrebbe soddisfare dal pool di buffer.	innodb_buffer_pool_read_requests - innodb_buffer_pool_reads

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
innodb_buffer_pool_usage	Cache	Percentuale	db.Cache.innoDB_buffer_pool_usage	La percentuale del pool di buffer di InnoDB che contiene dati (pagine). **Note** Quando si utilizzano tabelle compresse, questo valore può variare. Per ulteriori dettagli, consulta le informazioni su `Innodb_buffer_pool_pages_data` e `Innodb_buffer_pool_pages_total` in Variabili dello stato del server nella documentazione di MySQL.	`Innodb_buffer_pool_pages_data / Innodb_buffer_pool_pages_total * 100.0`

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
innodb_deadlocks	Locks	db.Lock.innodb_deadlocks	Il numero totale di deadlock.	`SELECT COUNT AS innodb_deadlocks FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_deadlocks'`	
innodb_lock_timeouts	Locks	db.Lock.innodb_lock_timeouts	Il numero totale di deadlock scaduti.	`SELECT COUNT AS innodb_lock_timeouts FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_timeouts'`	
innodb_row_lock_waits	Locks	db.Lock.innodb_row_lock_waits	Il numero totale di blocchi alle righe che ha determinato un'attesa.	`SELECT COUNT AS innodb_row_lock_waits FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='lock_row_lock_waits'`	

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
innodb_rows_changed	SQL	db.SQL nodb_r _chang	Il totale delle operazioni delle righe di InnoDB.	db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated	
query_cache_hit_rate	Cache	Percen le	db.Cache.query_cache_hit_rate	La percentuale di riscontri della cache (cache query) del set di risultati MySQL.	Qcache_hits / (QCache_hits + Com_select) * 100
temp_disk_tables_percent	Temp	db.temp emp_di tables_ rcent	La percentuale di tabelle temporanee create su disco dal server durante l'esecuzione delle istruzioni.	(db.Temp.Created_tmp_disk_tables / db.Temp.Created_tmp_tables) * 100	

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
trx_rseg_history_len	Transa: ni	Nessur	db.Transactions.trx_rseg_history_len	L'elenco delle pagine di log degli annullamenti per le transazioni confermate che viene gestito dal sistema di transazioni InnoDB per implementare il controllo della concorrenza tra più versioni. Per ulteriori informazioni sui dettagli dei record dei log degli annullamenti, consulta https://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html nella documentazione di MySQL.	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

Contatori di Performance Insights per Aurora PostgreSQL

I seguenti contatori del database sono disponibili in Performance Insights per Aurora PostgreSQL.

Argomenti

- Contatori nativi per Aurora PostgreSQL
- Contatori non nativi per Aurora PostgreSQL

Contatori nativi per Aurora PostgreSQL

I parametri nativi sono definiti dal motore del database e non da Amazon Aurora. Le definizioni di questi parametri nativi sono riportate in Viewing Statistics nella documentazione di PostgreSQL.

Contatore	Tipo	Unità	Parametro
tup_deleted	SQL	Tuple al secondo	db.SQL.tup_deleted
tup_fetched	SQL	Tuple al secondo	db.SQL.tup_fetched
tup_inserted	SQL	Tuple al secondo	db.SQL.tup_inserted
tup_returned	SQL	Tuple al secondo	db.SQL.tup_returned
tup_updated	SQL	Tuple al secondo	db.SQL.tup_updated
blks_hit	Cache	Blocchi al secondo	db.Cache.blks_hit
buffers_alloc	Cache	Blocchi al secondo	db.Cache.buffers_alloc
buffers_checkpoint	Checkpoint	Blocchi al secondo	db.Checkpoint.buffers_checkpoint
checkpoints_req	Checkpoint	Checkpoint al minuto	db.Checkpoint.checkpoints_req
checkpoint_sync_time	Checkpoint	Millisecondi per checkpoint	db.Checkpoint.checkpoint_sync_time
checkpoints_timed	Checkpoint	Checkpoint al minuto	db.Checkpoint.checkpoints_timed
checkpoint_write_time	Checkpoint	Millisecondi per checkpoint	db.Checkpoint.checkpoint_write_time
maxwritten_clean	Checkpoint	Interruzioni clean lettura in background al minuto	db.Checkpoint.maxwritten_clean
deadlocks	Concorrenza	Deadlock al minuto	db.Concurrency.deadlocks
blk_read_time	I/O	Millisecondi	db.IO.blk_read_time

Contatore	Tipo	Unità	Parametro
blks_read	I/O	Blocchi al secondo	db.IO.blks_read
buffers_backend	I/O	Blocchi al secondo	db.IO.buffers_backend
buffers_backend_fsync	I/O	Blocchi al secondo	db.IO.buffers_backend_fsync
buffers_clean	I/O	Blocchi al secondo	db.IO.buffers_clean
temp_bytes	Temp	Byte al secondo	db.Temp.temp_bytes
temp_files	Temp	File al minuto	db.Temp.temp_files
xact_commit	Transazioni	Commit al secondo	db.Transactions.xact_commit
xact_rollback	Transazioni	Rollback al secondo	db.Transactions.xact_rollback
numbackends	Utente	Connessioni	db.User.numbackends
archived_count	WAL	File al minuto	db.WAL.archived_count

Contatori non nativi per Aurora PostgreSQL

I parametri contatori non nativi sono contatori definiti da Amazon Aurora. Un parametro non nativo può essere un parametro che si ottiene con una query specifica. Un parametro non nativo può essere un parametro derivato, dove vengono utilizzati due o più contatori nativi nei calcoli di rapporti, percentuali di riscontri o latenze.

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
checkpoint_sync_latency	Checkpoint	Millisecondi	db.Checkpoint.checkpoint_sync_latency	La quantità totale di tempo impiegato nella parte dell'elaborazione dei checkpoint dove i file vengono sincronizzati sul disco.	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	Checkpoint	Millisecondi	db.Checkpoint.checkpoint_write_latency	La quantità totale di tempo impiegato nella parte dell'elaborazione dei checkpoint dove i file vengono scritti sul disco.	`checkpoint_write_time / (checkpoints_timed + checkpoints_req)`
local_blks_read	I/O	Blocchi	db.IO.local_blks_read	Numero totale di blocchi locali letti.	Non applicabile
local_blk_read_time	I/O	Millisecondi	db.IO.local_blk_read_time	Se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato per leggere i blocchi di file di dati locali, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.	Non applicabile

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
num_blocked_sessions	Locks	db.locks.num_BLOCKED_SESSIONI	Il numero di sessioni bloccate.	–	
orcache_blks_hit	I/O	Query	db.IO.orcache_blks_hit	Numero totale di blocchi condivisi provenienti dalla cache delle letture ottimizzate.	Non applicabile
orcache_blk_read_time	I/O	Millisecondi	db.IO.orcache_blk_read_time	Se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato per leggere i blocchi di file di dati dalla cache delle letture ottimizzate, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.	Non applicabile
read_latency	I/O	Millisecondi	db.IO.read_latency	Il tempo impiegato per la lettura dei blocchi di file di dati dai back-end in questa istanza.	`blk_read_time / blks_read`
storage_blks_read	I/O	Blocchi	db.IO.storage_blks_read	Numero totale di blocchi condivisi letti dall'archiviazione Aurora.	Non applicabile

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
storage_blk_read_time	I/O	Millisecondi	db.IO.storage_blk_read_time	Se `track_io_timing` è abilitato, tiene traccia del tempo totale, in millisecondi, impiegato per leggere i blocchi di file di dati dall'archiviazione Aurora, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.	Non applicabile
num_blocked_sessions	Locks	db.locks.num_BLOCED_SESSIONI	Il numero di sessioni bloccate.	–	
active_count	Stato	Sessioni	db.state.active_count	Il numero di sessioni nello stato. `active`	Non applicabile
idle_count	Stato	Sessioni	db.state.idle_count	Il numero di sessioni nello stato. `idle`	Non applicabile
idle_in_transaction_aborted_count	Stato	Sessioni	db.state.idle_in_transaction_aborted_count	Il numero di sessioni nello stato. `idle in transaction (aborted)`	Non applicabile

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
idle_in_transaction_count	Stato	Sessioni	db.state.idle_in_transaction_count	Il numero di sessioni nello stato. `idle in transaction`	Non applicabile
idle_in_transaction_max_time	Stato	Secondi	db.state.idle_in_transaction_max_time	La durata della transazione in esecuzione più lunga nello stato, in secondi. `idle in transaction`	Non applicabile
logical_reads	SQL	Blocchi	db.SQL.logical_reads	Il numero totale di occorrenze e letture di blocchi.	`blks_hit + blks_read`
queries_started	SQL	Query	db.SQL.queries	Il numero di interrogazioni avviate.	Non applicabile
interrogazioni_terminate	SQL	Query	db.SQL.queries	Il numero di interrogazioni completate.	Non applicabile
total_query_time	SQL	Millisecondi	db.SQL.total_query_time	Il tempo totale impiegato per l'esecuzione delle istruzioni, in millisecondi.	Non applicabile
active_transactions	Transazioni	Transazioni	db.Transactions.active_transactions	Il numero di transazioni attive.	Non applicabile

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
blocked_transactions	Transazioni	Transazioni	db.Transactions.blocked_transactions	Il numero di transazioni bloccate.	Non applicabile
commit_latency	Transazioni	Microsecondi	db.Transactions.commit_latency	La durata media delle operazioni di commit.	`db.Transactions.duration_commits / db.Transactions.xact_commit`
duration_commits	Transazioni	Millisecondi	db.Transactions.duration_commits	Il tempo totale di transazione trascorso nell'ultimo minuto, in millisecondi.	Non applicabile
max_used_xact_ids	Transazioni	Transazioni	db.Transactions.max_used_xact_ids	Il numero di transazioni che non sono state cancellate.	Non applicabile
oldest_inactive_logical_replication_slot_xid_age	Transazioni	Lunghezza	db.transactions.oldest_inactive_logical_replication_slot_xid_age	L'età della transazione più vecchia in uno slot di replica logica inattivo.	Non applicabile

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
oldest_active_logical_replication_slot_xid_age	Transazioni	Lunghezza	db.transactions.oldest_active_logical_replication_slot_xid_age	L'età della transazione più vecchia in uno slot di replica logica attivo.	Non applicabile
oldest_reader_feedback_xid_age	Transazioni	Lunghezza	db.transactions.oldest_reader_feedback_xid_age	L'età della transazione più vecchia di una transazione di lunga durata su un'istanza di lettura Aurora o un'istanza di lettore DB globale di Aurora.	Non applicabile
oldest_prepared_transaction_xid_age	Transazioni	Lunghezza	db.transactions.oldest_prepared_transaction_xid_age	L'età della transazione preparata più vecchia.	Non applicabile
oldest_running_transaction_xid_age	Transazioni	Lunghezza	db.transactions.oldest_running_transaction_xid_age	L'età della transazione più vecchia in esecuzione.	Non applicabile

Contatore	Tipo	Unità	Parametro	Descrizione	Definizione
max_connections	Utenti	Utenti	db.User.max_connections	Il numero massimo di connessioni consentite per un database, come configurato nel `max_connections` parametro.	Non applicabile
total_auth_attempts	Utenti	Utenti	db.User.total_auth_attempts	Il numero di tentativi di connessione a questa istanza.	Non applicabile
archive_failed_count	WAL	File al minuto	db.WAL.archive_failed_count	Il numero di tentativi falliti di archiviazione dei file WAL, in file al minuto.	Non applicabile

Statistiche SQL per Performance Insights

Le statistiche SQL sono parametri relativi alle prestazioni delle query SQL raccolti da Performance Insights. Performance Insights raccoglie statistiche durante ogni secondo in cui è in esecuzione una query e per ogni chiamata SQL. Le statistiche SQL sono una media per l'intervallo di tempo selezionato.

Un SQL Digest è un composito di tutte le query con un determinato modello ma che non hanno necessariamente gli stessi valori letterali. Il digest sostituisce i valori letterali con un punto interrogativo. Ad esempio `SELECT * FROM emp WHERE lname= ?`. Questo digest può essere costituito dalle seguenti query figlio:

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Tutti i motori supportano le statistiche SQL delle query a livello di digest.

Per informazioni sull'assistenza alla regione, al motore di database e alla classe di istanza per questa funzionalità, consulta Supporto di classe di istanza, regione e motore di database Amazon Aurora per funzionalità Performance Insights

Argomenti

- Statistiche SQL per Aurora MySQL
- Statistiche SQL per Aurora PostgreSQL

Statistiche SQL per Aurora MySQL

Aurora MySQL raccolgono le statistiche SQL solo a livello di digest. Nessuna statistica viene mostrata a livello di istruzione.

Argomenti

- Statistiche digest per Aurora MySQL
- Statistiche per seconod per Aurora MySQL
- Statistiche per chiamata per Aurora MySQL
- Statistiche primarie per

Statistiche digest per Aurora MySQL

Performance Insights raccoglie statistiche digest SQL dalla tabella `events_statements_summary_by_digest`. La tabella `events_statements_summary_by_digest` è gestita dal database.

La tabella digest non dispone di una policy di espulsione. Quando la tabella è piena, viene AWS Management Console visualizzato il seguente messaggio:

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

In questa situazione, ,Aurora MySQL non esegue il tracciamento delle query SQL. Per risolvere questo problema, Performance Insights tronca automaticamente la tabella del digest quando sono soddisfatte entrambe le condizioni seguenti:

- La tabella è piena.
- Performance Insights gestisce automaticamente Performance Schema.

  Per la gestione automatica, `performance_schema` deve essere impostato su `0` e la Fonte non deve essere impostata su `user`. Se Performance Insights non gestisce automaticamente lo schemda delle prestazioni, vedi Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

In AWS CLI, controlla l'origine del valore di un parametro eseguendo il describe-db-parameterscomando.

Statistiche per seconod per Aurora MySQL

Le seguenti statistiche SQL sono disponibili per cluster di database Aurora MySQL.

Parametro	Unità
db.sql_tokenized.stats.count_star_per_sec	Chiamate al secondo
db.sql_tokenized.stats.sum_timer_wait_per_sec	Latenza media al secondo (in ms)
db.sql_tokenized.stats.sum_select_full_join_per_sec	Seleziona full join al secondo
db.sql_tokenized.stats.sum_select_range_check_per_sec	Seleziona controllo intervallo al secondo
db.sql_tokenized.stats.sum_select_scan_per_sec	Seleziona scansione al secondo
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	Ordina i pass di unione al secondo
db.sql_tokenized.stats.sum_sort_scan_per_sec	Ordina scansioni al secondo
db.sql_tokenized.stats.sum_sort_range_per_sec	Ordina intervalli al secondo
db.sql_tokenized.stats.sum_sort_rows_per_sec	Ordina righe al secondo

Parametro	Unità
db.sql_tokenized.stats.sum_rows_affected_per_sec	Righe interessate al secondo
db.sql_tokenized.stats.sum_rows_examined_per_sec	Righe esaminate al secondo
db.sql_tokenized.stats.sum_rows_sent_per_sec	Righe inviate al secondo
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	Tabelle disco temporanee create al secondo
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	Tabelle temporanee create al secondo
db.sql_tokenized.stats.sum_lock_time_per_sec	Tempo di blocco al secondo (in ms)

Statistiche per chiamata per Aurora MySQL

I seguenti parametri forniscono le statistiche per chiamata di un'istruzione SQL.

Parametro	Unità
db.sql_tokenized.stats.sum_timer_wait_per_call	Latenza media per chiamata (in ms)
db.sql_tokenized.stats.sum_select_full_join_per_call	Seleziona full join per chiamata
db.sql_tokenized.stats.sum_select_range_check_per_call	Seleziona controllo intervallo per chiamata
db.sql_tokenized.stats.sum_select_scan_per_call	Seleziona scansioni per chiamata
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	Ordina pass di unione per chiamata
db.sql_tokenized.stats.sum_sort_scan_per_call	Ordinare scansioni per chiamata

Parametro	Unità
db.sql_tokenized.stats.sum_sort_range_per_call	Ordina intervalli per chiamata
db.sql_tokenized.stats.sum_sort_rows_per_call	Ordina righe per chiamata
db.sql_tokenized.stats.sum_rows_affected_per_call	Righe interessate per chiamata
db.sql_tokenized.stats.sum_rows_examined_per_call	Righe esaminate per chiamata
db.sql_tokenized.stats.sum_rows_sent_per_call	Righe inviate per chiamata
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	Tabelle disco temporanee create per chiamata
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	Tabelle temporanee create per chiamata
db.sql_tokenized.stats.sum_lock_time_per_call	Tempo di blocco per chiamata (in ms)

Statistiche primarie per

Le seguenti statistiche SQL sono disponibili per cluster di database Aurora MySQL.

Parametro	Unità
db.sql_tokenized.stats.count_star	Calls (Chiamate)
db.sql_tokenized.stats.sum_timer_wait	Tempo di attesa (in ms)
db.sql_tokenized.stats.sum_select_full_join	Seleziona la partecipazione completa
db.sql_tokenized.stats.sum_select_range_check	Seleziona i controlli dell'intervallo
db.sql_tokenized.stats.sum_select_scan	Seleziona le scansioni
db.sql_tokenized.stats.sum_sort_merge_passes	Ordina i passaggi di unione

Parametro	Unità
db.sql_tokenized.stats.sum_sort_scan	Ordina le scansioni
db.sql_tokenized.stats.sum_sort_range	Ordina intervalli
db.sql_tokenized.stats.sum_sort_rows	Ordinare le righe
db.sql_tokenized.stats.sum_rows_affected	Righe interessate
db.sql_tokenized.stats.sum_rows_examined	Righe esaminate
db.sql_tokenized.stats.sum_rows_sent	Righe inviate
db.sql_tokenized.stats.sum_created_tmp_disk_tables	Tabelle temporanee su disco create
db.sql_tokenized.stats.sum_created_tmp_tables	Tabelle temporanee create
db.sql_tokenized.stats.sum_lock_time	Tempo di blocco (in ms)

Statistiche SQL per Aurora PostgreSQL

Per ogni chiamata SQL e per ogni secondo di esecuzione di una query, Performance Insights raccoglie statistiche SQL. Tutti i motori Aurora raccolgono statistiche solo a livello di digest.

Di seguito sono disponibili informazioni sulle statistiche a livello di digest per Aurora PostgreSQL.

Argomenti

- Statistiche digest per Aurora PostgreSQL
- Statistiche digest al secondo per Aurora PostgreSQL
- Statistiche digest per chiamata per Aurora PostgreSQL
-

Statistiche digest per Aurora PostgreSQL

Per visualizzare le statistiche digest SQL è necessario caricare la libreria `pg_stat_statements`. Per i cluster di database Aurora PostgreSQL che sono compatibili con PostgreSQL 10, questa libreria viene caricata per impostazione predefinita. Per i cluster di database Aurora PostgreSQL

che sono compatibili con PostgreSQL 9.6, dovrai abilitare manualmente questa libreria. Per abilitarla manualmente, aggiungere `pg_stat_statements` a `shared_preload_libraries` nel gruppo di parametri database associati all'istanza database. Riavviare quindi l'istanza database. Per ulteriori informazioni, consulta .

> **Note**
>
> Performance Insights può raccogliere statistiche solo per le query non troncate in `pg_stat_activity`. Per impostazione predefinita, i database PostgreSQL troncano le query più lunghe di 1.024 byte. Per aumentare la dimensione della query, modificare il parametro `track_activity_query_size` nel gruppo di parametri database associato all'istanza database. Quando si modifica questo parametro, è necessario riavviare un'istanza database.

Statistiche digest al secondo per Aurora PostgreSQL

Le seguenti statistiche digest SQL sono disponibili per le istanze database Aurora PostgreSQL.

Parametro	Unità
db.sql_tokenized.stats.calls_per_sec	Chiamate al secondo
db.sql_tokenized.stats.rows_per_sec	Righe al secondo
db.sql_tokenized.stats.total_time_per_sec	Media delle esecuzioni attive al secondo (AAE, Average active executions)
db.sql_tokenized.stats.shared_blks_hit_per_sec	Richieste in blocco al secondo
db.sql_tokenized.stats.shared_blks_read_per_sec	Letture in blocco al secondo
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	Blocchi sporchi al secondo
db.sql_tokenized.stats.shared_blks_written_per_sec	Scritture in blocco al secondo

Parametro	Unità
db.sql_tokenized.stats.local_blks_hit_per_sec	Richieste in blocco locale al secondo
db.sql_tokenized.stats.local_blks_read_per_sec	Letture di blocchi locali al secondo
db.sql_tokenized.stats.local_blks_dirtied_per_sec	Blocco locale sporco al secondo
db.sql_tokenized.stats.local_blks_written_per_sec	Scritture di blocchi locali al secondo
db.sql_tokenized.stats.temp_blks_written_per_sec	Scritture temporanee al secondo
db.sql_tokenized.stats.temp_blks_read_per_sec	Letture temporanee al secondo
db.sql_tokenized.stats.blk_read_time_per_sec	Letture medie simultanee al secondo
db.sql_tokenized.stats.blk_write_time_per_sec	Scritture medie simultanee al secondo

Statistiche digest per chiamata per Aurora PostgreSQL

I seguenti parametri forniscono le statistiche per chiamata di un'istruzione SQL.

Parametro	Unità
db.sql_tokenized.stats.rows_per_call	Righe per chiamata
db.sql_tokenized.stats.avg_latency_per_call	Latenza media per chiamata (in ms)
db.sql_tokenized.stats.shared_blks_hit_per_call	Richieste in blocco per chiamata
db.sql_tokenized.stats.shared_blks_read_per_call	Letture in blocco per chiamata
db.sql_tokenized.stats.shared_blks_written_per_call	Scritture in blocco per chiamata

Parametro	Unità
db.sql_tokenized.stats.shared_blks_dirtied_per_call	Blocchi danneggiati per chiamata
db.sql_tokenized.stats.local_blks_hit_per_call	Richieste in blocco locale per chiamata
db.sql_tokenized.stats.local_blks_read_per_call	Letture di blocchi locali per chiamata
db.sql_tokenized.stats.local_blks_dirtied_per_call	Blocco locale sporco per chiamata
db.sql_tokenized.stats.local_blks_written_per_call	Scritture di blocchi locali per chiamata
db.sql_tokenized.stats.temp_blks_written_per_call	Scritture temporanee di blocchi per chiamata
db.sql_tokenized.stats.temp_blks_read_per_call	Letture temporanee di blocchi per chiamata
db.sql_tokenized.stats.blk_read_time_per_call	Tempo di lettura per chiamata (in ms)
db.sql_tokenized.stats.blk_write_time_per_call	Tempo di scrittura per chiamata (in ms)

Le seguenti statistiche SQL sono disponibili per le istanze DB di Aurora PostgreSQL.

Parametro	Unità
db.sql_tokenized.stats.calls	Calls (Chiamate)
db.sql_tokenized.stats.rows	Righe
db.sql_tokenized.stats.total_time	Tempo totale (in ms)
db.sql_tokenized.stats.shared_blks_hit	Blocca accessi
db.sql_tokenized.stats.shared_blks_read	Blocca letture
db.sql_tokenized.stats.shared_blks_dirtied	Blocchi sporchi

Parametro	Unità
db.sql_tokenized.stats.shared_blks_written	Block scrive
db.sql_tokenized.stats.local_blks_hit	Block hit locali
db.sql_tokenized.stats.local_blks_read	Lettura locale dei blocchi
db.sql_tokenized.stats.local_blks_dirtied	Blocchi locali sporchi
db.sql_tokenized.stats.local_blks_written	Scritture a blocchi locali
db.sql_tokenized.stats.temp_blks_written	Scritture temporanee
db.sql_tokenized.stats.temp_blks_read	Letture temporanee
db.sql_tokenized.stats.blk_read_time	Letture simultanee medie (in ms)
db.sql_tokenized.stats.blk_write_time	Scritture simultanee medie (in ms)

Per ulteriori informazioni su questi parametri, consultare pg_stat_statements nella documentazione PostgreSQL.

Metriche OS nel monitoraggio avanzato

Amazon Aurora fornisce parametri in tempo reale per il sistema operativo (OS) su cui viene eseguito il cluster di DB. Aurora fornisce i parametri di Enhanced Monitoring al tuo account Amazon CloudWatch Logs. Le tabelle seguenti elencano i parametri del sistema operativo disponibili utilizzando Amazon CloudWatch Logs.

Argomenti

- Parametri del sistema operativo per Aurora

Parametri del sistema operativo per Aurora

Group (Gruppo)	Parametro	Nome console	Descrizione
`General`	`engine`	Non applicabile	Il motore del database per l'istanza database.
	`instanceID`	Non applicabile	Identificatore istanze DB.
	`instanceResourceID`	Non applicabile	Un identificatore immutabile per l'istanza DB che è unico per una AWS regione, utilizzato anche come identificatore del flusso di registro.
	`numVCPUs`	Non applicabile	Il numero di elementi virtuali CPUs per l'istanza DB.
	`timestamp`	Non applicabile	L'ora in cui sono stati presi i parametri.
	`uptime`	Non applicabile	Il periodo di esecuzione dell'istanza database è stato attivato.
	`version`	Non applicabile	La versione del formato di flusso JSON delle metriche del sistema operativo.
`cpuUtilization`	`guest`	CPUOspite	La percentuale di CPU utilizzo da parte dei programmi per gli ospiti.
	`idle`	CPUInattivo	La percentuale è CPU inattiva.
	`irq`	CPU IRQ	La percentuale di CPU utilizzo da parte delle interruzioni del software.
	`nice`	CPUBello	La percentuale di CPU utilizzo da parte dei programmi eseguiti con la priorità più bassa.

Group (Gruppo)	Parametro	Nome console	Descrizione
	`steal`	CPURuba	La percentuale di CPU utilizzo da parte di altre macchine virtuali.
	`system`	CPUSistema	La percentuale di CPU utilizzo da parte del kernel.
	`total`	CPUTotale	La percentuale totale di quella CPU in uso. Questo valore include il valore `nice`.
	`user`	CPUUtente	La percentuale CPU di utilizzo dei programmi utente.
	`wait`	CPUAttendi	La percentuale di dati CPU inutilizzati durante l'attesa dell'accesso I/O.
`diskIO`	`avgQueueLen`	Dimensione media coda	Il numero di richieste in attesa nella coda del dispositivo I/O.
	`avgReqSz`	Dimensione richiesta media	La dimensione della richiesta media, in kilobyte.
	`await`	I/O su disco in attesa	Il numero di millisecondi necessari per rispondere alle richieste, compreso il tempo della coda e il tempo del servizio.
	`device`	Non applicabile	L'identificatore del dispositivo del disco in uso.
	`readIOsPS`	IO/s di lettura	Il numero di operazioni di lettura al secondo.
	`readKb`	Totale lettura	Il numero totale di kilobyte letti.

Group (Gruppo)	Parametro	Nome console	Descrizione
	`readKbPS`	KB/s di lettura	Il numero di kilobytes letti al secondo.
	`readLatency`	Latenza di lettura	Il tempo trascorso tra l'invio di una richiesta di I/O di lettura e il relativo completamento, espresso in millisecondi. Questo parametro è disponibile solo per Amazon Aurora.
	`readThroughput`	Throughput di lettura	La quantità di velocità effettiva di rete utilizzata dalle richieste al cluster di database, espressa in byte al secondo. Questo parametro è disponibile solo per Amazon Aurora.
	`rrqmPS`	Rqms	Il numero di richieste di lettura unite in coda al secondo.
	`tps`	TPS	Il numero di transazioni I/O al secondo.
	`util`	Util I/O su disco	La percentuale di CPU tempo durante la quale sono state emesse le richieste.
	`writeIOsPS`	IO/s di scrittura	Il numero di operazioni di scrittura al secondo.
	`writeKb`	Totale scrittura	Il numero totale di kilobyte scritti.
	`writeKbPS`	KB/s di scrittura	Il numero di kilobytes scritti al secondo.

Group (Gruppo)	Parametro	Nome console	Descrizione
	`writeLatency`	Latenza di scrittura	Tempo medio trascorso tra l'invio di una richiesta di I/O di scrittura e il relativo completamento, in millisecondi. Questo parametro è disponibile solo per Amazon Aurora.
	`writeThroughput`	Throughput di scrittura	La quantità di throughput di rete effettivo utilizzato dalle risposte del cluster di database, espressa in byte al secondo. Questo parametro è disponibile solo per Amazon Aurora.
	`wrqmPS`	Wrqms	Il numero di richieste di scrittura unite in coda al secondo.
`fileSys`	`maxFiles`	Inode max	Il numero massimo di file che è possibile creare per il sistema di file.
	`mountPoint`	Non applicabile	Il percorso del sistema di file.
	`name`	Non applicabile	Il nome del sistema di file.
	`total`	File system totale	Il numero totale di spazio su disco disponibile per il sistema di file, in kilobyte.
	`used`	Filesystem usato	La quantità totale di spazio su disco utilizzato dai file nel sistema di file, in kilobyte.
	`usedFilePercent`	Inode usati	La percentuale di file disponibili in uso.
	`usedFiles`	% di utilizzo	Il numero di file audio nel sistema di file.

Group (Gruppo)	Parametro	Nome console	Descrizione
	`usedPercent`	Filesystem usato	La percentuale dello spazio su disco del sistema di file in uso.
`loadAverageMinute`	`fifteen`	Carico medio 15 min	Il numero di processi che hanno richiesto il CPU tempo negli ultimi 15 minuti.
	`five`	Carico medio 5 min	Il numero di processi che hanno richiesto il CPU tempo negli ultimi 5 minuti.
	`one`	Carico medio 1 min	Il numero di processi che hanno richiesto il CPU tempo nell'ultimo minuto.
`memory`	`active`	Memoria attiva	La quantità di memoria assegnata, in kilobyte.
	`buffers`	Memoria bufferizzata	La quantità di memoria utilizzata per il buffering delle richieste di I/O prima della scrittura sul dispositivo di storage, in kilobyte.
	`cached`	Memoria cache	La quantità di memoria utilizzata per la memorizzazione nella cache dell'I/O basato sul file system.
	`dirty`	Memoria sporca	La quantità di pagine di memoria in RAM esse contenute sono state modificate ma non scritte nel relativo blocco di dati presente nella memoria, espressa in kilobyte.
	`free`	Memoria libera	La quantità di memoria non assegnata, in kilobyte.
	`hugePagesFree`	Huge Pages libere	Il numero di pagine di grandi dimensioni gratuite. Le pagine di grandi dimensioni sono una caratteristica del kernel di Linux.

Group (Gruppo)	Parametro	Nome console	Descrizione
	`hugePagesRsvd`	Huge Pages Rsvd	Il numero di pagine di grandi dimensioni impegnate.
	`hugePagesSize`	Dimensioni Huge Pages	La dimensione per ogni unità delle pagine di grandi dimensioni, in kilobyte.
	`hugePagesSurp`	Huge Pages Surp	Il numero di pagine di grandi dimensioni in eccesso disponibili sul totale.
	`hugePagesTotal`	Totale Huge Pages	Il numero totale di Huge Pages.
	`inactive`	Memoria inattiva	La quantità di pagine di memoria utilizzate meno frequentemente, in kilobyte.
	`mapped`	Memoria mappata	La quantità totale di contenuti del sistema di file che è mappata in memoria all'interno di uno spazio di indirizzamento del processo, in kilobyte.
	`pageTables`	Tabelle delle pagine	La quantità di memoria utilizzata dalle tabelle della pagina, in kilobyte.
	`slab`	Memoria slab	La quantità di strutture dati del kernel riutilizzabili, in kilobyte.
	`total`	Memoria totale	La quantità totale di memoria, in kilobyte.
	`writeback`	Memoria writeback	La quantità di pagine RAM sporche che contiene viene ancora scritta nello storage di supporto, espressa in kilobyte.

Group (Gruppo)	Parametro	Nome console	Descrizione
network	interface	Non applicabile	L'identificatore per l'interfaccia di rete utilizzato per l'istanza database.
	rx	RX	Il numero di bytes ricevuti al secondo.
	tx	TX	Il numero di bytes caricati al secondo.
processList	cpuUsedPc	CPU %	La percentuale di dati CPU utilizzati dal processo.
	id	Non applicabile	L'identificatore del processo.
	memoryUsedPc	MEM%	Percentuale della memoria totale utilizzata dal processo.
	name	Non applicabile	Il nome del processo.
	parentID	Non applicabile	L'identificatore del processo per il processo genitore del processo.
	rss	RES	La quantità di dati RAM allocati al processo, espressa in kilobyte.
	tgid	Non applicabile	L'identificatore del gruppo di thread, che è un numero che rappresenta l'ID del processo a cui appartiene un thread. Questo identificatore è utilizzato per raggruppare i thread dallo stesso processo.
	vss	VIRT	La quantità di memoria virtuale allocata al processo, in kilobyte.
swap	total	Swap	La quantità di memoria di scambio disponibile, in kilobyte.

Group (Gruppo)	Parametro	Nome console	Descrizione
	`in`	Swap in	Quantità di memoria, in kilobyte, scambiata in ingresso nel disco.
	`out`	Swap out	Quantità di memoria, in kilobyte, scambiata in uscita dal disco.
	`free`	Swap libera	La quantità di memoria di scambio libera, in kilobyte.
	`cached`	Swap occupata	La quantità di memoria di scambio, in kilobyte, utilizzata come memoria cache.
`tasks`	`blocked`	Attività bloccate	Il numero di attività che sono bloccate.
	`running`	Attività in esecuzione	Il numero di attività che sono in esecuzione.
	`sleeping`	Attività inattive	Il numero di attività che sono a riposo.
	`stopped`	Attività interrotte	Il numero di attività che sono arrestate.
	`total`	Totale attività	Il numero totale di attività.
	`zombie`	Attività Zombie	Il numero di attività secondarie che sono inattive con un'attività genitore attiva.

Quando monitori i tuoi database Amazon Aurora e le tue altre AWS soluzioni, il tuo obiettivo è mantenere quanto segue:

- Affidabilità
- Disponibilità
- Prestazioni
- Sicurezza

Monitoraggio dei parametri in un cluster di database Amazon Aurora spiega come monitorare il cluster usando i parametri. Una soluzione completa deve inoltre monitorare gli eventi del database, i file di registro e i flussi di attività. AWS fornisce i seguenti strumenti di monitoraggio:

- Amazon EventBridge è un servizio di bus eventi senza server che semplifica la connessione delle applicazioni con dati provenienti da una varietà di fonti. EventBridge offre un flusso di dati in tempo reale dalle tue applicazioni, applicazioni oftware-as-a S-Service (SaaS) e servizi. AWS EventBridge indirizza tali dati verso obiettivi come. AWS Lambda In questo modo, puoi monitorare gli eventi che si verificano nei servizi e creare architetture basate su eventi. Per ulteriori informazioni, consulta la Amazon EventBridge User Guide.
- Amazon CloudWatch Logs offre un modo per monitorare, archiviare e accedere ai file di log da istanze Aurora e altre AWS CloudTrail fonti. Amazon CloudWatch Logs può monitorare le informazioni nei file di registro e avvisarti quando vengono raggiunte determinate soglie. Puoi inoltre archiviare i dati del log in storage estremamente durevole. Per ulteriori informazioni, consulta la Amazon CloudWatch Logs User Guide.
- AWS CloudTrailregistra le API chiamate e gli eventi correlati effettuati da o per conto del tuo. Account AWS CloudTrail consegna i file di log a un bucket Amazon S3 specificato dall'utente. Puoi identificare quali utenti e account hanno effettuato le chiamate AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute le chiamate. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudTrail.
- Database Activity Streams Amazon Aurora trasferisce le attività a un flusso di dati Amazon Kinesis. Il flusso Kinesis viene creato automaticamente. Da Kinesis, puoi configurare AWS servizi come Amazon Data Firehose e consumare lo stream e AWS Lambda archiviare i dati.

Argomenti

- Visualizzazione di log, eventi e stream nella console Amazon RDS

-
- Monitoraggio dei file di log di Aurora
- Monitoraggio delle chiamate API Aurora in AWS CloudTrail
- Monitoraggio di Amazon Aurora tramite i flussi di attività del database
- Monitoraggio delle minacce con Amazon GuardDuty RDS Protection per Amazon Aurora

Visualizzazione di log, eventi e stream nella console Amazon RDS

Amazon RDS si integra con Servizi AWS per mostrare informazioni su log, eventi e flussi di attività del database nella console. RDS

La scheda Registri ed eventi per il cluster di istanze Aurora mostra le seguenti informazioni:

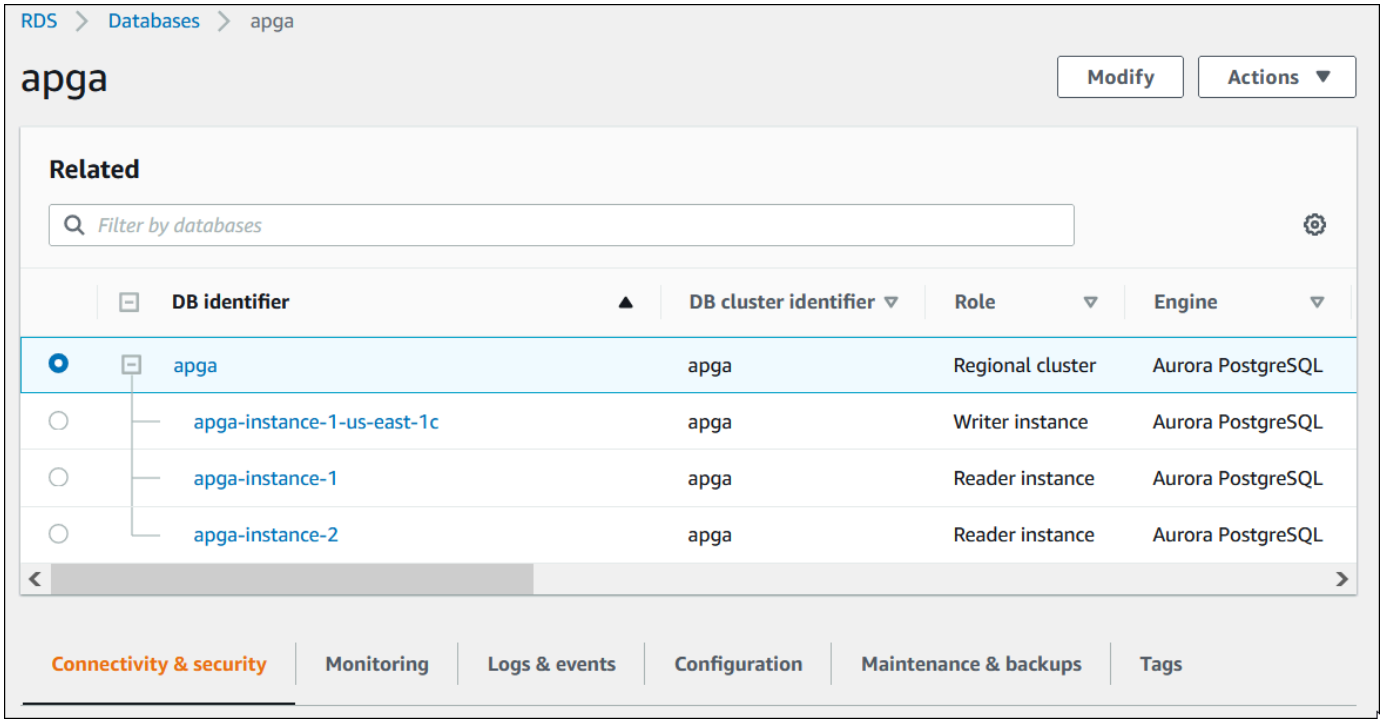
- Auto scaling policies and activities (Policy e attività di scalabilità automatica): mostra le policy e le attività relative alla funzione di scalabilità automatica di Aurora. Queste informazioni vengono visualizzate solo nella scheda Logs & events (Registri ed eventi) a livello di cluster.
- Amazon CloudWatch alarms: mostra tutti gli allarmi metrici che hai configurato per l'istanza DB nel tuo cluster Aurora. Se non hai configurato gli allarmi, puoi crearli nella console. RDS
- Eventi recenti: mostra un riepilogo degli eventi (modifiche all'ambiente) per l'istanza o il cluster di Aurora DB. Per ulteriori informazioni, consulta Visualizzazione degli RDS eventi Amazon.
- Log (Registro): mostra i file di log del database generati da un'istanza database nel cluster Aurora. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Aurora.

La scheda Configuration (Configurazione) mostra le informazioni sui flussi di attività di database.

Per visualizzare log, eventi e stream per il cluster di Aurora DB nella console RDS

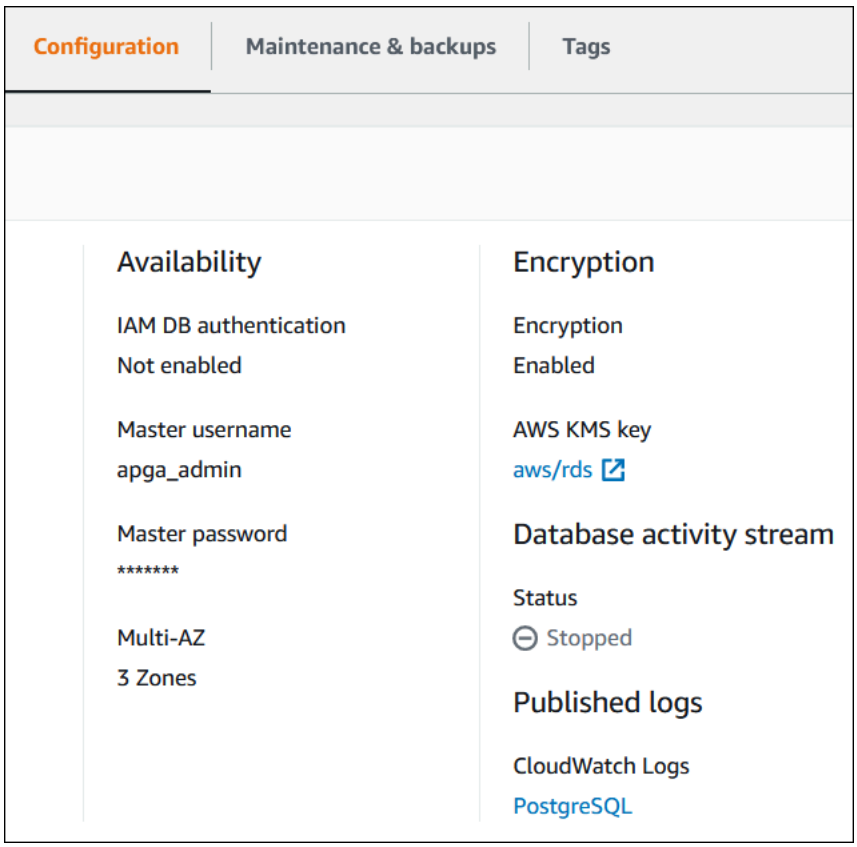
1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome del cluster di Aurora che si desidera monitorare.

   Verrà visualizzata la pagina Databases (Database). L'esempio seguente mostra un Amazon Aurora Postgre SQL DB denominato. `apga`



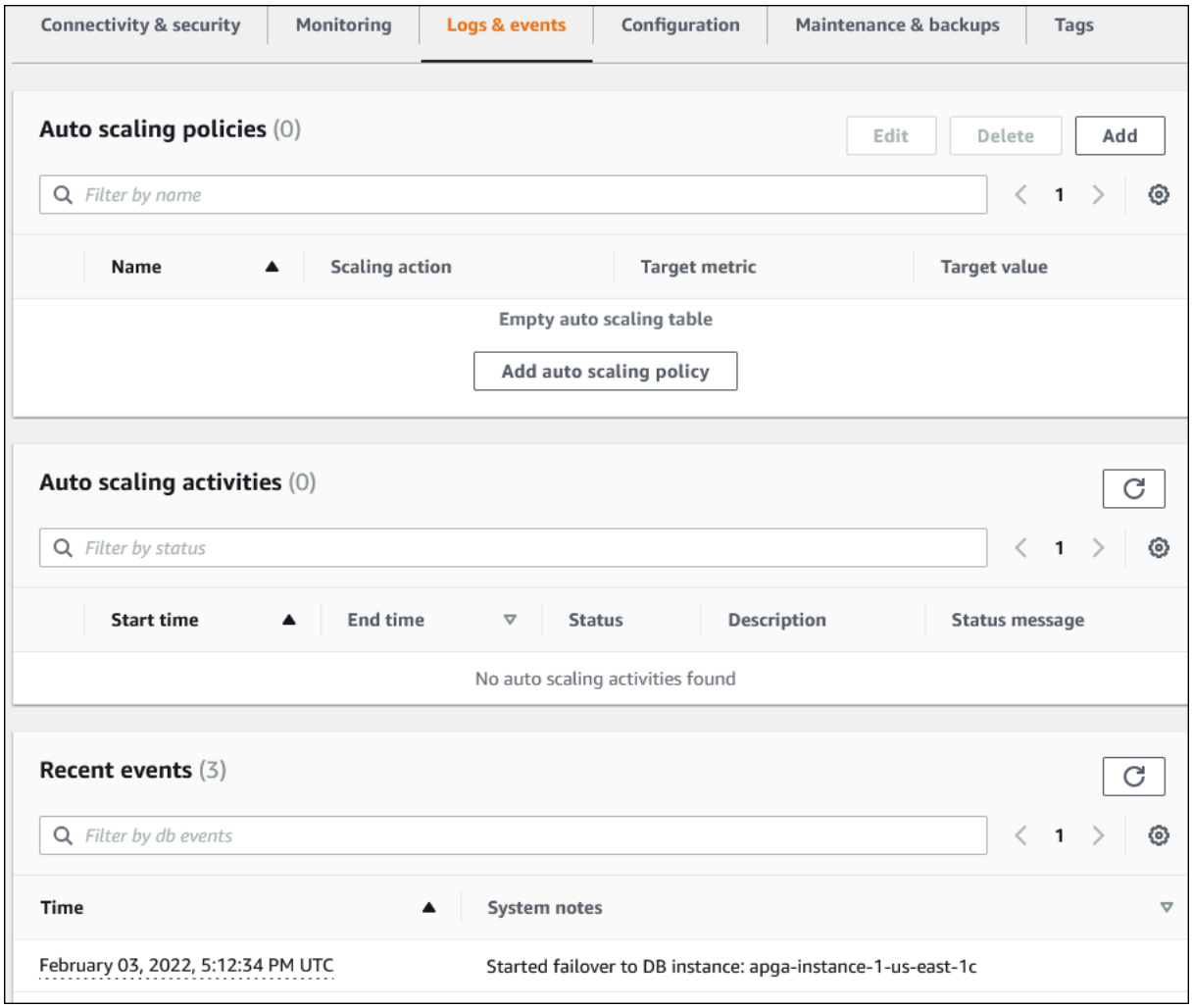
4. Scorrere verso il basso e selezionare Configuration (Configurazione).

   L'esempio seguente mostra lo stato dei flussi di attività di database per il cluster.



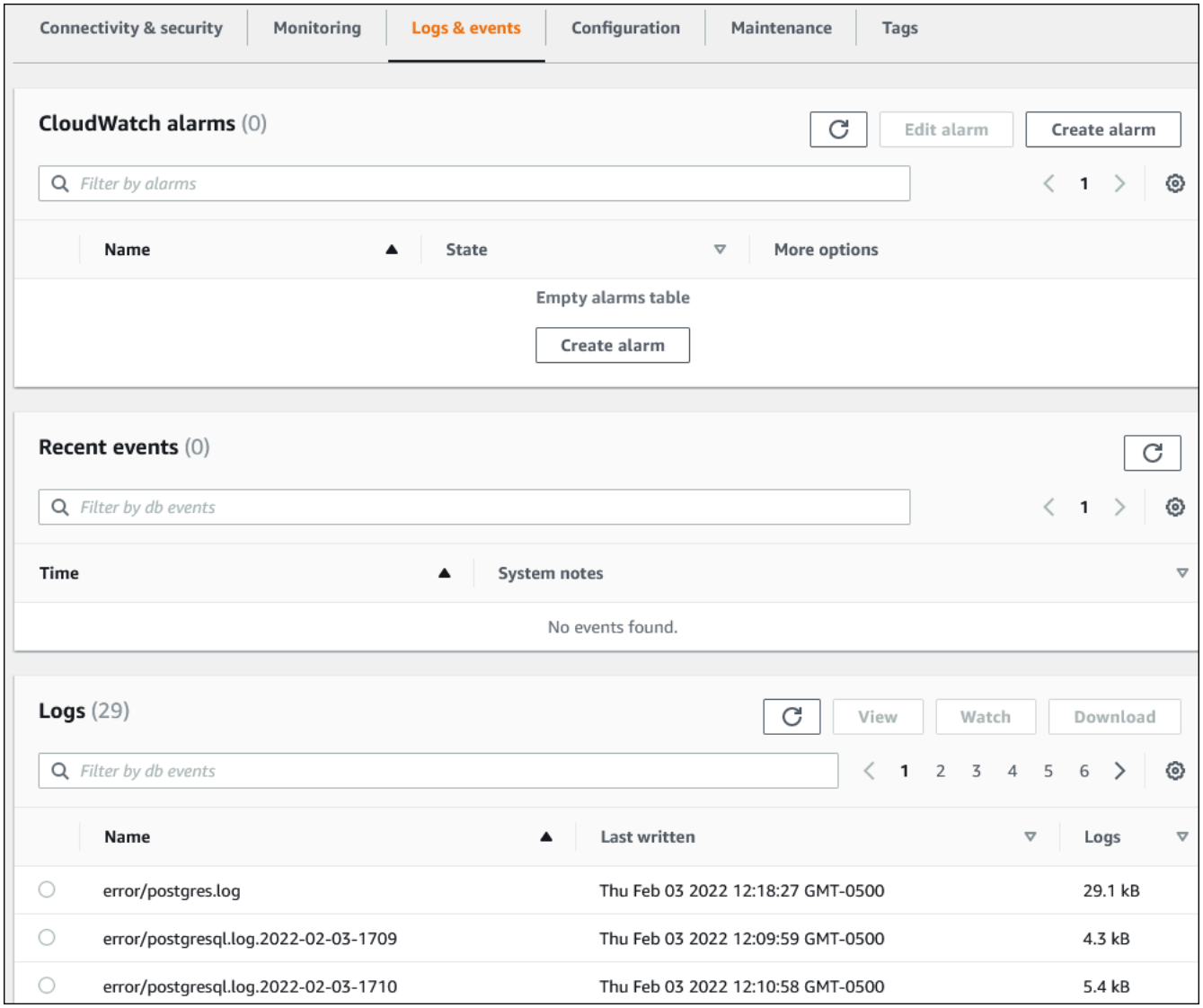
5. Scegliere Logs & events (Log ed eventi).

   Viene visualizzata la sezione Logs & events (Log ed eventi).



6. Scegliere un'istanza database nel cluster Aurora e quindi scegliere Logs & events (Registri ed eventi) per l'istanza.

   L'esempio seguente mostra che il contenuto è diverso tra la pagina dell'istanza database e la pagina del cluster di database. La pagina dell'istanza database mostra i registri e gli allarmi.


Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance
Tags
CloudWatch alarms (0)
Edit alarm
Create alarm
Filter by alarms
1
Name
State
More options
Empty alarms table
Create alarm
Recent events (0)
Filter by db events
1
Time
System notes
No events found.
Logs (29)
View
Watch
Download
Filter by db events
1 2 3 4 5 6
Name
Last written
Logs
error/postgres.log
Thu Feb 03 2022 12:18:27 GMT-0500
29.1 kB
error/postgresql.log.2022-02-03-1709
Thu Feb 03 2022 12:09:59 GMT-0500
4.3 kB
error/postgresql.log.2022-02-03-1710
Thu Feb 03 2022 12:10:58 GMT-0500
5.4 kB

Un evento indica una modifica in un ambiente. Può trattarsi di un AWS ambiente, un servizio o un'applicazione partner SaaS oppure un'applicazione o un servizio personalizzato. Per le descrizioni degli eventi di Aurora, vedere. Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora

Argomenti

-
- Visualizzazione degli RDS eventi Amazon
- Utilizzo delle notifiche di RDS eventi di Amazon
- Creazione di una regola che si attiva su un evento Amazon Aurora
- Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora

Un RDSevento indica un cambiamento nell'ambiente Aurora. Ad esempio, Aurora genera un evento quando viene applicata una patch a un cluster DB, . Amazon Aurora offre eventi quasi EventBridge in tempo reale.

> **Note**
>
> Amazon RDS emette eventi con la massima diligenza possibile. Si consiglia di non scrivere programmi che dipendono dall'ordine o dall'esistenza di eventi di notifica, poiché potrebbero essere fuori sequenza o mancanti.

Amazon RDS registra gli eventi relativi alle seguenti risorse:

- Cluster database

  Per un elenco degli eventi di un cluster, consulta Eventi di cluster di database.
- Istanze DB

  Per un elenco degli eventi dell'istanza database, consulta Eventi di istanza database.
- Gruppi di parametri database

  Per un elenco degli eventi del gruppo parametri del database, consulta Eventi gruppo di parametri database.
- Gruppi di sicurezza DB

Per un elenco di eventi del gruppo di sicurezza DB, consulta Eventi gruppo di sicurezza DB.

- Snapshot cluster DB

  Per un elenco di eventi degli snapshot del cluster database, consulta Eventi snapshot cluster di database.

- RDSEventi proxy

  Per un elenco degli eventi RDS Proxy, vedereEventi RDS Proxy.

- Eventi di implementazione blu/verde

  Per l'elenco degli eventi di implementazione blu/verde, consulta Eventi di implementazione blu/verde.

Queste informazioni comprendono:

- La data e l'ora dell'evento
- Il nome di origine e il tipo di origine dell'evento
- Un messaggio associato all'evento
- Le notifiche degli eventi includono i tag che fanno riferimento al momento in cui il messaggio è stato inviato e potrebbero non riflettere i tag riferiti al momento in cui si è verificato l'evento.

Visualizzazione degli RDS eventi Amazon

Puoi recuperare le seguenti informazioni sugli eventi per le tue risorse Aurora:

- Nome risorsa
- Tipo di risorsa
- Ora dell'evento
- Riepilogo del messaggio dell'evento

Puoi accedere agli eventi nelle seguenti parti di: AWS Management Console

- La scheda Eventi, che mostra gli eventi delle ultime 24 ore.
- La tabella Eventi recenti nella sezione Registri ed eventi della scheda Database, che può mostrare gli eventi relativi alle ultime 2 settimane.

È inoltre possibile recuperare gli eventi utilizzando il AWS CLI comando describe-events o l'operazione. DescribeEventsRDSAPI Se si utilizza AWS CLI o il RDS API per visualizzare gli eventi, è possibile recuperare gli eventi relativi agli ultimi 14 giorni.

> **Note**
>
> Se devi archiviare eventi per periodi di tempo più lunghi, puoi inviare RDS eventi Amazon a EventBridge. Per ulteriori informazioni, consulta Creazione di una regola che si attiva su un evento Amazon Aurora

Per le descrizioni degli eventi di Amazon Aurora, consulta. Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora

Per accedere a informazioni dettagliate sull'utilizzo degli eventi AWS CloudTrail, inclusi i parametri di richiesta, consultaCloudTrail eventi.

Console

Per visualizzare tutti RDS gli eventi Amazon delle ultime 24 ore

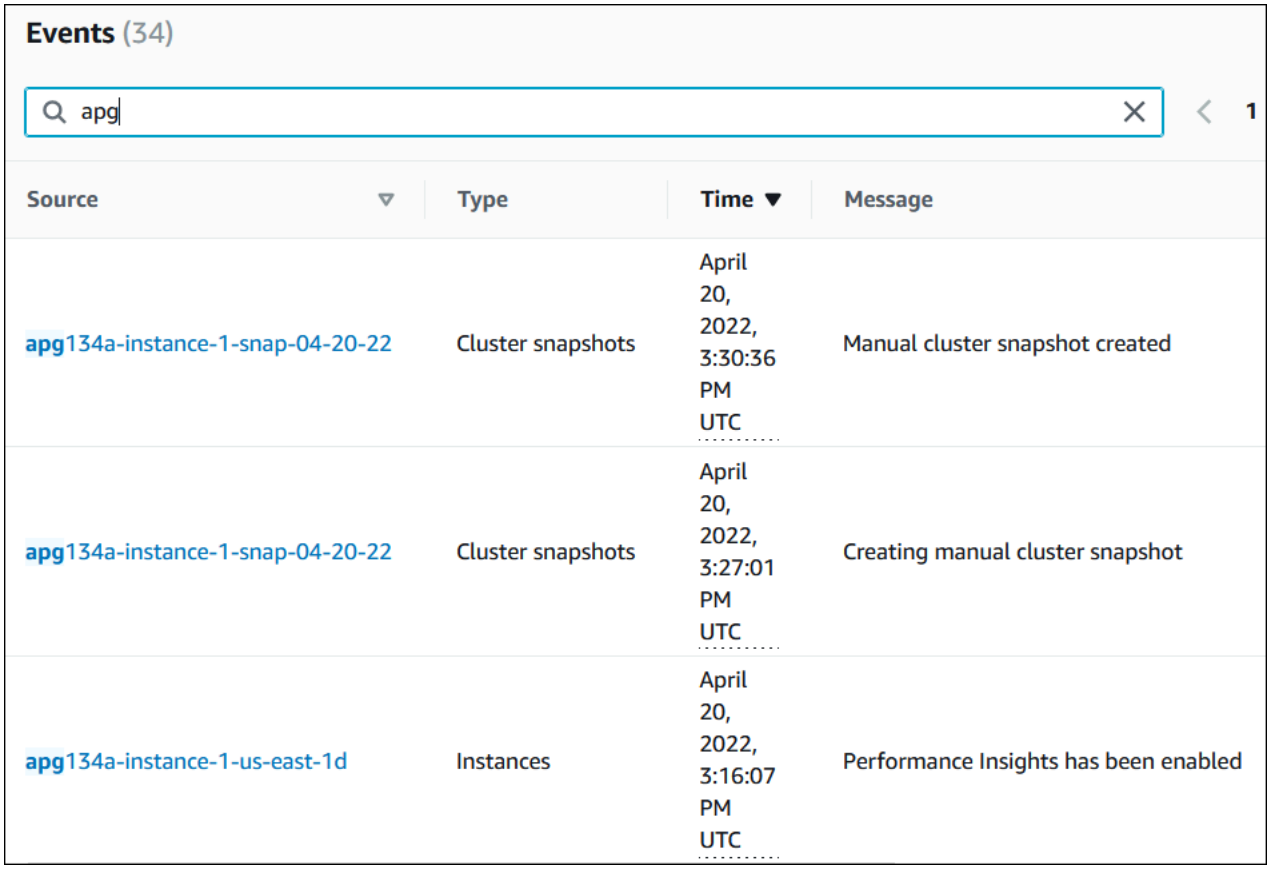
1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione selezionare Events (Eventi).

   Gli eventi disponibili sono indicati all'interno di un elenco.

3. (Opzionale) Inserisci un termine di ricerca per filtrare i risultati.

   Il seguente esempio mostra un elenco di eventi filtrati mediante i caratteri **apg**.



AWS CLI

Per visualizzare tutti gli eventi generati nell'ultima ora, invoca describe-events senza parametri.

```
aws rds describe-events
```

Il seguente output di esempio mostra che un'istanza del cluster database ha avviato il ripristino.

```
{
    "Events": [
        {
            "EventCategories": [
                "recovery"
```

```
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mycluster-instance-1",
            "Date": "2022-04-20T15:02:38.416Z",
            "Message": "Recovery of the DB instance has started. Recovery time will
 vary with the amount of data to be recovered.",
            "SourceIdentifier": "mycluster-instance-1"
        }, ...
```

Per visualizzare tutti RDS gli eventi Amazon degli ultimi 10080 minuti (7 giorni), chiama il AWS CLI comando describe-events e imposta il parametro su. `--duration 10080`

```
aws rds describe-events --duration 10080
```

L'esempio seguente mostra gli eventi nell'intervallo di tempo specificato per l'istanza DB *`test-instance`*.

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

L'output di esempio seguente mostra lo stato di un backup.

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
```

```
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

API

Puoi visualizzare tutti gli eventi delle RDS istanze Amazon degli ultimi 14 giorni chiamando l'DescribeEventsRDSAPIoperazione e impostando il `Duration` parametro su`20160`.

Utilizzo delle notifiche di RDS eventi di Amazon

Amazon RDS utilizza Amazon Simple Notification Service (AmazonSNS) per fornire notifiche quando si verifica un RDS evento Amazon. Queste notifiche possono essere contenute in qualsiasi modulo di notifica supportato da Amazon SNS per una AWS regione, ad esempio un'e-mail, un messaggio di testo o una chiamata a un HTTP endpoint.

Argomenti

- Panoramica della notifica degli RDS eventi di Amazon
- Concessione di autorizzazioni per pubblicare le notifiche in un argomento Amazon SNS
- Iscrizione alla notifica RDS degli eventi di Amazon
- Tag e attributi di notifica degli RDS eventi di Amazon
- Pubblicazione degli abbonamenti Amazon per le notifiche di RDS eventi
- Modifica di un abbonamento per la notifica di RDS eventi Amazon
- Aggiungere un identificatore di origine a un abbonamento di notifica di RDS eventi Amazon
- Rimozione di un identificatore di origine da un abbonamento di notifica di RDS eventi Amazon
- Elenco delle categorie di notifiche di RDS eventi di Amazon
- Eliminazione di un abbonamento per la notifica di RDS eventi Amazon

Panoramica della notifica degli RDS eventi di Amazon

Amazon RDS raggruppa gli eventi in categorie a cui puoi iscriverti in modo da ricevere una notifica quando si verifica un evento in quella categoria.

Argomenti

- RDSrisorse idonee per l'abbonamento agli eventi
- Procedura di base per l'iscrizione alle notifiche RDS degli eventi di Amazon
- Invio di notifiche di RDS eventi
- Fatturazione per le notifiche di RDS eventi Amazon
- Esempi di RDS eventi Aurora Amazon che EventBridge

RDSrisorse idonee per l'abbonamento agli eventi

Per Amazon Aurora, gli eventi si verificano sia a livello di cluster di database che di istanza database. È possibile sottoscrivere una categoria di eventi per le seguenti risorse:

- Istanza database
- cluster di database
- Snapshot cluster di database
- DB parameter group (Gruppo di parametri database)
- Gruppo di sicurezza DB
- RDSProxy
- Versioni personalizzate del motore

Ad esempio, sottoscrivendo la categoria Backup per una determinata istanza database, riceverai una notifica ogni volta che si verifica un evento relativo al backup che interessa l'istanza database. Sottoscrivendo una categoria Modifica della configurazione per un'istanza database, riceverai una notifica quando l'istanza database viene modificata. Riceverai una notifica anche quando viene modificata la sottoscrizione a una notifica eventi.

È possibile creare alcuni sottoscrizioni diverse. Per esempio, è possibile creare una sottoscrizione che riceve tutte le notifiche eventi per tutte le istanze database e un'altra sottoscrizione che include solo eventi critici per un sottoinsieme di istanze database. Per la seconda sottoscrizione, specifica una o più istanze database nel filtro.

Procedura di base per l'iscrizione alle notifiche RDS degli eventi di Amazon

La procedura per l'iscrizione alla notifica RDS degli eventi di Amazon è la seguente:

1. Crei un abbonamento Amazon RDS Event Notification utilizzando la RDS console Amazon AWS CLI, oppureAPI.

   Amazon RDS utilizza un SNS argomento Amazon per identificare ogni abbonamento. ARN La RDS console Amazon crea il ARN file per te quando crei l'abbonamento. ARNCrealo utilizzando la SNS console Amazon AWS CLI, o Amazon SNSAPI.
2. Amazon RDS invia un'e-mail o un SMS messaggio di approvazione agli indirizzi che hai inviato con l'abbonamento.
3. Puoi confermare la sottoscrizione selezionando il collegamento nella notifica ricevuta.

4. La RDS console Amazon aggiorna la sezione Abbonamenti ai miei eventi con lo stato dell'abbonamento.
5. Amazon RDS inizia a inviare le notifiche agli indirizzi che hai fornito al momento della creazione dell'abbonamento.

Per ulteriori informazioni sulla gestione delle identità e degli accessi quando usi AmazonSNS, consulta la sezione Gestione delle identità e degli accessi in Amazon SNS nella Amazon Simple Notification Service Developer Guide.

Puoi utilizzarlo AWS Lambda per elaborare notifiche di eventi da un'istanza DB. Per ulteriori informazioni, consulta Using AWS Lambda with Amazon RDS nella AWS Lambda Developer Guide.

Invio di notifiche di RDS eventi

Amazon RDS invia notifiche agli indirizzi che fornisci quando crei l'abbonamento. La notifica può inserire gli attributi del messaggio che forniscono i metadati strutturati sul messaggio. Per ulteriori informazioni sugli attributi del messaggio, consulta Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora.

La distribuzione delle notifiche degli eventi può richiedere fino a cinque minuti.

> **Important**
>
> Amazon RDS non garantisce l'ordine degli eventi inviati in un flusso di eventi. Tale ordine è soggetto a modifiche.

Quando Amazon SNS invia una notifica a un abbonato HTTP o a un HTTPS endpoint, il POST messaggio inviato all'endpoint ha un corpo del messaggio che contiene un documento. JSON Per ulteriori informazioni, consulta i SNSmessaggi e i JSON formati di Amazon nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Puoi configurare la modalità SNS di notifica tramite messaggi di testo. Per ulteriori informazioni, consulta Messaggi di testo mobili (SMS) nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Per disattivare le notifiche senza eliminare un abbonamento, scegli No per Enabled nella RDS console Amazon. Oppure puoi impostare il `Enabled` parametro per `false` utilizzare AWS CLI o Amazon RDSAPI.

Fatturazione per le notifiche di RDS eventi Amazon

La fatturazione per la notifica RDS degli eventi Amazon avviene tramite AmazonSNS. Quando si utilizza la notifica degli eventi, si applicano le SNS tariffe di Amazon. Per ulteriori informazioni sulla SNS fatturazione di Amazon, consulta i prezzi di Amazon Simple Notification Service.

Esempi di RDS eventi Aurora Amazon che EventBridge

Gli esempi seguenti illustrano diversi tipi di JSON eventi Amazon in formato. Per un tutorial che mostra come acquisire e visualizzare gli eventi in JSON formato, consulta. Tutorial: registra le modifiche allo stato delle istanze DB utilizzando Amazon EventBridge

Argomenti

- Esempio di evento cluster di database
- Esempio di evento del gruppo parametri del database
- Esempio di evento snapshot cluster di database

Esempio di evento cluster di database

Di seguito è riportato un esempio di evento del cluster DB in JSON formato. L'evento mostra che al cluster denominato `my-db-cluster` è stata applicata la patch. L'ID evento è `RDS-EVENT-0173`.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster"
  ],
  "detail": {
    "EventCategories": [
      "notification"
    ],
    "SourceType": "CLUSTER",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Database cluster has been patched",
```

```
    "SourceIdentifier": "my-db-cluster",
    "EventID": "RDS-EVENT-0173"
  }
}
```

Esempio di evento del gruppo parametri del database

Di seguito è riportato un esempio di evento DB Parameter Group in JSON formato. L'evento mostra che il parametro `time_zone` è stato aggiornato nel gruppo di parametri `my-db-param-group`. L'ID dell'evento è RDS - EVENT -0037.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

Esempio di evento snapshot cluster di database

Di seguito è riportato un esempio di evento snapshot del cluster DB in JSON formato. L'evento mostra la creazione dello snapshot denominato `my-db-cluster-snapshot`. L'ID dell'evento è RDS - EVENT -0074.

```
{
```

```
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Snapshot Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-cluster-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "backup"
    ],
    "SourceType": "CLUSTER_SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-
cluster-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "SourceIdentifier": "my-db-cluster-snapshot",
    "Message": "Creating manual cluster snapshot",
    "EventID": "RDS-EVENT-0074"
  }
}
```

Concessione di autorizzazioni per pubblicare le notifiche in un argomento Amazon SNS

Per concedere ad Amazon RDS le autorizzazioni per pubblicare notifiche su un argomento di Amazon Simple Notification Service (Amazon SNS), allega AWS Identity and Access Management una policy (IAM) all'argomento di destinazione. Per maggiori informazioni sulle autorizzazioni, consulta Esempi di casi per il controllo degli accessi Amazon SNS nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Per impostazione predefinita, un argomento Amazon SNS dispone di una policy che consente a tutte le risorse Amazon RDS nello stesso account di pubblicare notifiche nello stesso. Puoi collegare una policy personalizzata per consentire notifiche tra account o per limitare l'accesso a determinate risorse.

Di seguito è riportato un esempio di policy IAM collegata all'argomento Amazon SNS di destinazione. Limita l'argomento alle istanze database con nomi corrispondenti al prefisso specificato. Per utilizzare questa policy, specifica i seguenti valori:

- `Resource` – Il nome della risorsa Amazon (ARN) per l'argomento Amazon SNS
- `SourceARN` – L'ARN della risorsa RDS
- `SourceAccount`— Il tuo ID Account AWS

Per visualizzare un elenco dei tipi di risorse e relativi ARNs, consulta Resources Defined by Amazon RDS nel Service Authorization Reference.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
```

```
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Iscrizione alla notifica RDS degli eventi di Amazon

Il modo più semplice per creare un abbonamento è tramite la RDS console. Se scegli di creare abbonamenti per le notifiche di eventi utilizzando CLI oAPI, devi creare un argomento Amazon Simple Notification Service e abbonarti a tale argomento con la SNS console Amazon o Amazon SNSAPI. Dovrai inoltre conservare l'Amazon Resource Name (ARN) dell'argomento perché viene utilizzato per inviare CLI comandi o API operazioni. Per informazioni sulla creazione di un SNS argomento e sulla sottoscrizione, consulta la sezione Guida introduttiva ad Amazon SNS nella Amazon Simple Notification Service Developer Guide.

Puoi specificare il tipo di fonte che desideri ricevere notifiche e la RDS fonte Amazon che attiva l'evento:

Source type (Tipo di origine)

Il tipo di origine Ad esempio: Source Type (Tipo di origine) potrebbe essere Instances (Istanze). Devi scegliere un tipo di origine.

Resources da includere

Le RDS risorse Amazon che generano gli eventi. Ad esempio, puoi scegliere Seleziona istanze specifiche e quindi myDBInstance1.

La tabella seguente spiega il risultato quando si specifica o non si specifica ***Resources*** includere.

Risorse da includere	Descrizione	Esempio
Specificato	RDSnotifica all'utente tutti gli eventi solo per la risorsa specificata.	Se il tipo di origine è Istanze e la risorsa è myDBInstance1, RDS invia notifiche relative a tutti gli eventi solo per myDBInstance 1.
Non specificato	RDSti notifica gli eventi per il tipo di sorgente specificato per tutte le tue RDS risorse Amazon.	Se il tipo di origine è Istanze, ti RDS invia notifiche su tutti gli eventi relativi alle istanze nel tuo account.

Per impostazione predefinita, un abbonato all'SNSargomento di Amazon riceve tutti i messaggi pubblicati sull'argomento. Per ricevere solo un sottoinsieme dei messaggi, l'abbonato deve assegnare una policy di filtro all'abbonamento all'argomento. Per ulteriori informazioni sul filtraggio dei SNS messaggi, consulta Amazon SNS Message Filtering nella Amazon Simple Notification Service Developer Guide

Console

Per iscriverti alla notifica degli eventi RDS

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Event subscriptions (Sottoscrizioni a eventi).
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere Create event subscription (Crea sottoscrizione di eventi).
4. Inserisci i dettagli dell'abbonamento come segue:

   a. Per Name (Nome), immettere un nome per la sottoscrizione alle notifiche eventi.

   b. Nel campo Send notifications to (Invia notifica a), esegui una delle seguenti operazioni:

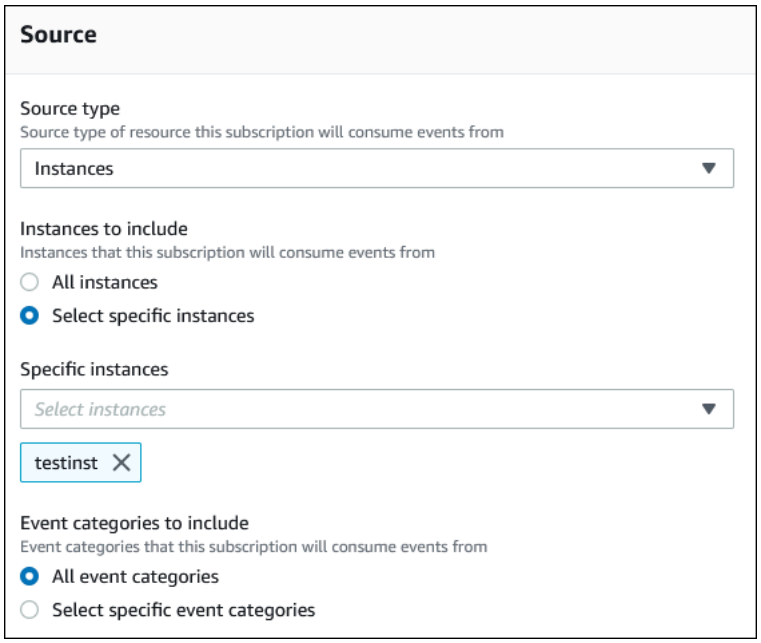
      - Scegli New email topic (Nuovo argomento e-mail). Inserisci un nome per l'argomento dell'email e un elenco di destinatari. Ti consigliamo di configurare le sottoscrizioni agli eventi con lo stesso indirizzo e-mail del contatto dell'account principale. I suggerimenti, gli eventi di assistenza e i messaggi personali vengono inviati utilizzando canali diversi. Le sottoscrizioni con lo stesso indirizzo e-mail assicurano che tutti i messaggi siano consolidati in un'unica posizione.
      - Scegli Amazon Resource Name (ARN). Quindi scegli Amazon esistente SNS ARN per un SNS argomento Amazon.

        Se desideri utilizzare un argomento che è stato abilitato per la crittografia lato server (SSE), concedi ad Amazon RDS le autorizzazioni necessarie per accedere a. AWS KMS key Per ulteriori informazioni, consulta Abilitare la compatibilità tra le fonti di eventi provenienti dai AWS servizi e gli argomenti crittografati nella Amazon Simple Notification Service Developer Guide.

   c. Per Source type (Tipo di origine) scegliere un tipo di origine. Ad esempio, scegli Clusters (Cluster) o Cluster snapshots (Snapshot cluster).

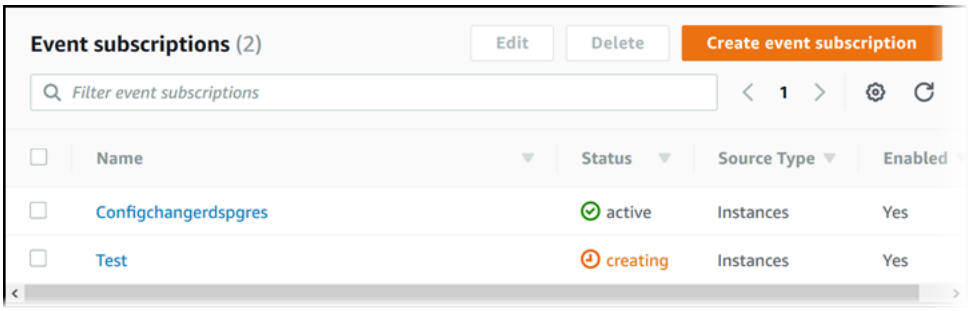
   d. Scegli le categorie di eventi e le risorse per i quali desideri ricevere notifiche eventi.

L'esempio seguente configura le notifiche degli eventi per l'istanza database denominata `testinst`.



e. Scegli Create (Crea).

La RDS console Amazon indica che l'abbonamento è in fase di creazione.



AWS CLI

Per iscriverti alla notifica RDS degli eventi, usa il AWS CLI `create-event-subscription`comando. Includi i parametri obbligatori seguenti:

- `--subscription-name`
- `--sns-topic-arn`

Example

Per LinuxmacOS, oUnix:

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

Per Windows:

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

API

Per abbonarti alla notifica RDS degli eventi di Amazon, chiama la RDS API funzione Amazon `CreateEventSubscription`. Includi i parametri obbligatori seguenti:

- `SubscriptionName`
- `SnsTopicArn`

Tag e attributi di notifica degli RDS eventi di Amazon

Quando Amazon RDS invia una notifica di evento ad Amazon Simple Notification Service (SNS) o Amazon EventBridge, la notifica contiene gli attributi del messaggio e i tag degli eventi. RDSinvia gli attributi del messaggio separatamente insieme al messaggio, mentre i tag dell'evento si trovano nel corpo del messaggio. Usa gli attributi dei messaggi e i RDS tag Amazon per aggiungere metadati alle tue risorse. Puoi modificare questi tag con notazioni personalizzate relative alle istanze database , cluster Aurora e così via. Per ulteriori informazioni sull'etichettatura RDS delle risorse Amazon, consultaEtichettatura di Amazon Aurora e risorse Amazon RDS.

Per impostazione predefinita, Amazon SNS e Amazon EventBridge ricevono ogni messaggio inviato loro. SNSe EventBridge può filtrare il messaggio e inviare le notifiche nella modalità di comunicazione preferita, ad esempio un'e-mail, un messaggio di testo o una chiamata a un HTTP endpoint.

> **Note**
>
> La notifica inviata tramite e-mail o SMS non includerà tag di evento.

La tabella seguente mostra gli attributi dei messaggi per gli RDS eventi inviati al sottoscrittore dell'argomento.

Attributo di RDS evento Amazon	Descrizione
EventID	Identificatore per il messaggio RDS dell'evento, ad esempio, RDS - EVENT -0006.
Risorsa	L'ARNidentificatore della risorsa che emette l'evento, ad esempio,. `arn:aws:rds:ap-southeast-2:123456789012:db:database-1`

I RDS tag forniscono dati sulla risorsa interessata dall'evento di servizio. RDSaggiunge lo stato corrente dei tag nel corpo del messaggio quando la notifica viene inviata a SNS o EventBridge.

Per ulteriori informazioni sul filtraggio degli attributi dei messaggi perSNS, consulta Amazon SNS Message Filtering nella Amazon Simple Notification Service Developer Guide.

Per ulteriori informazioni sul filtraggio dei tag degli eventi per EventBridge, consulta Operatori di confronto da utilizzare nei modelli di eventi in Amazon EventBridge nella Amazon EventBridge User Guide.

Per ulteriori informazioni sul filtraggio dei tag basati sul payload perSNS, consulta Introduzione al filtraggio dei messaggi basato sul payload per Amazon SNS

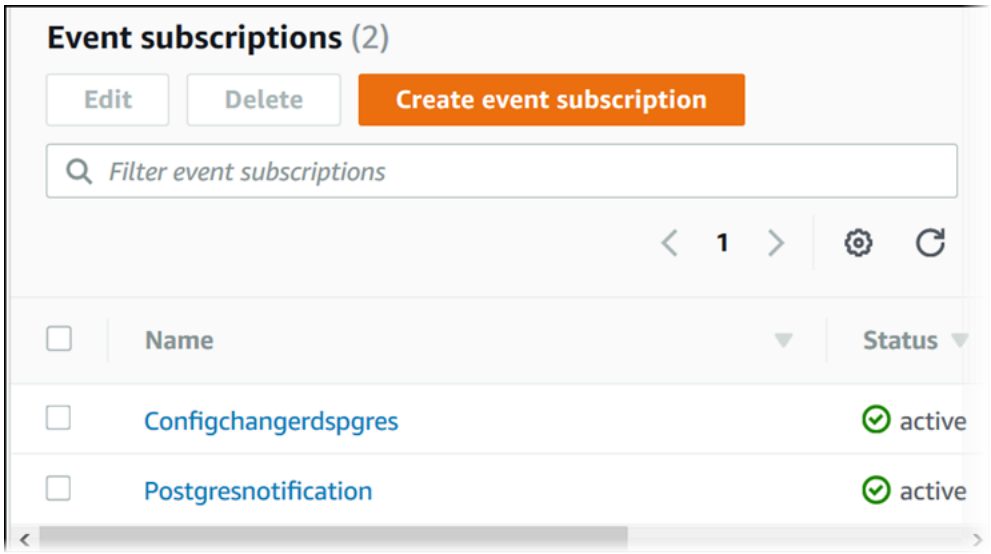
Pubblicazione degli abbonamenti Amazon per le notifiche di RDS eventi

Puoi elencare i tuoi attuali abbonamenti alle notifiche di RDS eventi Amazon.

Console

Per elencare i tuoi attuali abbonamenti alle notifiche di RDS eventi Amazon

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi). Il riquadro Event subscriptions (Sottoscrizioni di eventi) mostra tutte le sottoscrizioni delle notifiche degli eventi.



AWS CLI

Per elencare i tuoi attuali abbonamenti alle notifiche di RDS eventi Amazon, usa il AWS CLI `describe-event-subscriptions`comando.

Example

L'esempio seguente illustra tutte le sottoscrizioni a eventi.

```
aws rds describe-event-subscriptions
```

L'esempio seguente illustra `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

Per elencare i tuoi attuali abbonamenti alle notifiche di RDS eventi Amazon, chiama Amazon RDS API DescribeEventSubscriptionsaction.

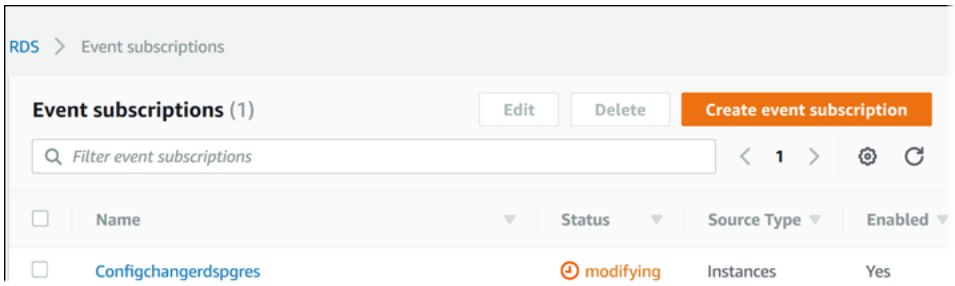
Modifica di un abbonamento per la notifica di RDS eventi Amazon

Dopo aver creato un abbonamento, puoi modificare il nome dell'abbonamento, l'identificatore della fonte, le categorie o l'argomentoARN.

Console

Per modificare un abbonamento ad Amazon RDS Event Notification

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione selezionare Event subscriptions (Sottoscrizioni di eventi).
3. Nel riquadro Event subscriptions (Sottoscrizioni di eventi) scegliere la sottoscrizione da modificare e selezionare Edit (Modifica).
4. Apportare le modifiche alla sottoscrizione nella sezione Target (Destinazione) o Source (Origine).
5. Seleziona Edit (Modifica). La RDS console Amazon indica che l'abbonamento è in fase di modifica.



AWS CLI

Per modificare un abbonamento ad Amazon RDS Event Notification, usa il AWS CLI `modify-event-subscription`comando. Includi il seguente parametro obbligatorio:

- `--subscription-name`

Example

Il codice seguente abilita `myeventsubscription`.

Per LinuxmacOS, oUnix:

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

Per Windows:

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

Per modificare un RDS evento Amazon, chiama l'RDSAPIoperazione Amazon `ModifyEventSubscription`. Includi il seguente parametro obbligatorio:

- `SubscriptionName`

Aggiungere un identificatore di origine a un abbonamento di notifica di RDS eventi Amazon

Puoi aggiungere un identificatore di origine (la RDS fonte Amazon che genera l'evento) a un abbonamento esistente.

Console

Puoi aggiungere o rimuovere facilmente gli identificatori di origine utilizzando la RDS console Amazon selezionandoli o deselezionandoli quando modifichi un abbonamento. Per ulteriori informazioni, consulta Modifica di un abbonamento per la notifica di RDS eventi Amazon.

AWS CLI

Per aggiungere un identificatore di origine a un abbonamento Amazon RDS Event Notification, usa il AWS CLI `add-source-identifier-to-subscription`comando. Includi i parametri obbligatori seguenti:

- `--subscription-name`
- `--source-identifier`

Example

L'esempio seguente aggiunge l'identificatore di origine `mysqldb` alla sottoscrizione `myrdseventsubscription`.

Per LinuxmacOS, oUnix:

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Per Windows:

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Per aggiungere un identificatore di origine a un abbonamento Amazon RDS Event Notification, chiama Amazon RDS API `AddSourceIdentifierToSubscription`. Includi i parametri obbligatori seguenti:

- `SubscriptionName`
- `SourceIdentifier`

Rimozione di un identificatore di origine da un abbonamento di notifica di RDS eventi Amazon

Puoi rimuovere un identificatore di origine (la RDS fonte Amazon che genera l'evento) da un abbonamento se non desideri più ricevere notifiche sugli eventi per quella fonte.

Console

Puoi aggiungere o rimuovere facilmente gli identificatori di origine utilizzando la RDS console Amazon selezionandoli o deselezionandoli quando modifichi un abbonamento. Per ulteriori informazioni, consulta Modifica di un abbonamento per la notifica di RDS eventi Amazon.

AWS CLI

Per rimuovere un identificatore di origine da un abbonamento Amazon RDS Event Notification, usa il AWS CLI `remove-source-identifier-from-subscription`comando. Includi i parametri obbligatori seguenti:

- `--subscription-name`
- `--source-identifier`

Example

L'esempio seguente rimuove l'identificatore dell'origine `mysqldb` dalla sottoscrizione `myrdseventsubscription`.

Per LinuxmacOS, oUnix:

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Per Windows:

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

Per rimuovere un identificatore di origine da un abbonamento Amazon RDS Event Notification, usa il RDS API `RemoveSourceIdentifierFromSubscription`comando Amazon. Includi i parametri obbligatori seguenti:

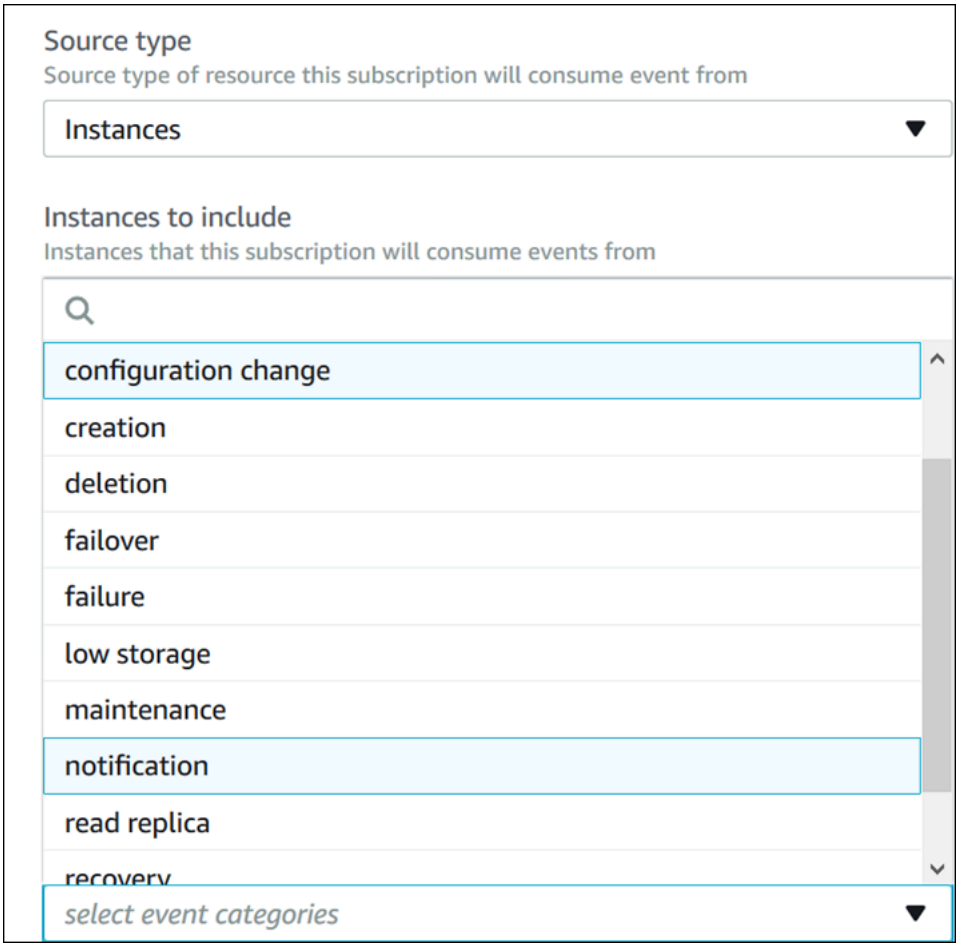
- `SubscriptionName`
- `SourceIdentifier`

Elenco delle categorie di notifiche di RDS eventi di Amazon

Tutti gli eventi per un tipo di risorsa sono raggruppati in categorie. Per visualizzare l'elenco delle categorie disponibili, utilizza le procedure riportate di seguito.

Console

Quando crei o modifichi un abbonamento per la notifica degli eventi, le categorie di eventi vengono visualizzate nella RDS console Amazon. Per ulteriori informazioni, consulta Modifica di un abbonamento per la notifica di RDS eventi Amazon.



AWS CLI

Per elencare le categorie di notifica RDS degli eventi di Amazon, usa il AWS CLI `describe-event-categories`comando. Questo comando non prevede parametri obbligatori.

Example

```
aws rds describe-event-categories
```

API

Per elencare le categorie di notifiche di RDS eventi Amazon, usa il RDS API `DescribeEventCategories`comando Amazon. Questo comando non prevede parametri obbligatori.

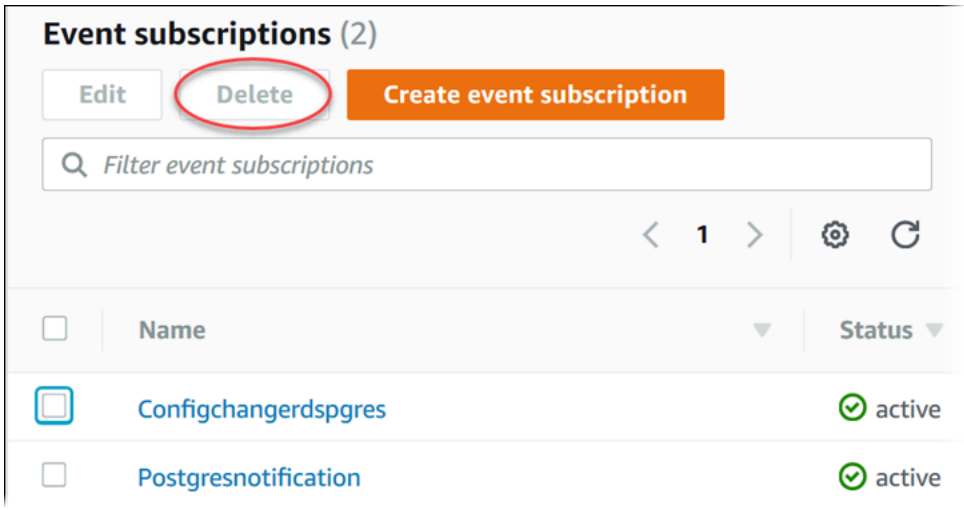
Eliminazione di un abbonamento per la notifica di RDS eventi Amazon

Puoi eliminare un abbonamento quando questo non è più necessario. Tutti gli abbonati all'argomento non riceveranno più le notifiche di eventi specificate dall'abbonamento.

Console

Per eliminare un abbonamento ad Amazon RDS Event Notification

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegliere DB Event Subscriptions (Sottoscrizioni di eventi database).
3. Nel riquadro My DB Event Subscriptions (Sottoscrizioni di eventi database personali) scegliere la sottoscrizione che si vuole eliminare.
4. Scegli Elimina.
5. La RDS console Amazon indica che l'abbonamento è stato eliminato.



AWS CLI

Per eliminare un abbonamento Amazon RDS Event Notification, usa il AWS CLI `delete-event-subscription`comando. Includi il seguente parametro obbligatorio:

- `--subscription-name`

Example

L'esempio seguente elimina la sottoscrizione `myrdssubscription`.

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

API

Per eliminare un abbonamento Amazon RDS Event Notification, usa il RDS API `DeleteEventSubscription`comando. Includi il seguente parametro obbligatorio:

- `SubscriptionName`

Creazione di una regola che si attiva su un evento Amazon Aurora

Con Amazon EventBridge, puoi automatizzare AWS i servizi e rispondere a eventi di sistema come problemi di disponibilità delle applicazioni o modifiche delle risorse.

Argomenti

- Tutorial: registra le modifiche allo stato delle istanze DB utilizzando Amazon EventBridge

Tutorial: registra le modifiche allo stato delle istanze DB utilizzando Amazon EventBridge

In questo tutorial, crei una AWS Lambda funzione che registra le modifiche di stato per un'istanza Successivamente crei una regola che esegua la funzione ogni volta che si verifica un cambiamento di stato di un'istanza database RDS esistente. Il tutorial presuppone che si dispone di una piccola istanza di test in esecuzione che è possibile arrestare temporaneamente.

> **Important**
>
> Non eseguire questo tutorial su un'istanza database di produzione in esecuzione.

Argomenti

- Passaggio 1: creare una funzione AWS Lambda
- Fase 2: Creazione di una regola
- Fase 3: Test della regola

Passaggio 1: creare una funzione AWS Lambda

Crea una funzione Lambda per registrare gli eventi di modifica dello stato. È necessario specificare questa funzione alla creazione della regola.

Per creare una funzione Lambda

1. Apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Se è la prima volta che utilizzi Lambda, verrà visualizzata una pagina di benvenuto. Selezionare Get Started Now (Inizia subito). Altrimenti, scegliere Create function (Crea funzione).
3. Scegli Author from scratch (Crea da zero).

4. Nella pagina Create function (Crea funzione), procedere come segue:

   a. Digitare un nome e una descrizione per la funzione Lambda. Ad esempio, denomina la funzione **`RDSInstanceStateChange`**.

   b. In Runtime, seleziona Node.js 16x.

   c. In Architecture (Architettura), scegli x86_64.

   d. In Execution role (Ruolo di esecuzione), esegui una delle operazioni seguenti:

      - Scegliere Create a new role with basic Lambda permissions (Crea un nuovo ruolo con le autorizzazioni Lambda di base).
      - In Execution role (Ruolo di esecuzione), sceglie Use an existing role (Utilizza un ruolo esistente). Scegli il ruolo che desideri usare.

   e. Scegli Crea funzione.

5. Nella InstanceStateChange pagina RDS, procedi come segue:

   a. In Origine codice, seleziona index.js.

   b. Nel riquadro di index.js, elimina il codice esistente.

   c. Immetti il seguente codice:

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

   d. Selezionare Deploy (Distribuisci).

Fase 2: Creazione di una regola

Crea una regola per l'esecuzione della funzione Lambda ogni volta che avvii un'istanza Amazon RDS.

Per creare la regola EventBridge

1. Apri la EventBridge console Amazon all'indirizzo https://console.aws.amazon.com/events/.
2. Nel pannello di navigazione, scegli Regole.
3. Scegli Create rule (Crea regola).

4. Immettere un nome e una descrizione per la regola. Ad esempio, specifica **`RDSInstanceStateChangeRule`**.
5. Scegli Rule with an event pattern (Regola con un modello di eventi), quindi seleziona Next (Successivo).
6. Per Event source, scegli AWS eventi o eventi EventBridge partner.
7. Scorri verso il basso fino alla sezione Event pattern (Modello di eventi).
8. In Event source (Origine eventi), selezionare Servizi AWS.
9. In AWS service (Servizio AWS), scegli Relational Database Service (RDS).
10. Per Tipo di evento, seleziona Evento istanza database RDS.
11. Lascia il modello di eventi predefinito. Quindi scegli Successivo.
12. Per Target types (Tipi di destinazione), scegli AWS service (Servizio).
13. Per Select a target (Seleziona destinazione), scegli Lambda function (Funzione Lambda).
14. In Function (Funzione), seleziona la funzione Lambda che hai creato. Quindi scegli Successivo.
15. In Configure tags (Configura tag), scegli Next (Successivo).
16. Esamina i passaggi nella regola. Quindi scegli Create rule (Crea regola).

Fase 3: Test della regola

Per verificare la regola, arresta un'istanza database RDS. Dopo aver atteso alcuni minuti perché l'istanza venga avviata e inizializzata, verifica che la funzione Lambda sia stata richiamata.

Per effettuare il test della regola arrestando un'istanza database

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Arresta un'istanza database RDS.
3. Apri la EventBridge console Amazon all'indirizzo https://console.aws.amazon.com/events/.
4. Nel pannello di navigazione, seleziona Regole, scegli il nome della regola creata.
5. In Dettagli della regola scegli Monitoraggio.

   Verrai reindirizzato alla CloudWatch console Amazon. Se non vieni reindirizzato, fai clic su Visualizza le metriche in. CloudWatch
6. In Tutti i parametri, seleziona il nome della regola creata.

   Il grafico deve indicare che la regola è stata richiamata.

7. Nel pannello di navigazione, selezionare Log groups (Gruppi di log).
8. Scegli il nome del gruppo di log per la funzione Lambda (/aws/lambda/***nome-funzione***).
9. Scegliere il nome del flusso di log per visualizzare i dati forniti dalla funzione per l'istanza avviata. Sarà visualizzato un evento ricevuto simile a quello seguente:

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

   Per altri esempi di eventi RDS in formato JSON, vedere .
10. (Facoltativo) Al termine, puoi aprire la console Amazon RDS e avviare l'istanza terminata.

Categorie di eventi e messaggi di eventi di Amazon RDS per Aurora

Amazon RDS genera un numero significativo di eventi in categorie a cui puoi abbonarti utilizzando la console Amazon RDS o AWS CLI l'API.

Argomenti

- Eventi di cluster di database
- Eventi snapshot cluster di database
- Eventi di istanza database
- Eventi gruppo di parametri database
- Eventi gruppo di sicurezza DB
- Eventi del gruppo DB Shard
- Eventi RDS Proxy
- Eventi di implementazione blu/verde

Eventi di cluster di database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un cluster database.

> **Note**
>
> Non esiste alcuna categoria di eventi per Aurora Serverless nel tipo di evento cluster database. Gli eventi serverless Aurora vanno da RDS-EVENT-0141 a RDS-EVENT-0149.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0016	Reimpostare le credenziali master.	Nessuno
modifica della configurazione	RDS-EVENT-0179	I flussi di attività del database vengono avviati nel cluster di database.	Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora

Categoria	ID evento RDS	Messaggio	Note
			tramite i flussi di attività del database.
modifica della configurazione	RDS-EVENT-0180	I flussi di attività del database vengono arrestati nel cluster di database.	Per ulteriori informazioni consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.
creazione	RDS-EVENT-0170	Cluster di database creato.	Nessuno
eliminazione	RDS-EVENT-0171	Cluster di database eliminato.	Nessuno
failover	RDS-EVENT-0069	Failover del cluster non riuscito. Verificare lo stato delle istanze del cluster e riprovare.	Nessuno
failover	RDS-EVENT-0070	Promuovere nuovamente le primarie precedenti:. *name*	Nessuno
failover	RDS-EVENT-0071	Failover completato sull'istanza DB:*name*.	Nessuno
failover	RDS-EVENT-0072	È stato avviato il failover dello stesso tipo AZ sull'istanza DB:. *name*	Nessuno
failover	RDS-EVENT-0073	Avviato il failover cross-AZ verso un'istanza DB:. *name*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0083	Amazon RDS non è riuscito a creare credenziali per accedere al tuo bucket Amazon S3 per il tuo cluster DB. *name* Possibili cause: il ruolo IAM di acquisizione di snapshot S3 non è configurato correttamente nel tuo account o non è stato trovato il bucket Amazon S3 specificato. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documenta zione di Amazon RDS.	Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3 .
errore	RDS-EVENT-0143	Il cluster DB non è riuscito a scalare da *units* a *units* per questo motivo:. *reason*	Scaling non riuscito per il cluster di database Aurora Serverless.
errore	RDS-EVENT-0354	Non è possibile creare il cluster DB a causa di risorse incompatibili. *message*.	*message*Include dettagli sull'errore.
errore	RDS-EVENT-0355	Il cluster DB non può essere creato a causa di limiti di risorse insufficienti. *message*.	*message*Include dettagli sull'errore.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0387	Impossibile partizionare le istanze DB nel cluster DB per l'applicazione di patch. *name*	Gli aggiornamenti del sistema operativo per le istanze DB nel cluster DB non sono riusciti.
Failover globale	RDS-EVENT-0181	È iniziato il passaggio globale al cluster *name* DB nella regione. *name*	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito"). Il processo può essere ritardato perché altre operazioni sono in esecuzione sul cluster di database.
Failover globale	RDS-EVENT-0182	Il vecchio cluster DB primario *name* nella regione è *name* stato chiuso con successo.	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito"). La vecchia istanza principale nel database globale non accetta scritture. Tutti i volumi sono sincronizzati.

Categoria	ID evento RDS	Messaggio	Note
Failover globale	RDS-EVENT-0183	In attesa della sincroniz zazione dei dati tra i membri del cluster globale. Attualmente è in ritardo rispetto al cluster DB primario:*reason*.	Questo evento riguarda un'operazione di switchove r (precedentemente denominata "failover pianificato gestito"). Si verifica un ritardo di replica durante la fase di sincronizzazione del failover globale del database.
Failover globale	RDS-EVENT-0184	Il nuovo cluster DB primario *name* nella regione *name* è stato promosso con successo.	Questo evento riguarda un'operazione di switchove r (precedentemente denominata "failover pianificato gestito"). La topologia del volume del database globale viene ristabilita con il nuovo volume primario.

Categoria	ID evento RDS	Messaggio	Note
Failover globale	RDS-EVENT-0185	Il passaggio globale al cluster DB *name* nella regione *name* è terminato.	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito"). Lo switchover globale del database è stato completato nel cluster database primario. Le repliche potrebbero richiedere molto tempo per arrivare online dopo il completamento del failover.
Failover globale	RDS-EVENT-0186	Il passaggio globale al cluster DB nella regione è stato *name* annullato. *name*	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito").
Failover globale	RDS-EVENT-0187	Il passaggio globale al cluster DB nella regione non è riuscito. *name name*	Questo evento riguarda un'operazione di switchover (precedentemente denominata "failover pianificato gestito").
Failover globale	RDS-EVENT-0238	Il failover globale verso il cluster DB *name* nella regione è stato completato. *name*	Nessuno
Failover globale	RDS-EVENT-0239	Il failover globale sul cluster DB *name* nella regione *name* non è riuscito.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
Failover globale	RDS-EVENT-0240	È iniziata la risincron izzazione dei membri del cluster DB *name* nella regione *name* dopo il failover globale.	Nessuno
Failover globale	RDS-EVENT-0241	È stata completata la risincronizzazione dei membri del cluster DB nella regione dopo il failover globale. *name name*	Nessuno
Failover globale	RDS-EVENT-0397	Aurora ha terminato la modifica del nome DNS in cui si risolve l'endpoint Global Writer.	Nessuno
Failover globale	RDS-EVENT-0423	In attesa della sincroniz zazione dei dati con il cluster DB di destinazione. Il cluster DB di destinazione attuale è in ritardo rispetto al cluster DB primario:. *%s*	Questo evento riguarda un'operazione di switchove r (precedentemente denominata "failover pianificato gestito"). Si verifica un ritardo di replica durante la fase di sincronizzazione del failover globale del database.
manutenzi one	RDS-EVENT-0156	È disponibile un aggiornam ento della versione secondaria del motore del database per il cluster di database.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0173	La versione del motore del cluster database è stata aggiornata.	L'applicazione di patch del cluster database è stata completata.
manutenzione	RDS-EVENT-0174	Il cluster di database si trova in uno stato che non può essere aggiornato.	Nessuno
manutenzione	RDS-EVENT-0176	La versione principale del motore del cluster database è stata aggiornata.	Nessuno
manutenzione	RDS-EVENT-0177	L'aggiornamento del cluster di database è in corso.	Nessuno
manutenzione	RDS-EVENT-0286	L'aggiornamento della *version_number* versione del motore del cluster di database è iniziato. Il cluster rimane online.	Nessuno
manutenzione	RDS-EVENT-0287	Requisito di aggiornamento del sistema operativo rilevato.	Nessuno
manutenzione	RDS-EVENT-0288	Avvio dell'aggiornamento del sistema operativo del cluster in corso.	Nessuno
manutenzione	RDS-EVENT-0289	Aggiornamento del sistema operativo del cluster completato.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0290	Il cluster di database è stato patchato: versione sorgente *version_number new_version_number* =>.	Nessuno
manutenzione	RDS-EVENT-0363	Preparazione dell'aggiornamento in corso: *cluster_name*	I precontrolli di aggiornamento sono iniziati per il cluster DB.
manutenzione	RDS-EVENT-0388	Avvio di patch offline alle istanze DB nella partizione/per il cluster DB:. *name name name partition_n*	Avvio degli aggiornamenti del sistema operativo per le istanze DB nel cluster DB.
manutenzione	RDS-EVENT-0389	Non è stato possibile aggiornare il sistema operativo del cluster DB. È possibile attendere la finestra di manutenzione successiva oppure aggiornare manualmente il sistema operativo del cluster DB.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0424	La versione in esecuzione del cluster DB *name* è superiore alla versione *version* di aggiornamento di destinazione per il cluster globale. *version* Non è consigliabile avere un cluster secondario su una versione superiore rispetto al cluster globale, poiché può causare problemi durante il failover o lo switchover. Valuta la possibilità di aggiornare il tuo cluster globale in modo da renderlo compatibile.	Nessuno
notification	RDS-EVENT-0076	Migrazione da a non riuscita. *name name* Motivo:*reason*.	La migrazione a un cluster di database Aurora non è riuscita.
notifica	RDS-EVENT-0077	Conversione non riuscita*name*. *name*a InnoDB. Motivo:*reason*.	Un tentativo di convertire una tabella dal database di origine a InnoDB non è riuscito durante la migrazione a un cluster di database Aurora.

Categoria	ID evento RDS	Messaggio	Note
notifica	RDS-EVENT-0085	Impossibile aggiornare il cluster DB *name* perché lo stato dell'istanza *name* è di*name*. Risolvere il problema o eliminare l'istanza e riprovare.	Si è verificato un errore durante il tentativo di patch al cluster Aurora DB. Controlla lo stato dell'istanza, risolvi il problema e riprova. Per ulteriori informazioni, consulta Manutenzione di un cluster database Amazon Aurora.
notifica	RDS-EVENT-0141	Scalabilità del cluster DB da *units* a *units* per questo motivo:*reason*.	Scaling avviato per il cluster di database Aurora Serverless.
notifica	RDS-EVENT-0142	Il cluster DB è stato scalato da a*units*. *units*	Scaling completato per il cluster di database Aurora Serverless.
notifica	RDS-EVENT-0144	Il cluster database verrà messo in pausa.	È stata avviata una pausa automatica per il cluster database Aurora Serverless.
notification	RDS-EVENT-0145	Il cluster database è in pausa.	Il cluster database Aurora Serverless è stato messo in pausa.
notification	RDS-EVENT-0146	Pausa annullata per il cluster database.	Pausa per il cluster database Aurora Serverless annullata.
notification	RDS-EVENT-0147	Il cluster database è in fase di riattivazione.	Operazione di riattivazione avviata per il cluster database Aurora Serverless.

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0148	Il cluster database è stato riattivato.	Operazione di riattivazione per il cluster database Aurora Serverless completata.
notification	RDS-EVENT-0149	Il cluster DB è stato scalato da *units* a*units*, ma la scalabilità non è stata perfetta per questo motivo:. *reason*	Scaling continuo completato con l'opzione forzata per il cluster di database Aurora Serverless. Le connessioni potrebbero essere state interrotte come richiesto.
notifica	RDS-EVENT-0150	Cluster database arrestato.	Nessuno
notification	RDS-EVENT-0151	Cluster database avviato.	Nessuno
notification	RDS-EVENT-0152	Arresto del cluster database non riuscito.	Nessuno
notification	RDS-EVENT-0153	Il cluster di database è in fase di avvio perché è stato superato il tempo massimo concesso per l'arresto.	Nessuno
notification	RDS-EVENT-0172	Cluster rinominato da a. *name name*	Nessuno
notification	RDS-EVENT-0234	Attività di esportazione non riuscita.	L'attività di esportazione del cluster database non è riuscita.
notification	RDS-EVENT-0235	Attività di esportazione annullata.	L'attività di esportazione del cluster database è stata annullata.

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0236	Attività di esportazione completata.	L'attività di esportazione del cluster database è stata completata.
notification	RDS-EVENT-0386	Le istanze DB nel cluster DB sono state partizion ate:. *name list_of_p artitions* Il cluster DB è online.	Gli aggiornamenti del sistema operativo per le istanze DB nel cluster DB hanno avuto esito positivo.
notification	RDS-EVENT-0426	RDS non è in grado di configurare il cluster DB *name* come origine di replica a causa di transazio ni inattive o di lunga durata. Attendi il completamento delle transazioni o annullale e riprova.	Nessuno

Eventi snapshot cluster di database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è uno snapshot del cluster database.

Categoria	ID evento RDS	Messaggio	Note
backup	RDS-EVENT-0074	Creazione di uno snapshot del cluster manuale in corso.	Nessuno
backup	RDS-EVENT-0075	Snapshot del cluster manuale creato.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
notification	RDS-EVENT-0162	Attività di esportazione dello snapshot del cluster non riuscita.	Nessuno
notification	RDS-EVENT-0163	Attività di esportazione dello snapshot del cluster annullata.	Nessuno
notification	RDS-EVENT-0164	Attività di esportazione dello snapshot del cluster completata.	Nessuno
backup	RDS-EVENT-0168	Creazione snapshot cluster automatizzato.	Nessuno
backup	RDS-EVENT-0169	Snapshot di cluster automatizzato creato.	Nessuno

Eventi di istanza database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un'istanza database.

Categoria	ID evento RDS	Messaggio	Note
disponibilità	RDS-EVENT-0004	L'istanza database è stata arrestata.	Nessuno
availability	RDS-EVENT-0006	L'istanza database è stata riavviata.	Nessuno
disponibilità	RDS-EVENT-0022	Errore durante il riavvio di mysql:. *message*	Si è verificato un errore durante il riavvio di Aurora MySQL o RDS per MariaDB.

Categoria	ID evento RDS	Messaggio	Note
disponibilità	RDS-EVENT-0419	Amazon RDS non è riuscito ad accedere alla chiave di KMS crittografia per l'istanza *name* del database. Questo database verrà collocato in uno stato inaccessibile. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Nessuno
backtrack	RDS-EVENT-0131	La finestra di backtrack effettiva è più piccola della finestra di backtrack target specificata. Prendi in considerazione di ridurre il numero di ore della finestra di backtrack target.	Per ulteriori informazioni sul backtrack, consulta Backtrack di un cluster database Aurora.
backtrack	RDS-EVENT-0132	La finestra di backtrack effettiva è uguale alla finestra di backtrack target specificata.	Nessuno
modifica della configurazione	RDS-EVENT-0011	Aggiornato per utilizzare DBParameter Group*name*.	Nessuno
modifica della configurazione	RDS-EVENT-0012	Vengono applicate le modifiche alla classe di istanza database.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0014	Applicazione delle modifiche alla classe di istanza database completata.	Nessuno
modifica della configurazione	RDS-EVENT-0017	Applicazione delle modifiche all'archiviazione allocata completata.	Nessuno
modifica della configurazione	RDS-EVENT-0025	Applicazione delle modifiche per la conversione in un'istanza database Multi-AZ completata.	Nessuno
modifica della configurazione	RDS-EVENT-0029	Applicazione delle modifiche per la conversione in un'istanza database standard (Single-AZ) completata.	Nessuno
modifica della configurazione	RDS-EVENT-0033	Ci sono *number* utenti che corrispondono al nome utente principale; viene reimpostato solo quello non legato a un host specifico.	Nessuno
modifica della configurazione	RDS-EVENT-0067	Impossibile reimpostare la password. Informazioni sull'errore:. *message*	Nessuno
modifica della configurazione	RDS-EVENT-0078	Intervallo di monitoraggio modificato in. *number*	La configurazione Enhanced Monitoring è stata modificata.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0092	Aggiornamento gruppo di parametri DB terminato.	Nessuno
creazione	RDS-EVENT-0005	Viene creata l'istanza database.	Nessuno
eliminazione	RDS-EVENT-0003	Istanza database eliminata.	Nessuno
errore	RDS-EVENT-0035	Istanza di database inserita in*state*. *message*.	L'istanza database include parametri non validi. Ad esempio, se non è stato possibile avviare l'istanza database perché un parametro relativo alla memoria è troppo elevato per questa classe di istanze, è necessario modificare il parametro relativo alla memoria e riavviare l'istanza database.
errore	RDS-EVENT-0036	Istanza di database in*state*. *message*.	L'istanza database si trova in una rete non compatibile. Alcune delle sottoreti specificate non IDs sono valide o non esistono.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0079	Amazon RDS non è stato in grado di creare le credenziali per un monitoraggio avanzato e questa funzionalità è stata disattivata. Ciò è probabilmente dovuto al fatto che rds-monitoring-role non è presente e non è configurato correttamente nel tuo account. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Impossibile abilitare la funzionalità Monitoraggio avanzato senza il ruolo IAM di monitoraggio avanzato. Per ulteriori informazioni sulla creazione del ruolo IAM, consulta Per creare un ruolo IAM per Amazon RDS Enhanced Monitoring.
errore	RDS-EVENT-0080	Amazon RDS non è stato in grado di configurare il monitoraggio avanzato sulla tua istanza *name* e questa funzionalità è stata disabilitata. Ciò è probabilmente dovuto al fatto che rds-monitoring-role non è presente e non è configurato correttamente nel tuo account. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	La funzionalità Monitoraggio avanzato è stata disabilitata a causa di un errore che ha provocato la modifica della configurazione. È possibile che il ruolo IAM di monitoraggio avanzato non sia configurato correttamente. Per informazioni sulla creazione del ruolo IAM di monitoraggio avanzato, consulta Per creare un ruolo IAM per Amazon RDS Enhanced Monitoring.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0082	Amazon RDS non è riuscito a creare credenziali per accedere al tuo Amazon S3 Bucket per la tua istanza DB. *name* Possibili cause: il ruolo IAM di acquisizione di snapshot S3 non è configurato correttamente nel tuo account o non è stato trovato il bucket Amazon S3 specificato. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Aurora non è riuscito a copiare i dati di backup da un bucket Simple Storage Service (Amazon S3). È probabile che le autorizzazioni necessarie perché Aurora possa accedere al bucket Simple Storage Service (Amazon S3) non siano configurate correttamente. Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3 .
errore	RDS-EVENT-0353	L'istanza DB non può essere creata a causa di limiti di risorse insufficienti. *message*.	*message*Include dettagli sull'errore.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0418	Amazon RDS non è in grado di accedere alla chiave di KMS crittografia per l'istanza *name* del database. Ciò è probabilmente dovuto al fatto che la chiave è disabilitata o che Amazon RDS non è in grado di accedervi. Se il problema persiste, il database verrà posto in uno stato inaccessibile. Per ulteriori dettagli, consulta la sezione relativa alla risoluzione dei problemi nella documentazione di Amazon RDS.	Nessuno
errore	RDS-EVENT-0420	Amazon RDS ora può accedere correttamente alla chiave di KMS crittografia per l'istanza *name* di database.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
storage insufficiente	RDS-EVENT-0007	Lo spazio di archiviazione allocato è stato esaurito. Allocare spazio di archiviazione aggiuntivo per risolvere il problema.	Lo storage allocato per l'istanza database è esaurito. Per risolvere questo problema, assegna ulteriore storage per l'istanza database. Per ulteriori informazioni, consulta le domande frequenti su RDS. Puoi monitorare lo spazio di storage per un'istanza database usando il parametro Free Storage Space (Spazio di storage libero).
storage insufficiente	RDS-EVENT-0089	La capacità di storage gratuita per l'istanza DB: *name* è bassa rispetto allo *percentage* storage fornito [Provisioned Storage:*size*, Free Storage:]. *size* È possibile aumentare lo spazio di archiviazione assegnato per risolvere questo problema.	L'istanza database ha utilizzato oltre il 90% dello storage allocato. Puoi monitorare lo spazio di storage per un'istanza database usando il parametro Free Storage Space (Spazio di storage libero).
storage insufficiente	RDS-EVENT-0227	Lo storage del cluster Aurora è pericolosamente basso e rimangono solo terabyte. *amount* Prendere misure idonee per ridurre il carico di archiviazione sul cluster.	Il sottosistema di archiviazione di Aurora sta esaurendo lo spazio.

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0026	Applicazione di patch offline all'istanza database in corso.	È in corso la manutenzione offline dell'istanza database. L'istanza database non è attualmente disponibile.
manutenzione	RDS-EVENT-0027	Applicazione di patch offline all'istanza database terminata.	La manutenzione offline dell'istanza database è stata completata. L'istanza database è ora disponibile.
manutenzione	RDS-EVENT-0047	Applicazione delle patch all'istanza database completata.	Nessuno
manutenzione	RDS-EVENT-0155	È disponibile un aggiornamento della versione secondaria del motore di database per l'istanza database.	Nessuno
manutenzione	RDS-EVENT-0178	L'aggiornamento dell'istanza del database è in corso.	Nessuno
manutenzione	RDS-EVENT-0422	RDS sostituirà l'host dell'istanza DB a *name* causa di un'azione di manutenzione in sospeso.	Nessuno
notifica	RDS-EVENT-0044	*message*	Si tratta di una notifica emessa dall'operatore. Per ulteriori informazioni, consulta il messaggio di evento.

Categoria	ID evento RDS	Messaggio	Note
notifica	RDS-EVENT-0048	Ritardo dell'aggiornamento del motore di database in corso perché questa istanza contiene repliche di lettura che prima devono essere aggiornate.	L'applicazione di patch all'istanza database è stata ritardata.
notifica	RDS-EVENT-0087	Istanza database arrestata.	Nessuno
notification	RDS-EVENT-0088	Istanza database creata.	Nessuno
notification	RDS-EVENT-0365	I file del fuso orario sono stati aggiornati. Riavvia l'istanza RDS per rendere effettive le modifiche.	Nessuno
notifica, senza server	RDS-EVENT-0370	Pausa avviata per l'istanza DB.	È stato avviato un nuovo tentativo di mettere in pausa un'istanza Aurora Serverless v2 DB inattiva.
notifica, senza server	RDS-EVENT-0371	La pausa è stata annullata per l'istanza DB.	Un tentativo di mettere in pausa un'istanza Aurora Serverless v2 DB inattiva non ha avuto successo a causa del carico di lavoro.
notifica, senza server	RDS-EVENT-0372	Sospensione dell'istanza DB completata.	L'istanza Aurora Serverles s v2 DB è stata messa in pausa.

Categoria	ID evento RDS	Messaggio	Note
notifica, senza server	RDS-EVENT-0373	Curriculum avviato per l'istanza DB.	L'istanza Aurora Serverless v2 DB ha iniziato a riprendere a causa di un nuovo carico di lavoro o di attività amministrative o di manutenzione.
notifica, senza server	RDS-EVENT-0374	L'istanza DB è stata ripresa con successo.	L'istanza Aurora Serverless v2 DB è stata ripresa.
notification	RDS-EVENT-0385	La topologia del cluster viene aggiornata.	Sono state apportate modifiche DNS al cluster DB per l'istanza DB. Ciò include quando vengono aggiunte o eliminate nuove istanze DB o si verifica un failover.
notifica, database globale	RDS-EVENT-0390	Il tentativo di bloccare le scritture per il cluster *cluster_id* DB nella regione è riuscito. *region_id*	Aurora ha iniziato a bloccare le scritture a livello di storage in preparazione dello switchover o del failover di un database globale Aurora.

Categoria	ID evento RDS	Messaggio	Note
notifica, database globale	RDS-EVENT-0391	Il tentativo di bloccare le scritture per il cluster *cluster_id* DB nella regione è scaduto. *region_id*	Aurora non è stata in grado di bloccare le scritture a livello di storage in preparazione dello switchover o del failover di un database globale Aurora. Lo switchover o il failover procederanno, ma potrebbe essere necessari o recuperare i dati scritti di recente dall'istantanea del cluster primario originale.
replica di lettura	RDS-EVENT-0045	La replica è stata interrotta.	Questo messaggio viene visualizzato quando si verifica un errore durante la replica. Per determina re il tipo di errore, consulta Risoluzione di un problema di replica di lettura MySQL.
replica di lettura	RDS-EVENT-0046	Replica della replica di lettura ripristinata.	Questo messaggio viene visualizzato quando crei per la prima volta una replica di lettura o come messaggio di monitorag gio che conferma il corretto funzionamento della replica. Se il messaggio segue una notifica `RDS-EVENT-0045` , la replica viene ripristinata in seguito a un errore o dopo l'arresto della replica.

Categoria	ID evento RDS	Messaggio	Note
replica di lettura	RDS-EVENT-0057	Lo streaming della replica è stata interrotta.	Nessuno
recupero	RDS-EVENT-0020	È stato avviato il recupero dell'istanza database. La durata del recupero varia in funzione della quantità di dati da recuperare.	Nessuno
recupero	RDS-EVENT-0021	È stato completato il recupero dell'istanza database.	Nessuno
recupero	RDS-EVENT-0023	Richiesta istantanea emergente:. *message*	È stato richiesto un backup manuale, ma in Amazon RDS è in corso la creazione di uno snapshot DB. Invia di nuovo la richiesta quando Amazon RDS avrà completato lo snapshot DB.
recupero	RDS-EVENT-0052	Ripristino dell'istanza Multi-AZ avviato.	La durata del recupero varia in funzione della quantità di dati da recuperare.
recupero	RDS-EVENT-0053	Ripristino dell'istanza Multi-AZ completato. Failover o attivazione in sospeso.	Questo messaggio indica che Amazon RDS ha preparato l'istanza DB per avviare un failover sull'istanza secondaria, se necessario.

Categoria	ID evento RDS	Messaggio	Note
recupero	RDS-EVENT-0361	Il ripristino dell'istanza DB in standby è iniziato.	L'istanza DB in standby viene ricostruita durante il processo di ripristino. Le prestazioni del database vengono influenzate durante il processo di ripristino.
recupero	RDS-EVENT-0362	Il ripristino dell'istanza DB in standby è stato completato.	L'istanza DB in standby viene ricostruita durante il processo di ripristino. Le prestazioni del database vengono influenzate durante il processo di ripristino.
ripristino	RDS-EVENT-0019	Ripristinato da un'istanza DB *name* a. *name*	L'istanza DB è stata ripristinata da un point-in-time backup.
applicazione di patch di sicurezza	RDS-EVENT-0230	È disponibile un aggiornamento del sistema per l'istanza database. Per informazioni sull'applicazione degli aggiornamenti, consulta "Manutenzione di un'istanza database" nella Guida per l'utente di RDS.	Una nuova versione secondaria della patch del sistema operativo è disponibile per l'istanza DB. Per informazioni sull'applicazione degli aggiornamenti, consulta Aggiornamenti del sistema operativo per i cluster Aurora DB.

Eventi gruppo di parametri database

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo dei parametri database.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0037	Parametro aggiornato *name* al metodo *value* with apply*method*.	Nessuno

Eventi gruppo di sicurezza DB

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un gruppo di sicurezza DB.

> **Note**
>
> I gruppi di sicurezza DB sono risorse per EC2 -Classic. EC2-Classic è stato ritirato il 15 agosto 2022. Se non hai effettuato la migrazione da EC2 -Classic a un VPC, ti consigliamo di migrare il prima possibile. Per ulteriori informazioni, consulta Migrare da EC2 -Classic a un VPC nella Amazon EC2 User Guide e nel blog EC2-Classic Networking is Retiring — Here's How to Prepare.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0038	Modifica al gruppo di sicurezza applicata.	Nessuno
errore	RDS-EVENT-0039	*user*Revoca dell'autorizzazione come.	Il gruppo di sicurezza di proprietà di *user* non esiste. L'autorizzazione per gruppo di sicurezza è stata revocata perché non è valida.

Eventi del gruppo DB Shard

La tabella seguente mostra la categoria di eventi e un elenco di eventi quando il tipo di origine è un gruppo di shard DB.

Categoria	ID evento RDS	Messaggio	Note
errore	RDS-EVENT-0324	Processo di caricamento dati non riuscito.	Il processo di caricamento dei dati non è riuscito per il motivo indicato nel messaggio di errore.
manutenzione	RDS-EVENT-0302	L'azione è necessaria per finalizzare un job split shard in sospeso *job_ID* per l'ID del subcluster nel gruppo di shard DB. *number name* Per completare l'operazione, richiamate il codice SQL: `SELECT rds_aurora.limitless_finalize_split_shard( job_ID);`	Nessuno
manutenzione	RDS-EVENT-0303	È stato avviato il processo di finalizzazione dello split shard per l'ID del *job_ID* subcluster nel gruppo di shard DB. *number name*	Nessuno
manutenzione	RDS-EVENT-0304	Il processo di divisione dello shard *job_ID* è stato completato correttamente per l'ID del subcluster nel gruppo di shard DB.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
		number name Un nuovo shard con ID di subcluster *number* è stato aggiunto al gruppo di shard DB. *name*	
manutenzione	RDS-EVENT-0305	Il processo di divisione dello shard non *job_ID* è riuscito per l'ID del subcluster nel gruppo di shard DB. *number name*	Nessuno
manutenzione	RDS-EVENT-0366	Il processo di split shard *job_ID* è stato avviato per l'ID del sottocluster nel gruppo di shard DB. *number name*	Nessuno
manutenzione	RDS-EVENT-0367	Il processo di aggiunta del router *job_ID* è stato avviato nel gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0368	Il processo di aggiunta del router è stato completato correttamente. *job_ID* Un nuovo router con ID di subcluster *number* è stato aggiunto al gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0369	Il processo di aggiunta del router non *job_ID* è riuscito nel gruppo di shard DB. *name*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
manutenzione	RDS-EVENT-0394	Il processo di aggiunta del router *job_ID* è stato annullato nel gruppo di shard DB. *name*	Nessuno
manutenzione	RDS-EVENT-0395	Il lavoro Split Shard *job_ID* è stato annullato nel gruppo di shard DB. *name*	Nessuno
notification	RDS-EVENT-0321	Inizializzazione dell'infrastruttura per il job Data Load.	Nessuno
notification	RDS-EVENT-0322	Il processo di caricamento dei dati è in corso.	Nessuno
notification	RDS-EVENT-0323	Processo di caricamento dei dati completato con successo.	Nessuno
notification	RDS-EVENT-0325	Annullamento del processo di caricamento dei dati secondo la richiesta del cliente.	Nessuno
notification	RDS-EVENT-0326	Processo di caricamento dati annullato.	Nessuno

Eventi RDS Proxy

La tabella seguente riporta la categoria di eventi e un elenco di eventi applicabili quando il tipo di origine è un proxy RDS.

Categoria	ID evento RDS	Messaggio	Note
modifica della configurazione	RDS-EVENT-0204	Proxy DB modificato da RDS. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0207	RDS ha modificato il punto finale del proxy DB. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0213	RDS ha rilevato l'aggiunta dell'istanza DB e l'ha aggiunta automaticamente al gruppo di destinazione del proxy DB. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0214	RDS ha rilevato l'eliminazione dell'istanza DB *name* e l'ha rimossa automaticamente dal gruppo target *name* del proxy DB. *name*	Nessuno
modifica della configurazione	RDS-EVENT-0215	RDS ha rilevato l'eliminazione del cluster DB *name* e lo ha rimosso automaticamente dal gruppo *name* di destinazione del proxy DB. *name*	Nessuno
creazione	RDS-EVENT-0203	Proxy DB creato da RDS. *name*	Nessuno
creazione	RDS-EVENT-0206	Endpoint creato da RDS *name* per il proxy DB. *name*	Nessuno
eliminazione	RDS-EVENT-0205	RDS ha eliminato il proxy DB. *name*	Nessuno

Categoria	ID evento RDS	Messaggio	Note
eliminazione	RDS-EVENT-0208	RDS ha eliminato l'endpoint *name* per il proxy DB. *name*	Nessuno
errore	RDS-EVENT-0243	RDS non è riuscito a fornire la capacità per il proxy *name* perché non ci sono abbastanza indirizzi IP disponibili nelle sottoreti :. *name* Per risolvere il problema, assicurarsi che le sottoreti abbiano il numero minimo di indirizzi IP non utilizzati come consigliato nella documentazione di Server proxy per Amazon RDS.	Per determinare il numero consigliato per la classe di istanza, consulta Pianificazione della capacità degli indirizzi IP.
errore	RDS-EVENT-0275	RDS ha limitato alcune connessioni al proxy DB. *name* Il numero di richieste di connessione simultane e dal client al proxy ha superato il limite.	Nessuno

Eventi di implementazione blu/verde

Nella tabella seguente sono indicati la categoria di evento e un elenco di eventi quando l'implementazione blu/verde è un tipo di origine.

Per ulteriori informazioni sulle implementazioni blu/verde, consulta Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Categoria	ID evento Amazon RDS	Messaggio	Note
creazione	RDS-EVENT-0244	Le attività di implement azione blu/verde sono state completate. È possibile apportare ulteriori modifiche ai database dell'ambiente verde o passare all'imple mentazione.	Nessuno
errore	RDS-EVENT-0245	La creazione della distribuz ione blu/verde non è riuscita perché. *reason*	Nessuno
eliminazione	RDS-EVENT-0246	L'implementazione blu/ verde è stata eliminata.	Nessuno
notification	RDS-EVENT-0247	Switchover da inizio a inizio. *blue green*	Nessuno
notification	RDS-EVENT-0248	Lo switchover è stato completato per l'impleme ntazione blu/verde.	Nessuno
errore	RDS-EVENT-0249	Lo switchover è stato annullato per l'impleme ntazione blu/verde.	Nessuno
notification	RDS-EVENT-0259	Passaggio dal cluster DB di replica lettura all'avvio. *blue green*	Nessuno
notification	RDS-EVENT-0260	*blue green blueblue-old*Rinominato *green* in *blue* e in.	Nessuno
errore	RDS-EVENT-0261	Il passaggio dal cluster DB di a è stato annullato	Nessuno

Categoria	ID evento Amazon RDS	Messaggio	Note
		a causa di. *blue green reason*	
notification	RDS-EVENT-0311	È iniziata la sincroniz zazione della sequenza per il passaggio del cluster DB a. *blue green* Quando si utilizzano le sequenze lo switchover può comportare tempi di inattività prolungati.	Nessuno
notification	RDS-EVENT-0312	La sincronizzazione della sequenza per il passaggio del cluster DB a è stata completata. *blue green*	Nessuno
errore	RDS-EVENT-0314	La sincronizzazione della sequenza per il passaggio del cluster DB *blue* a *green* è stata annullata perché le sequenze non sono state sincronizzate.	Nessuno
notification	RDS-EVENT-0409	*message*	Nessuno

Monitoraggio dei file di log di Aurora

Ogni motore di RDS database genera log a cui è possibile accedere per il controllo e la risoluzione dei problemi. Il tipo di registri dipende dal motore di database.

Puoi accedere ai log del database per le istanze DB utilizzando AWS Management Console, the AWS Command Line Interface (AWS CLI) o Amazon. RDS API Non puoi visualizzare, controllare o scaricare registri delle transazioni.

> **Note**
>
> In alcuni casi, i log contengono dati nascosti. Pertanto, AWS Management Console potrebbe mostrare il contenuto in un file di registro, ma il file di registro potrebbe essere vuoto quando lo scarichi.

Argomenti

- Visualizzazione ed elenco dei file di log del database
- Download di un file di log di database
- Controllo di un file di log di database
- Pubblicazione dei log del database su Amazon CloudWatch Logs
- Lettura del contenuto del file di registro utilizzando REST
- File di log del database Aurora MySQL
- File di log del database Aurora PostgreSQL

Visualizzazione ed elenco dei file di log del database

Puoi visualizzare i file di log del database per il tuo motore Aurora DB utilizzando il. AWS Management Console Puoi elencare quali file di registro sono disponibili per il download o il monitoraggio utilizzando Amazon AWS CLI o Amazon RDSAPI.

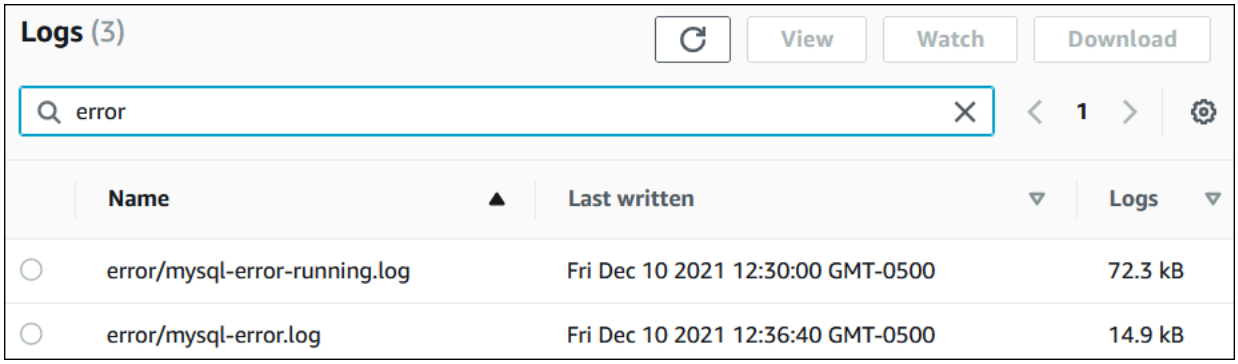
> **Note**
>
> Non è possibile visualizzare i file di log per i cluster Aurora Serverless v1 DB nella RDS console. Tuttavia, puoi visualizzarli nella CloudWatch console Amazon all'indirizzo https://console.aws.amazon.com/cloudwatch/.

Console

Per visualizzare un file di log di database

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome dell'istanza di database che ha il file di log che si desidera visualizzare.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Scorrere fino alla sezione Logs (Log).
6. (Opzionale) Inserisci un termine di ricerca per filtrare i risultati.

   Nell'esempio seguente vengono elencati i registri filtrati dal testo **`error`**.



7. Scegli il log che desideri visualizzare, quindi seleziona View (Visualizza).

AWS CLI

Per elencare i file di log del database disponibili per un'istanza DB, usa il AWS CLI `describe-db-log-files`comando.

Il seguente esempio restituisce un elenco di file di log per un'istanza database denominata `my-db-instance`.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

RDS API

Per elencare i file di log del database disponibili per un'istanza DB, utilizza l'RDSAPI`DescribeDBLogFiles`azione Amazon.

Download di un file di log di database

È possibile utilizzare AWS Management Console AWS CLI, o API per scaricare un file di registro del database.

Console

Per scaricare un file di log di database

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome dell'istanza di database che ha il file di log che si desidera visualizzare.
4. Scegliere la scheda Logs & events (Log ed eventi).
5. Scorrere fino alla sezione Logs (Log).
6. Nella sezione Logs (Log), selezionare il pulsante accanto al log che si desidera scaricare, quindi selezionare Download (Scarica).
7. Aprire il menu contestuale (clic con il tasto destro del mouse) per il collegamento fornito, quindi scegliere Save Link As (Salva collegamento come). Immettere l'ubicazione in cui si intende salvare il file di log, quindi scegliere Save (Salva).

AWS CLI

Per scaricare un file di registro del database, usa il AWS CLI comando `download-db-log-file-portion`. Per impostazione predefinita, questo comando scarica la porzione più recente di un file di log. Tuttavia, puoi scaricare un intero file specificando il parametro `--starting-token 0`.

L'esempio seguente mostra come scaricare l'intero contenuto di un file di registro chiamato log/ERROR .4 e memorizzarlo in un file locale chiamato errorlog.txt.

Example

PerLinux, omacOS: Unix

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

Per Windows:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

RDS API

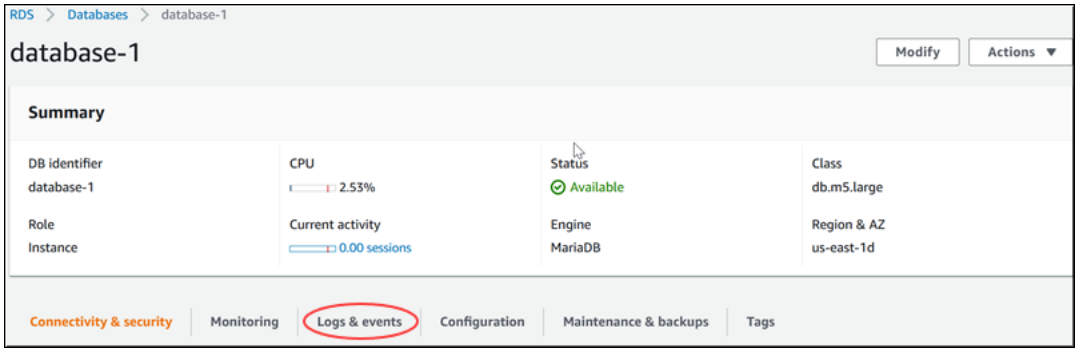
Per scaricare un file di registro del database, usa l'RDSAPIDownloadDBLogFilePortionazione Amazon.

Controllo di un file di log di database

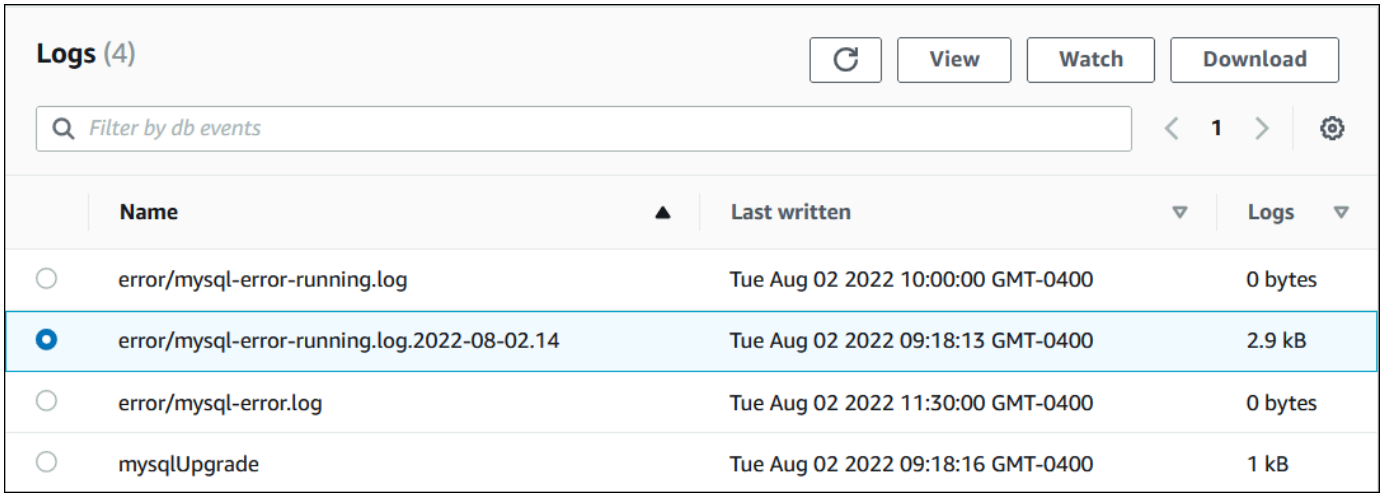
Guardare un file di registro del database equivale a mettere in coda il file su un UNIX sistema Linux. Puoi controllare un file di registro usando la AWS Management Console. RDSaggiorna la coda del registro ogni 5 secondi.

Per controllare un file di log di database

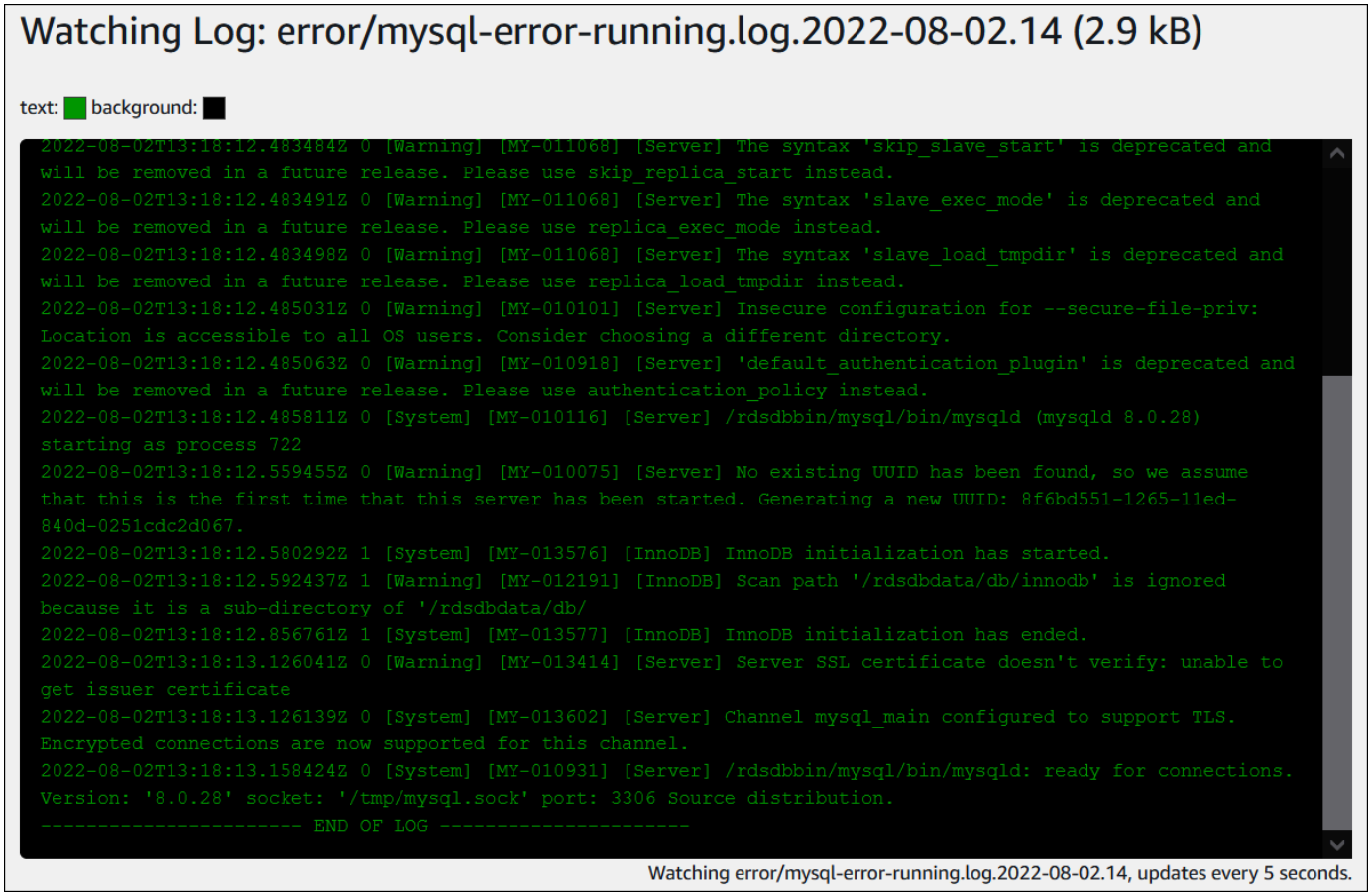
1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il nome dell'istanza di database che ha il file di log che si desidera visualizzare.
4. Scegliere la scheda Logs & events (Log ed eventi).



5. Nella sezione Logs (Log), scegliere un file di log, quindi selezionare Watch (Controlla).



RDSmostra la coda del registro, come nel seguente SQL esempio.



Pubblicazione dei log del database su Amazon CloudWatch Logs

In un database on-premise, i registri del database risiedono nel file system. Amazon RDS non fornisce accesso host ai registri del database sul file system del cluster database. Per questo motivo, Amazon RDS consente di esportare i log del database in Amazon CloudWatch Logs. Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log. È inoltre possibile archiviare i dati in un archivio altamente durevole e gestirli con CloudWatch Logs Agent.

Argomenti

- Panoramica dell'integrazione di RDS con Logs CloudWatch
- Decidere quali log pubblicare in Logs CloudWatch
- Specificare i log da pubblicare su Logs CloudWatch
- Ricerca e filtraggio dei log in Logs CloudWatch

Panoramica dell'integrazione di RDS con Logs CloudWatch

In CloudWatch Logs, un flusso di log è una sequenza di eventi di registro che condividono la stessa origine. Ogni fonte separata di log in CloudWatch Logs costituisce un flusso di log separato. Un gruppo di log è un gruppo di flussi di log che condividono le stesse impostazioni di conservazione, monitoraggio e controllo degli accessi.

Amazon Aurora esegue lo streaming continuo dei record di log del cluster database in un gruppo di log. Ad esempio, disponi di un gruppo di registri `/aws/rds/cluster/`*`cluster_name`*`/`*`log_type`* per ogni tipo di registro che pubblichi. Questo gruppo di log si trova nella stessa AWS regione dell'istanza del database che genera il log.

AWS conserva i dati di registro pubblicati in CloudWatch Logs per un periodo di tempo indefinito, a meno che non si specifichi un periodo di conservazione. Per ulteriori informazioni, consulta Modifica della conservazione dei dati di registro nei registri. CloudWatch

Decidere quali log pubblicare in Logs CloudWatch

Ogni motore di database RDS supporta il proprio set di registri. Per informazioni sulle opzioni per il motore di database, consulta i seguenti argomenti:

- the section called “Pubblicazione dei log di Aurora MySQL su Logs CloudWatch ”
- the section called “Pubblicazione dei log di Aurora PostgreSQL in Logs CloudWatch ”

Specificare i log da pubblicare su Logs CloudWatch

Puoi specificare quali registri pubblicare nella console. Assicurati di avere un ruolo collegato al servizio in (IAM). AWS Identity and Access Management Per ulteriori informazioni sui ruoli collegati al servizio, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

Per specificare i registri da pubblicare

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Esegui una delle operazioni seguenti:
   - Scegliere Crea database.
   - Scegli un database dall'elenco, quindi scegli Modify (Modifica).
4. In Logs exports (Esportazioni di log), scegli quali registri pubblicare.

Ricerca e filtraggio dei log in Logs CloudWatch

È possibile cercare le voci di registro che soddisfano criteri specifici utilizzando la CloudWatch console Logs. È possibile accedere ai log tramite la console RDS, che conduce alla console CloudWatch Logs, o direttamente dalla CloudWatch console Logs.

Per cercare registri RDS utilizzando la console RDS

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli un cluster database o un'istanza database.
4. Scegliere Configuration (Configurazione).
5. In Published logs (Log pubblicati), scegli il registro del database che desideri visualizzare.

Per cercare nei registri RDS utilizzando la console Logs CloudWatch

1. Apri la console all' CloudWatch indirizzo. https://console.aws.amazon.com/cloudwatch/
2. Nel pannello di navigazione, selezionare Log groups (Gruppi di log).
3. Nella casella del filtro, immetti **`/aws/rds`**.
4. In Log Groups (Gruppi di log), seleziona il nome del gruppo di log contenente il flusso di log da cercare.
5. In Log Streams (Flussi di log), seleziona il nome del flusso di log da cercare.
6. In Eventi di log, immettere la sintassi del filtro da utilizzare.

Per ulteriori informazioni, consulta la sezione Ricerca e filtraggio dei dati di log nella Amazon CloudWatch Logs User Guide. Per un tutorial sul blog che spiega come monitorare i log RDS, consulta Crea un monitoraggio proattivo del database per Amazon RDS con Amazon Logs CloudWatch , AWS Lambda e Amazon SNS.

Lettura del contenuto del file di registro utilizzando REST

Amazon RDS fornisce un REST endpoint che consente l'accesso ai file di registro delle istanze DB. Ciò è utile se devi scrivere un'applicazione per lo streaming dei contenuti dei file di RDS log di Amazon.

La sintassi è:

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

I parametri seguenti sono obbligatori:

- *DBInstanceIdentifier* — Nome dell'istanza database che contiene il file di log che vuoi scaricare.
- *LogFileName* — Nome del file di log da scaricare.

La risposta contiene i contenuti del file di log richiesto, come stream.

L'esempio seguente scarica il file di registro denominato log/ ERROR .6 per l'istanza DB denominata sample-sql nella regione us-west-2.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

Se specifichi un'istanza database non esistente, otterrai il seguente errore:

- `DBInstanceNotFound`—*DBInstanceIdentifier* non fa riferimento a un'istanza database esistente. (codice di stato: 404) HTTP

File di log del database Aurora MySQL

Puoi monitorare i log di Aurora MySQL direttamente tramite la console Amazon RDS, l'API Amazon RDS o. AWS CLI AWS SDKs Puoi anche eseguire l'accesso ai log MySQL indirizzando i log a una tabella del database nel database principale e facendo una ricerca in tale tabella. Puoi utilizzare la utility mysqlbinlog per scaricare un log binario.

Per ulteriori informazioni sulla visualizzazione, il download e la visione di log di database basati su file, consulta Monitoraggio dei file di log di Aurora.

Argomenti

- Panoramica dei registri di database Aurora MySQL
- Invio dell'output di Aurora My SQL log alle tabelle
- Configurazione di Aurora per database Single-AZ
- Accesso ai log binari MySQL

Panoramica dei registri di database Aurora MySQL

Puoi monitorare i seguenti tipi di file di registro Aurora MySQL:

- Log di errori
- Log delle query lente
- Log generale
- Log di audit
- Registro delle istanze
- Registro degli errori di autenticazione del database IAM

Il registro degli errori Aurora MySQL viene generato per impostazione predefinita. È possibile generare la query lenta e i log generali impostando i parametri nel gruppo di parametri di database.

Argomenti

- Registri degli errori Aurora MySQL
- Registri generali e delle query lente di Aurora MySQL
- Registro di verifica di Aurora MySQL
- Registro delle istanze MySQL di Aurora

- Rotazione e conservazione dei registri per Autora MySQL
- Pubblicazione dei log di Aurora MySQL su Amazon Logs CloudWatch

Registri degli errori Aurora MySQL

Aurora MySQL scrive errori nel file `mysql-error.log`. Ogni file di log ha l'ora di creazione (in UTC) accodata al nome. I file di log hanno anche un timestamp che ti aiuta a determinare quando le voci del log sono state scritte.

Aurora MySQL scrive nel registro degli errori solo durante l'avvio, l'arresto e quando si verificano errori. Un'istanza database può andare avanti ore senza che ci siano nuove voci scritte nel file di log degli errori. Se non vedi voci recenti, significa che il server non ha riscontrato errori che generano una voce di registro.

In base alla progettazione, i registri degli errori vengono filtrati in modo da visualizzare solo eventi imprevisti come errori. Tuttavia, i registri degli errori contengono anche altre informazioni sul database, ad esempio l'avanzamento della query, che non vengono visualizzate. Pertanto, anche senza errori effettivi, la dimensione dei registri degli errori potrebbe aumentare a causa delle attività del database in corso. Anche se potresti vedere una certa dimensione in byte o kilobyte per i log degli errori contenuti in AWS Management Console, potrebbero avere 0 byte quando li scarichi.

Aurora MySQL scrive `mysql-error.log` su disco ogni 5 minuti. Aggiunge il contenuto del registro a `mysql-error-running.log`.

Aurora MySQL ruota il file `mysql-error-running.log` ogni ora.

> **Note**
>
> Il periodo di conservazione dei log è diverso tra Amazon RDS e Aurora.

Registri generali e delle query lente di Aurora MySQL

Il registro delle query lente e il registro generale di Aurora MySQL possono essere scritti in un file o una tabella di database impostando i parametri nel gruppo parametri del database. Per informazioni sulla creazione e la modifica di un gruppo di parametri database, consulta . È necessario impostare questi parametri prima di poter visualizzare il registro delle query lente o il registro generale nella console Amazon RDS o utilizzando l'API Amazon RDS, l'interfaccia a riga di comando di Amazon RDS o. AWS SDKs

Puoi controllare la registrazione di Aurora MySQL utilizzando i parametri in questo elenco:

- `slow_query_log`: per creare il log delle query lente, imposta su 1. Il valore predefinito è 0.
- `general_log`: per creare il log generale, imposta su 1. Il valore predefinito è 0.
- `long_query_time`: per evitare che le query a esecuzione rapida vengano registrate nel registro delle query lente, specifica in secondi un valore per il runtime di query più breve da registrare. Il valore predefinito è 10 secondi, il minimo è 0 secondi. Se log_output = FILE, puoi specificare un valore in virgola mobile con risoluzione al microsecondo. Se log_output = TABLE, devi specificare un valore intero con risoluzione al secondo. Vengono registrate solo le query con runtime che supera il valore `long_query_time`. Ad esempio, impostando `long_query_time` su 0,1 si impedisce a tutte le query con tempo di esecuzione inferiore a 100 millisecondi di essere registrate.
- `log_queries_not_using_indexes`: per registrare tutte le query che non usano un indice sul log delle query lente, imposta su 1. Le query che non utilizzano un indice vengono registrate anche se il runtime è inferiore al valore del parametro `long_query_time`. Il valore predefinito è 0.
- `log_output` *`option`*: puoi specificare una delle seguenti opzioni per il parametro `log_output`.
  - TABLE `mysql.general_log` Scrive le query generali nella tabella – e le query lente nella tabella `mysql.slow_log`.
  - FILE – Scrive sia i log generali sia i log delle query lente nel file system.
  - NONE – Disabilita il logging.

  Per le versioni 2 e 3 di Aurora MySQL, l'impostazione predefinita per è. `log_output FILE`

Affinché i dati delle query lente vengano visualizzati in Amazon CloudWatch Logs, devono essere soddisfatte le seguenti condizioni:

- CloudWatch I log devono essere configurati per includere log di query lente.
- `slow_query_log`deve essere abilitato.
- `log_output` deve essere impostato su `FILE`.
- L'interrogazione deve richiedere più tempo del tempo configurato`long_query_time`.

Per ulteriori informazioni sui log delle query lente e i log generali, consulta i seguenti argomenti nella documentazione di MySQL:

- Log delle query lente
- Log delle query generali

Registro di verifica di Aurora MySQL

La registrazione di controllo per Aurora MySQL è denominata Advanced Auditing. Per attivare Advanced Auditing, imposta alcuni parametri del cluster database. Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.

Registro delle istanze MySQL di Aurora

Aurora crea un file di registro separato per le istanze DB con la pausa automatica abilitata. Questo file instance.log registra tutti i motivi per cui queste istanze DB non possono essere messe in pausa quando previsto. Per ulteriori informazioni sul comportamento dei file di registro delle istanze e sulla funzionalità di pausa automatica di Aurora, consulta Monitoraggio delle attività di pausa e ripresa di Aurora Serverless v2.

Rotazione e conservazione dei registri per Autora MySQL

Quando la registrazione è abilitata, Amazon Aurora ruota o elimina i file di registro Amazon a intervalli regolari. Questa è una misura preventiva per ridurre l'eventualità che un file di log molto grande comprometta l'uso del database o la performance. Autora MySQL gestisce la rotazione e l'eliminazione come segue:

- Le dimensioni del file di registro degli errori di Aurora MySQL sono limitate a un massimo del 15 per cento dello storage locale per un'istanza database. Per mantenere questa soglia, i log vengono ruotati automaticamente ogni ora. Aurora MySQL rimuove i registri dopo 30 giorni o quando è stato occupato il 15% dello spazio su disco. Se le dimensioni del file di log combinato superano tale soglia dopo la rimozione dei file di log più vecchi, i file di log più grandi vengono eliminati fino a che le dimensioni del file di log non rimangono inferiori alla soglia.
- Aurora MySQL rimuove i log di audit, generali, di query lente dopo 24 ore o quando è stato occupato il 15% dello spazio di archiviazione.
- Quando la registrazione `FILE` è abilitata, i file di registro generale e delle query lente vengono esaminati ogni ora e quelli più vecchi di 24 ore vengono eliminati. In alcuni casi, la dimensione del file di registro combinato restante dopo l'eliminazione supera la soglia del 15 per cento di spazio locale di un'istanza database. In questi casi, i file di log più vecchi vengono eliminati fino a che le dimensioni del file di log non rimangono inferiori alla soglia.
- Quando la registrazione `TABLE` è abilitata, le tabelle di registro non vengono ruotate o eliminate. Le tabelle di registro vengono troncate quando la dimensione di tutti i registri combinati è troppo grande. Puoi iscriverti alla categoria di `low storage` eventi per ricevere una notifica quando le tabelle di log devono essere ruotate o eliminate manualmente per liberare spazio. Per ulteriori informazioni, consulta Utilizzo delle notifiche di RDS eventi di Amazon.

Puoi ruotare la tabella `mysql.general_log` chiamando manualmente la procedura `mysql.rds_rotate_general_log`. Puoi ruotare la tabella `mysql.slow_log` chiamando la procedura `mysql.rds_rotate_slow_log`.

Quando ruoti manualmente le tabelle di registro, la tabella di registro corrente viene copiata in una tabella di registro di backup e le voci nella tabella di registro corrente vengono eliminate. Se esiste già una tabella di log di backup, questa viene eliminata prima che la tabella di log corrente sia copiata nel backup. Puoi eseguire una query sulla tabella di log di backup, se necessario. La tabella di log di backup per la tabella `mysql.general_log` è denominata `mysql.general_log_backup`. La tabella di log di backup per la tabella `mysql.slow_log` è denominata `mysql.slow_log_backup`.

- I registri di controllo Aurora MySQL vengono ruotati quando la dimensione del file raggiunge i 100 MB e rimossi dopo 24 ore.
- Amazon RDS ruota i file di log degli errori di autenticazione del database IAM di dimensioni superiori a 10 MB. Amazon RDS rimuove i file di log degli errori di autenticazione del database IAM più vecchi di cinque giorni o più grandi di 100 MB.

Per utilizzare i log dalla console Amazon RDS, dall'API Amazon RDS, dalla CLI di Amazon RDS oppure AWS SDKs imposta il parametro su FILE. `log_output` Analogamente al registro degli errori di Aurora MySQL, questi file di registro vengono ruotati ogni ora. I file di log che sono stati generati durante le precedenti 24 ore vengono conservati. Il periodo di conservazione è diverso tra Amazon RDS e Aurora.

Pubblicazione dei log di Aurora MySQL su Amazon Logs CloudWatch

Puoi configurare il tuo cluster Aurora MySQL DB per pubblicare i dati di log in un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log e utilizzarli CloudWatch per creare allarmi e visualizzare metriche. È possibile utilizzare CloudWatch Logs per archiviare i record di registro in un archivio altamente durevole. Per ulteriori informazioni, consulta Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch .

Invio dell'output di Aurora My SQL log alle tabelle

Puoi indirizzare il log generale e il log delle query lente alle tabelle sull'istanza database creando un gruppo di parametri del database e impostando il parametro server `log_output` su `TABLE`. Le query generali vengono quindi registrate sulla tabella `mysql.general_log`, mentre le query lente vengono registrate sulla tabella `mysql.slow_log`. Puoi eseguire query sulle tabelle per avere

accesso alle informazioni di log. L'abilitazione di questa registrazione aumenta il numero di dati scritti sul database, il che potrebbe compromettere le performance.

Sia il log generale che quello delle query lente sono disattivati per impostazione predefinita. Per abilitare la registrazione sulle tabelle devi impostare anche i parametri server `general_log` e `slow_query_log` su `1`.

Le tabelle di log continuano a crescere fino a che le rispettive attività di registrazione non vengono disattivate eseguendo la reimpostazione del parametro appropriato su `0`. Spesso nel corso del tempo si accumulano grandi quantità di dati che possono usare una percentuale considerevole dello spazio di archiviazione assegnato. Amazon Aurora non consente di troncare le tabelle di log, ma è possibile spostarne il contenuto. La rotazione delle tabelle ne salva il contenuto in una tabella di backup e crea una nuova tabella di log vuota. Puoi ruotare manualmente le tabelle di log con le seguenti procedure a riga di comando, nelle quali il prompt dei comandi è indicato da `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

Per rimuovere completamente i dati vecchi e recuperare lo spazio del disco, chiama la procedura adeguata due volte in successione.

Configurazione di Aurora per database Single-AZ

Il log binario è un insieme di file di log che contengono informazioni sulle modifiche apportate ai dati di un'istanza server Aurora MySQL. Il log binario contiene informazioni come le seguenti:

- Eventi che descrivono le modifiche al database come la creazione di tabelle o la modifica di righe
- Informazioni sulla durata di ogni istruzione che ha aggiornato i dati
- Eventi per istruzioni che avrebbero potuto aggiornare i dati ma non l'hanno fatto

Il log binario registra le istruzioni inviate durante la replica. È inoltre necessario per alcune operazioni di ripristino. Per ulteriori informazioni, consulta la pagina relativa al log binario nella documentazione di MySQL.

I registri binari sono accessibili solo dall'istanza database principale, non dalle repliche.

MySQL su Amazon Aurora supporta i formati di logging binario basati su righe, basati su istruzioni e misti. Si consiglia il formato misto a meno che non sia necessario un formato binlog specifico. Per

dettagli sui diversi formati di log binari di Aurora MySQL, vedere Formati di registrazione binaria nella documentazione di MySQL.

Se pianifichi di utilizzare la replica, il formato di logging binario è importante in quanto determina il record delle modifiche dei dati che viene registrato nella sorgente e inviato ai target della replica. Per ulteriori informazioni sui vantaggi e sugli svantaggi dei vari formati di logging binario per la replica, consulta la pagina relativa a vantaggi e svantaggi della replica basata su istruzioni e basata su riga nella documentazione di MySQL.

> **Important**
>
> Con MySQL 8.0.34, MySQL ha reso obsoleto il parametro. `binlog_format` Nelle versioni successive di MySQL, MySQL prevede di rimuovere il parametro e supportare solo la replica basata su righe. Di conseguenza, consigliamo di utilizzare la registrazione basata su righe per le nuove configurazioni di replica MySQL. Per ulteriori informazioni, vedere binlog_format nella documentazione di MySQL.
> Le versioni MySQL 8.0 e 8.4 accettano il parametro. `binlog_format` Quando si utilizza questo parametro, MySQL emette un avviso di deprecazione. In una delle future release principali, MySQL rimuoverà il parametro. `binlog_format`
> La replica basata sulle istruzioni può causare incoerenze tra l'cluster database di origine e una replica di lettura. Per ulteriori informazioni, consulta la pagina relativa alla determinazione delle istruzioni sicure e non sicure nel logging binario nella documentazione MySQL.
> Abilitando la registrazione binaria, aumenta il numero delle operazioni I/O di scrittura sul disco nel cluster di database. È possibile monitorare l'utilizzo degli IOPS con la metrica. `VolumeWriteIOPs` CloudWatch

Per impostare il formato di registrazione binaria MySQL

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Scegliete il gruppo di parametri del cluster DB, associato al cluster di DB, che desiderate modificare.

   Non è consentito modificare un gruppo di parametri predefinito. Se l'cluster database è usata da un gruppo di parametri predefinito, creare un nuovo gruppo di parametri e associarlo all'cluster database.

Per ulteriori informazioni sui gruppi di parametri, consulta .

4. Da Azioni, scegli Modifica.
5. Imposta il parametro `binlog_format` sul formato di registrazione binaria scelto (`ROW`, `STATEMENT` o `MIXED`). Puoi anche utilizzare il valore `OFF` per disattivare la registrazione binaria.

   > **Note**
   >
   > L'impostazione `binlog_format` su `OFF` nel gruppo di parametri del cluster DB disabilita la variabile di `log_bin` sessione. Ciò disabilita la registrazione binaria sul cluster Aurora MySQL DB, che a sua volta reimposta la variabile di `binlog_format` sessione al valore predefinito di nel database. `ROW`

6. Scegliere Salva modifiche per salvare gli aggiornamenti applicati al gruppo di parametri cluster database.

Dopo aver eseguito questi passaggi, è necessario riavviare l'istanza di scrittura nel cluster database per applicare le modifiche. In Aurora MySQL versione 2.09 e precedenti, quando si riavvia l'istanza di scrittura, vengono riavviate anche tutte le istanze di lettura nel cluster database. In Aurora MySQL versione 2.10 e successive, è necessario riavviare tutte le istanze di lettura manualmente. Per ulteriori informazioni, consulta Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB.

> **Important**
>
> La modifica di un gruppo di parametri cluster di database influisce su tutti i cluster di database che utilizzano tale gruppo di parametri. Se si desidera specificare diversi formati di registrazione binaria per diversi cluster Aurora MySQL DB AWS in una regione, i cluster DB devono utilizzare diversi gruppi di parametri del cluster DB. Questi gruppi di parametri identificano diversi formati di logging. Assegnare il gruppo di parametri cluster di database appropriato a ciascun cluster di database. Per ulteriori informazioni sui parametri Aurora MySQL, consulta Parametri di configurazione Aurora MySQL.

Accesso ai log binari MySQL

Puoi utilizzare la utility mysqlbinlog per il download o lo streaming di log binari dalle istanze database RDS for MySQL. Il log binario viene scaricato sul computer locale dove è possibile eseguire operazioni come la riproduzione del log tramite utility mysql. Per ulteriori informazioni sull'uso dell'utilità mysqlbinlog, consulta Utilizzo di mysqlbinlog per il backup di file di log binari nella documentazione di MySQL.

Per eseguire la utility mysqlbinlog su un'istanza Amazon RDS usa le seguenti opzioni:

- `--read-from-remote-server`: obbligatorio
- `--host`: il nome DNS dall'endpoint dell'istanza.
- `--port`: la porta utilizzata dall'istanza.
- `--user`: un utente MySQL al quale è stata concessa l'autorizzazione `REPLICATION SLAVE`.
- `--password`: la password dell'utente MySQL oppure ometti un valore di password affinché l'utilità richieda una password.
- `--raw`: scarica il file in formato binario.
- `--result-file`: il file locale per riceve l'output raw.
- `--stop-never`: trasmette in streaming i file di log binari.
- `--verbose`: quando utilizzi il formato binlog `ROW`, includi questa opzione per visualizzare gli eventi di riga come istruzioni pseudo-SQL. Per ulteriori informazioni sull'opzione `--verbose`, consulta Visualizzazione degli eventi di riga di mysqlbinlog nella documentazione di MySQL.
- Specifica il nome di uno o più file di log binari. Per ottenere l'elenco dei log disponibili, utilizza il comando SQL `SHOW BINARY LOGS`.

Per ulteriori informazioni sulle opzioni di mysqlbinlog, consulta Utilità mysqlbinlog per l'elaborazione di file di log binari nella documentazione di MySQL.

Gli esempi seguenti mostrano come utilizzare l'utilità mysqlbinlog.

In Linux, macOS, oppure Unix:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
```

```
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

In Windows:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

I log binari devono rimanere disponibili sull'istanza DB affinché l'utilità mysqlbinlog possa accedervi. Per garantirne la disponibilità, utilizzate la mysql.rds_set_configuration stored procedure e specificate un periodo di tempo sufficiente per scaricare i log. Se questa configurazione non è impostata, Amazon RDS elimina i log binari il prima possibile, causando lacune nei log binari recuperati dall'utilità mysqlbinlog.

L'esempio seguente imposta il periodo di conservazione su 1 giorno.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

Per visualizzare l'impostazione attuale, utilizza la procedura archiviata mysql.rds_show_configuration.

```
call mysql.rds_show_configuration;
```

File di log del database Aurora PostgreSQL

È possibile monitorare i seguenti tipi di file di registro di Aurora PostgreSQL:

- Registro PostgreSQL
- Registro dell'istanza
- Registro degli errori di autenticazione del database IAM

> **Note**
>
> Per ulteriori informazioni sull'attivazione dell'autenticazione del database IAM, consulta. Abilitazione e disabilitazione dell'autenticazione database IAM

Aurora PostgreSQL registra le attività del database nel file di log PostgreSQL predefinito. Per un'istanza database PostgreSQL on-premise, questi messaggi vengono archiviati localmente in `log/postgresql.log`. Per un cluster database Aurora PostgreSQL, il file di log è disponibile nel cluster Aurora. Questi log sono accessibili anche tramite AWS Management Console, dove è possibile visualizzarli o scaricarli. Il livello di registrazione predefinito rileva gli errori di accesso, gli errori irreversibili del server, i deadlock e gli errori delle query.

Per ulteriori informazioni su come visualizzare, scaricare e guardare i registri di database basati su file, consulta Monitoraggio dei file di log di Aurora. Per ulteriori informazioni sui registri PostgreSQL, consulta Working with Amazon RDS and Aurora PostgreSQL logs: Part 1 (Utilizzo dei registri RDS e Aurora PostgreSQL: parte 1) e Working with Amazon RDS and Aurora PostgreSQL logs: Part 2 (Utilizzo dei registri RDS e Aurora PostgreSQL: parte 2).

Oltre ai log PostgreSQL standard trattati in questo argomento, Aurora PostgreSQL supporta anche l'estensione di audit PostgreSQL (`pgAudit`). La maggior parte dei settori regolamentati e degli enti governativi deve mantenere un log di audit o un audit trail delle modifiche apportate ai dati per conformità ai requisiti legali. Per informazioni sull'installazione e sull'utilizzo di pgAudit, consulta Utilizzo pgAudit per registrare l'attività del database.

Aurora crea un file di registro separato per le istanze DB con la pausa automatica abilitata. Questo file instance.log registra tutti i motivi per cui queste istanze DB non possono essere messe in pausa quando previsto. Per ulteriori informazioni sul comportamento dei file di registro delle istanze e sulla funzionalità di pausa automatica di Aurora, consulta Monitoraggio delle attività di pausa e ripresa di Aurora Serverless v2.

Argomenti

- Parametri per la registrazione in Aurora PostgreSQL RDS per
- Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL

Parametri per la registrazione in Aurora PostgreSQL RDS per

È possibile personalizzare il comportamento di registrazione per il cluster database Aurora PostgreSQL modificando vari parametri. Nella tabella seguente sono riportati, tra le altre impostazioni, i parametri che stabiliscono la durata di archiviazione dei log, quando ruotarli e se l'output del log è in formato CSV (valori separati da virgole). Puoi anche trovare l'output di testo inviato a STDERR, tra le altre impostazioni. Per modificare le impostazioni per i parametri modificabili, utilizza un gruppo di parametri del·cluster database personalizzato per il cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta .

Parametro	Predefinito	Descrizione
log_destination	stderr	Imposta il formato di output per il registro. L'impostazione predefinita è `stderr`, ma puoi anche specificare il formato CSV aggiungendo `csvlog` all'impostazione. Per ulteriori informazioni, consulta Impostazione della destinazione del registro (`stderr`, `csvlog`).
log_filename	postgresql.log.%Y-%m-%d-%H%M	Specifica il modello per il nome del file di log. Oltre al valore predefinito, questo parametro supporta `postgresql.log.%Y-%m-%d` e `postgresql.log.%Y-%m-%d-%H` per il modello del nome del file. Per Aurora PostgreSQL versione 17.4 e successive, non è possibile modificare questo parametro.
log_line_prefix	%t:%r:%u@%d:[%p]:	Definisce il prefisso per ogni riga di log che viene scritta in `stderr`, per annotare l'ora (%t), l'host remoto (%r), l'utente (%u), il database (%d) e l'ID del processo (%p).

Parametro	Predefinito	Descrizione
log_rotation_age	60	I minuti dopo i quali il file di log viene ruotato automaticamente. È possibile modificare questo valore nell'intervallo compreso tra 1 e 1440 minuti. Per ulteriori informazioni, consulta Impostazione della rotazione dei file di log.
log_rotation_size	–	La dimensione (KB) che stabilisce la rotazione automatica del log. È possibile modificare questo valore nell'intervallo compreso tra 50.000 e 1.000.000 di kilobyte. Per ulteriori informazioni, consulta Impostazione della rotazione dei file di log.
rds.log_retention_period	4320	I registri PostgreSQL più vecchi del numero di minuti specificato vengono eliminati. Il valore di default di 4.320 minuti elimina i file di log dopo 3 giorni. Per ulteriori informazioni, consulta Impostazione del periodo di retention dei log.

Per identificare i problemi dell'applicazione, puoi cercare fallimenti di query, errori di accesso, deadlock ed errori irreversibili del server nel registro. Ad esempio, supponi di convertire un'applicazione legacy da Oracle ad Aurora PostgreSQL, ma non tutte le query sono state convertite correttamente. Queste query formattate in modo errato generano messaggi di errore nei registri che puoi utilizzare per identificare i problemi. Per ulteriori informazioni sulla registrazione delle query, consulta Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL.

Negli argomenti seguenti sono disponibili informazioni su come impostare vari parametri che controllano i dettagli di base dei log PostgreSQL.

Argomenti

- Impostazione del periodo di retention dei log
- Impostazione della rotazione dei file di log
- Impostazione della destinazione del registro (stderr, csvlog)
- Informazioni sul parametro log_line_prefix

Impostazione del periodo di retention dei log

Il parametro `rds.log_retention_period` specifica per quanto tempo il cluster database Aurora PostgreSQL conserva i file di log. L'impostazione predefinita è 3 giorni (4.320 minuti), ma è possibile impostare qualsiasi valore compreso tra 1 giorno (1.440 minuti) e 7 giorni (10.080 minuti). Assicurati che il cluster database Aurora PostgreSQL abbia spazio di archiviazione sufficiente per contenere i file di log per il periodo di tempo specificato.

Ti consigliamo di pubblicare regolarmente i log su Amazon CloudWatch Logs in modo da poter visualizzare e analizzare i dati di sistema molto tempo dopo che i log sono stati rimossi dal cluster Aurora PostgreSQL DB. Per ulteriori informazioni, consulta Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs CloudWatch . Dopo aver impostato la CloudWatch pubblicazione, Aurora elimina un registro solo dopo la pubblicazione su Logs. CloudWatch

Amazon Aurora comprime i log PostgreSQL meno recenti quando lo spazio di archiviazione per l'istanza database raggiunge una determinata soglia. Aurora comprime i file utilizzando l'utilità di compressione gzip. Per ulteriori informazioni, consulta il sito web di gzip.

Quando lo spazio di archiviazione per l'istanza database si riduce e tutti i log disponibili sono compressi, si riceve un avviso del tipo seguente:

```
Warning: local storage for PostgreSQL log files is critically low for
this Aurora PostgreSQL instance, and could lead to a database outage.
```

Se lo spazio di archiviazione non è sufficiente, Aurora potrebbe eliminare i log PostgreSQL compressi prima della fine del periodo di conservazione specificato. Se ciò accade, viene visualizzato un messaggio simile al seguente:

```
The oldest PostgreSQL log files were deleted due to local storage constraints.
```

Impostazione della rotazione dei file di log

Per impostazione predefinita, nuovi file di log vengono creati da Aurora ogni ora. La tempistica è controllata dal parametro `log_rotation_age`. Questo parametro ha un valore predefinito di 60 (minuti), ma è possibile impostarlo su qualsiasi valore tra 1 minuto e 24 ore (1.440 minuti). Al momento della rotazione, viene creato un nuovo file di log distinto. Il file è denominato in base al modello specificato dal parametro `log_filename`.

I file di log possono anche essere ruotati in base alle loro dimensioni, come specificato dal parametro `log_rotation_size`. Questo parametro specifica che il log deve essere ruotato quando raggiunge

la dimensione specificata (in kilobyte). Il valore di timeout predefinito per `log_rotation_size` è di 100.000 kB (kilobyte) per un cluster di database Aurora PostgreSQL, ma puoi impostare questo parametro a qualsiasi valore compreso tra 50.000 e 1.000.000 kilobyte.

I nomi dei file di registro si basano sul modello di nome di file specificato nel parametro `log_filename`. Le impostazioni disponibili per questo parametro sono le seguenti:

- `postgresql.log.%Y-%m-%d` : formato predefinito per il nome del file di registro. Include l'anno, il mese e la data nel nome del file di log.
- `postgresql.log.%Y-%m-%d-%H`: include l'ora nel formato del nome del file di registro.
- `postgresql.log.%Y-%m-%d-%H%M`: include ora:minuto nel formato del nome del file di log.

Se imposti il parametro `log_rotation_age` su un valore inferiore a 60 minuti, imposta il parametro `log_filename` sul formato minuti.

Per ulteriori informazioni, consulta `log_rotation_age` e `log_rotation_size` nella documentazione di PostgreSQL.

Impostazione della destinazione del registro (**`stderr`**, **`csvlog`**)

Per impostazione predefinita, Aurora PostgreSQL genera i log in formato errore standard (stderr). Questo formato è l'impostazione predefinita per il parametro `log_destination`. Ogni messaggio ha un prefisso che utilizza il modello specificato nel parametro `log_line_prefix`. Per ulteriori informazioni, consulta Informazioni sul parametro log_line_prefix.

Aurora PostgreSQL può anche generare log in formato `csvlog`. Il formato `csvlog` è utile per analizzare i dati dei registri in formato CSV. Ad esempio, supponi di utilizzare l'estensione `log_fdw` per lavorare con i log come tabelle esterne. La tabella esterna creata sui file di log di `stderr` contiene una singola colonna con i dati degli eventi di log. Aggiungendo `csvlog` al parametro `log_destination`, ottieni il file di log in formato CSV con le demarcazioni per le diverse colonne della tabella esterna. In tal modo puoi ordinare e analizzare i log più facilmente.

Se specifichi `csvlog` per questo parametro, tieni presente che vengono generati entrambi i file `stderr` e `csvlog`. Ti consigliamo di monitorare lo spazio di archiviazione consumato dai registri tenendo conto di `rds.log_retention_period` e delle altre impostazioni che influiscono sull'archiviazione e sulla rotazione dei registri. Utilizzando `stderr` e `csvlog` lo spazio di archiviazione consumato dai registri aumenta più del doppio.

Se aggiungi `csvlog` a `log_destination` e vuoi ripristinare solo `stderr`, devi reimpostare il parametro. Per farlo, nella console Amazon RDS apri il gruppo di parametri del·cluster database personalizzato per la tua istanza. Scegli il parametro `log_destination`, seleziona Edit parameter (Modifica parametro), quindi Reset (Reimposta).

Per ulteriori informazioni sulla configurazione dei registri, consulta Utilizzo dei log Amazon RDS e Aurora PostgreSQL: Parte 1.

Informazioni sul parametro log_line_prefix

Il formato di `stderr` registro prefissa ogni messaggio di registro con i dettagli specificati dal parametro. `log_line_prefix` Il valore predefinito è:

```
%t:%r:%u@%d:[%p]:t
```

A partire dalla versione 16 di Aurora PostgreSQL, puoi anche scegliere:

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

Ogni voce di registro inviata a stderr include le seguenti informazioni in base al valore selezionato:

- `%t`— Ora di immissione del registro senza millisecondi
- `%m`— Ora di immissione del registro in millisecondi
- `%r` - Indirizzo dell'host remoto
- `%u@%d` - Nome utente @ nome del database
- `[%p]` - ID del processo, se disponibile
- `%l`— Numero di riga di registro per sessione
- `%e`— Codice di errore SQL
- `%s`— Timestamp di inizio del processo
- `%v`— ID della transazione virtuale
- `%x`— ID della transazione
- `%c`— ID della sessione
- `%q`— Terminatore non di sessione
- `%a`— Nome dell'applicazione

Attivazione della registrazione delle query per il cluster database Aurora PostgreSQL

È possibile raccogliere informazioni più approfondite sulle attività dei database, tra cui query, query in attesa di blocchi, checkpoint e molti altri dettagli impostando alcuni parametri elencati nella tabella seguente. Questo argomento illustra la registrazione delle query.

Parametro	Predefinito	Descrizione
log_connections	–	Registra ogni connessione riuscita. Per informazioni su come utilizzare questo parametro con `log_disconnections` per rilevare l'abbandono della connessio ne, consulta Gestione dell'abbandono della connessione Aurora PostgreSQL con pooling .
log_disconnections	–	Registra il momento in cui termina ciascuna sessione e la relativa durata. Per informazi oni su come utilizzare questo parametro con `log_connections` per rilevare l'abbandono della connessione, consulta Gestione dell'abba ndono della connessione Aurora PostgreSQL con pooling .
log_checkpoints		Non applicabile per Aurora PostgreSQL
log_lock_waits	–	Registra lunghe attese di lock. Per impostazi one predefinita, questo parametro non è impostato.
log_min_duration_s ample	–	Imposta il tempo (ms) minimo di esecuzione oltre il quale viene registrato un campione di istruzioni. La dimensione del campione viene impostata utilizzando il parametro `log_state ment_sample_rate` .
log_min_duration_s tatement	–	Viene registrata qualsiasi istruzione SQL che viene eseguita per il periodo specificato o per più tempo. Per impostazione predefinita, questo

Parametro	Predefinito	Descrizione
		parametro non è impostato. L'attivazione di questo parametro può aiutarti a trovare query non ottimizzate.
log_statement	–	Imposta il tipo di istruzioni registrate. Per impostazione predefinita, questo parametro non è impostato, ma puoi modificarlo in `all`, `ddl` o `mod` per specificare i tipi di istruzioni SQL che vuoi registrare. Se specifichi un valore diverso da `none` per questo parametro, dovrai adottare ulteriori misure per evitare l'esposizione delle password nei file di log. Per ulteriori informazioni, consulta Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query.
log_statement_sample_rate	–	La percentuale di istruzioni che superano il tempo specificato in `log_min_duration_sample` da registrare, espressa come valore in virgola mobile compreso tra 0,0 e 1,0.
log_statement_stats	–	Scrive le statistiche cumulative sulla prestazione nel registro del server.

Utilizzo della registrazione per trovare query lente

È possibile registrare istruzioni e query SQL per trovare le query con prestazioni lente. Puoi attivare questa funzionalità modificando le impostazioni nei parametri `log_statement` e `log_min_duration` come descritto in questa sezione. Prima di attivare la registrazione delle query per il cluster database Aurora PostgreSQL, è necessario essere consapevoli della possibile esposizione delle password nei registri e di come mitigare i rischi. Per ulteriori informazioni, consulta Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query.

Di seguito sono disponibili informazioni di riferimento sui parametri `log_statement` e `log_min_duration`.

log_statement

Questo parametro specifica il tipo di istruzioni SQL che devono essere inviate al registro. Il valore predefinito è `none`. Se modifichi questo parametro in `all`, `ddl` o `mod`, esegui le azioni consigliate per ridurre il rischio di esporre le password nei log. Per ulteriori informazioni, consulta Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query.

tutto

Registra tutte le istruzioni. Questa impostazione è consigliata per il debug.

ddl

Registra tutte le istruzioni DDL (Data Definition Language), come CREATE, ALTER, DROP e così via.

mod

Registra tutte le istruzioni DDL e DML (Data Manipulation Language), come INSERT, UPDATE e DELETE, che modificano i dati.

nessuno

Nessuna istruzione SQL viene registrata. Consigliamo questa impostazione per evitare il rischio di esporre le password nei registri.

log_min_duration_statement

Viene registrata qualsiasi istruzione SQL che viene eseguita per il periodo specificato o per più tempo. Per impostazione predefinita, questo parametro non è impostato. L'attivazione di questo parametro può aiutarti a trovare query non ottimizzate.

–1–2147483647

Il numero di millisecondi (ms) di runtime durante il quale un'istruzione viene registrata.

Per configurare la registrazione delle query

Questi passaggi presuppongono che il cluster database Aurora PostgreSQL utilizzi un gruppo di parametri cluster database personalizzato.

1. Imposta il parametro `log_statement` su `all`. L'esempio seguente mostra le informazioni scritte nel file `postgresql.log` con questa impostazione del parametro.

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
---------------------- END OF LOG ----------------------
```

2. Impostare il parametro `log_min_duration_statement`. L'esempio seguente mostra le informazioni scritte nel file `postgresql.log` quando il parametro è impostato su 1.

   Le query che superano la durata specificata nel parametro `log_min_duration_statement` vengono registrate. Di seguito viene riportato un esempio. Puoi visualizzare il file di log per il cluster database Aurora PostgreSQL nella console Amazon RDS.

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
```

```
---------------------- END OF LOG ----------------------
```

Riduzione del rischio di esposizione delle password quando si utilizza la registrazione delle query

Ti consigliamo di mantenere `log_statement` impostato su `none` per evitare di esporre le password. Se imposti `log_statement` su `all`, `ddl` o `mod`, ti consigliamo di eseguire una o più delle seguenti operazioni.

- Per il client, applica la crittografia delle informazioni sensibili. Per ulteriori informazioni, consulta Encryption Options (Opzioni di crittografia) nella documentazione di PostgreSQL. Usa le opzioni `ENCRYPTED` (e `UNENCRYPTED`) delle istruzioni `CREATE` e `ALTER`. Per ulteriori informazioni, consulta CREATE USER nella documentazione di PostgreSQL.
- Per il cluster database Aurora PostgreSQL, configura e usa l'estensione di audit PostgreSQL (pgAudit). Questa estensione oscura le informazioni sensibili nelle istruzioni CREATE e ALTER inviate al registro. Per ulteriori informazioni, consulta Utilizzo pgAudit per registrare l'attività del database.
- Limita l'accesso ai log. CloudWatch
- Utilizza meccanismi di autenticazione più efficaci come IAM.

Monitoraggio delle chiamate API Aurora in AWS CloudTrail

AWS CloudTrail è un AWS servizio che ti aiuta a controllare il tuo AWS account. AWS CloudTrail viene attivato per il tuo AWS account al momento della creazione. Per ulteriori informazioni in merito CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

Argomenti

- CloudTrail
- Amazon Aurora Voci dei file di RDS log di Aurora

CloudTrail

Tutte le RDS azioni Aurora vengono registrate da. CloudTrail CloudTrail

CloudTrail eventi

CloudTrail acquisisce le API chiamate per Amazon Amazon come eventi. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, data e ora dell'operazione, parametri della richiesta e così via. Gli eventi includono chiamate dalla RDS console Amazon e chiamate in codice alle RDS API operazioni Amazon.

Amazon Aurora RDS L'attività di viene registrata in un CloudTrail evento nella cronologia degli eventi. Puoi utilizzare la CloudTrail console per visualizzare gli ultimi 90 giorni di API attività ed eventi registrati in una AWS regione. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

CloudTrail sentieri

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Amazon Aurora , crea un percorso. Un trail è una configurazione che consente l'invio di eventi a un bucket Amazon S3 specifico. CloudTrail in genere fornisce i file di registro entro 15 minuti dall'attività dell'account.

> **Note**
>
> Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi.

Puoi creare due tipi di percorsi per un AWS account: un percorso che si applica a tutte le regioni o un percorso che si applica a una regione. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni .

Inoltre, è possibile configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail registri. Per ulteriori informazioni, consultare:

- Panoramica della creazione di un percorso
- CloudTrail servizi e integrazioni supportati
- Configurazione delle SNS notifiche Amazon per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Amazon Aurora Voci dei file di RDS log di Aurora

CloudTrail i file di registro contengono una o più voci di registro. CloudTrail i file di registro non sono una traccia ordinata delle API chiamate pubbliche, quindi non appaiono in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`CreateDBInstance`azione.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2018-07-30T22:14:06Z",
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
            "audit",
            "error",
```

```
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
        "readReplicaDBInstanceIdentifiers": [],
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetStatus": "Active"
                },
```

```
                {
                    "subnetAvailabilityZone": {"name": "us-east-1e"},
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1f"},
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1c"},
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1d"},
                    "subnetIdentifier": "subnet-42599426",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1a"},
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetStatus": "Active"
                }
            ],
            "vpcId": "vpc-136a4c6a",
            "subnetGroupStatus": "Complete"
        },
        "masterUsername": "myawsuser",
        "multiAZ": false,
        "autoMinorVersionUpgrade": true,
        "engine": "mysql",
        "cACertificateIdentifier": "rds-ca-2015",
        "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
        "dBSecurityGroups": [],
        "pendingModifiedValues": {
            "masterUserPassword": "****",
            "pendingCloudwatchLogsExports": {
                "logTypesToEnable": [
                    "audit",
                    "error",
                    "general",
                    "slowquery"
```

```
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Come illustrato nell'elemento `userIdentity` nell'esempio precedente, ogni voce di evento o di registro contiene informazioni su chi ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni su`userIdentity`, vedete l'CloudTrail userIdentity elemento. Per ulteriori informazioni `CreateDBInstance` e altre RDS azioni Amazon Aurora, consulta Reference.

Monitoraggio di Amazon Aurora tramite i flussi di attività del database

Con Flussi di attività del database puoi monitorare pressoché in tempo reale i flussi di attività del database.

Argomenti

- Panoramica dei flussi di attività di database
- Prerequisiti di rete per Database Activity Streams Aurora MySQL
- Avvio di un flusso di attività di database
- Recupero dello stato di un flusso di attività del database
- Arresto di un flusso di attività di database
- Monitoraggio di flussi di attività di database
- IAMesempi di policy per i flussi di attività del database

Panoramica dei flussi di attività di database

Come amministratore del database Amazon Aurora, devi proteggere il database e soddisfare i requisiti normativi e di conformità. Una strategia consiste nell'integrare i flussi di attività del database con gli strumenti di monitoraggio.

Le minacce alla sicurezza sono sia esterne che interne. Per proteggersi dalle minacce interne, è possibile controllare l'accesso degli amministratori ai flussi di dati configurando la funzionalità flussi di attività del database. DBAs non ha accesso alla raccolta, alla trasmissione, all'archiviazione e all'elaborazione degli stream.

Indice

- Come funzionano i flussi di attività del database
- Modalità asincrona e sincrona per flussi di attività di database
- Requisiti e limitazioni per flussi di attività del database
- Disponibilità di regioni e versioni
- Classi di istanza database supportate per i flussi di attività di database

Come funzionano i flussi di attività del database

In Amazon Aurora, puoi avviare un flusso di attività del database a livello di cluster. Tutte le istanze database all'interno del cluster hanno i flussi di attività del database abilitati.

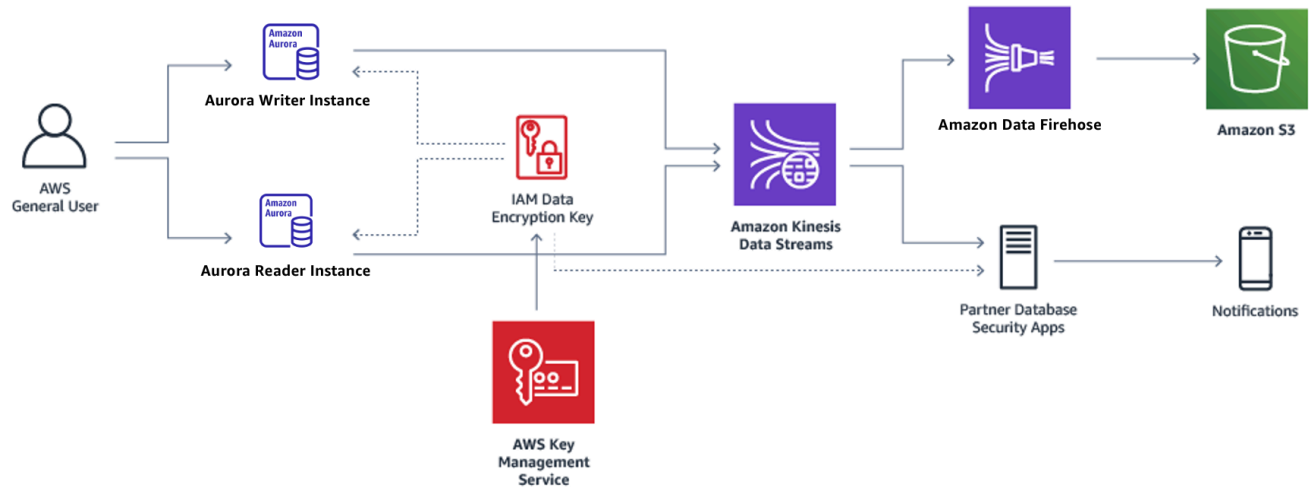
Il cluster di database Aurora inserisce le attività in un flusso di dati Amazon Kinesis pressoché in tempo reale. Il flusso Kinesis viene creato automaticamente. Da Kinesis, puoi configurare AWS servizi come Amazon Data Firehose e consumare lo stream e AWS Lambda archiviare i dati.

> **Important**
>
> L'utilizzo della funzionalità Flussi di attività del database in Amazon Aurora è gratuito, ma Amazon Kinesis addebita i costi del flusso di dati. Per ulteriori informazioni, consulta Prezzi di Amazon Kinesis Data Streams.

Se utilizzi un database globale Aurora, avvia un flusso di attività del database su ciascun cluster di database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS. I flussi di attività non funzionano in modo diverso durante un failover. Continuano a verificare il database globale come al solito.

Puoi configurare le applicazioni per la gestione della conformità affinché attingano dai flussi di attività del database. Per Aurora PostgreSQL, le applicazioni di conformità includono Security Guardium di IBM e Database Audit and Protection di Imperva. SecureSphere Tali applicazioni possono utilizzare il flusso per generare avvisi e attività di verifica per il cluster di database Aurora.

L'immagine seguente mostra un cluster Aurora DB configurato con Amazon Data Firehose.



Modalità asincrona e sincrona per flussi di attività di database

Puoi decidere di impostare la sessione del database per gestire gli eventi dell'attività di database in una delle seguenti modalità:

- Modalità asincrona: quando una sessione di database genera un evento del flusso di attività, vengono ripristinate immediatamente le normali attività della sessione. In background, l'evento di flusso di attività è reso un record durevole. Se si verifica un errore nell'attività in background, viene inviato une evento RDS. Questo indica l'inizio e la fine di qualsiasi finestra temporale in cui i record dell'evento del flusso di attività potrebbero essere stati persi.

  La modalità asincrona favorisce le prestazioni del database rispetto alla precisione del flusso di attività.

  > **Note**
  >
  > La modalità asincrona è disponibile per entrambi Aurora PostgreSQL e Aurora MySQL.

- Modalità sincrona: quando una sessione di database genera un evento del flusso di attività, la sessione si blocca finché l'evento non viene reso durevole. Se per qualche motivo non è possibile rendere l'evento durevole, vengono ripristinate le normali attività della sessione di database. Tuttavia, viene inviato un evento RDS che indica che i record del flusso di attività possono essere persi per qualche secondo. Un secondo evento RDS viene inviato dopo che viene ripristinato lo stato di integrità del sistema.

La modalità sincrona favorisce la precisione del flusso di attività sulle prestazioni del database.

> **Note**
>
> La modalità sincrona è disponibile per Aurora PostgreSQL. Non è possibile utilizzare la modalità sincrona con Aurora MySQL.

Requisiti e limitazioni per flussi di attività del database

In Aurora, i flussi di attività del database hanno i requisiti e le limitazioni riportati di seguito:

- I flussi di attività del database richiedono l'utilizzo di Amazon Kinesis.
- AWS Key Management Service (AWS KMS) è necessario per i flussi di attività del database perché sono sempre crittografati.
- L'applicazione di una crittografia aggiuntiva al flusso di dati di Amazon Kinesis è incompatibile con i flussi di attività del database, che sono già crittografati con la tua chiave. AWS KMS
- Avvia il flusso di attività del database a livello di cluster di database. Se aggiungi un'istanza database al cluster, non è necessario avviare un flusso di attività sull'istanza: questa viene controllata automaticamente.
- In un database globale Aurora, avvia un flusso di attività del database su ciascun cluster di database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS.
- In Aurora PostgreSQL, assicurati di interrompere il flusso di attività del database prima di un aggiornamento della versione principale. È possibile avviare il flusso di attività del database al termine dell'aggiornamento.

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e Regioni con Aurora e i flussi di attività del database, consulta Regioni supportate e motori Aurora DB per i flussi di attività del database.

Classi di istanza database supportate per i flussi di attività di database

Per Aurora MySQL, è possibile utilizzare flussi di attività di database con le seguenti classi di istanza database:

- db.r8g. *grande
- db.r7g.*large
- db.r7i. *grande
- db.r6g.*large
- db.r6i.*large
- db.r5.*large
- db.x2g.*

Per Aurora PostgreSQL, è possibile utilizzare flussi di attività di database con le seguenti classi di istanza database:

- db.r8g. *grande
- db.r7i. *grande
- db.r7g.*large
- db.r6g.*large
- db.r6i.*large
- db.r6id.*large
- db.r5.*large
- db.r4.*large
- db.x2g.*

Prerequisiti di rete per Database Activity Streams Aurora MySQL

Nella sezione seguente, puoi scoprire come configurare il cloud privato virtuale (VPC) per l'utilizzo con flussi di attività di database.

> **Note**
>
> I prerequisiti della rete MySQL di Aurora sono applicabili alle seguenti versioni del motore:

- Aurora MySQL versione 2, fino alla 2.11.3
- Aurora MySQL versione 2.12.0
- Aurora MySQL versione 3, fino alla 3.04.2

Argomenti

- AWS KMS Prerequisiti per gli endpoint
- Prerequisiti per la disponibilità pubblica
- Prerequisiti per la disponibilità privata

AWS KMS Prerequisiti per gli endpoint

Le istanze in un cluster Aurora MySQL che utilizzano flussi di attività devono essere in grado di accedere agli endpoint. AWS KMS Assicurarsi che questo requisito sia soddisfatto prima di abilitare i flussi di attività del database per il cluster Aurora MySQL. Se il cluster Aurora è disponibile al pubblico, questo requisito viene soddisfatto automaticamente.

> **Important**
>
> Se il cluster Aurora MySQL DB non riesce ad accedere all' AWS KMS endpoint, il flusso di attività si interrompe. In tal caso, Aurora notifica questo problema utilizzando eventi RDS.

Prerequisiti per la disponibilità pubblica

Affinché un cluster di database Aurora sia pubblico, deve soddisfare i seguenti requisiti:

- L'opzione Accessibile pubblicamente è Sì nella pagina dei dettagli del cluster AWS Management Console .
- Inoltre, il cluster di database è in una sottorete pubblica di Amazon VPC. Per ulteriori informazioni sulle istanze database accessibili pubblicamente, consulta Uso di un cluster database in un VPC. Per ulteriori informazioni sulle sottoreti pubbliche Amazon VPC, consulta VPC e sottoreti.

Prerequisiti per la disponibilità privata

Se il cluster database Aurora si trova in una sottorete pubblica VPC e non è accessibile pubblicamente, significa che è privato. Per mantenere privato il cluster e utilizzarlo con flussi di attività del database, sono disponibili le opzioni seguenti:

- Configurare Network Address Translation (NAT) nel VPC. Per ulteriori informazioni, consulta Gateway NAT.
- Crea un AWS KMS endpoint nel tuo VPC. Questa opzione è consigliata perché è più semplice da configurare.

Per creare un AWS KMS endpoint nel tuo VPC

1. Accedere alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione, seleziona Endpoints (Endpoint).
3. Scegliere Create Endpoint (Crea endpoint).

   La pagina Creazione endpoint viene visualizzata.
4. Esegui questa operazione:
   - In Service category (Categoria dei servizi), scegliere AWS services (Servizi AWS).
   - In Service Name, scegli com.amazonaws. ***region*** .kms, dove la *regione* è il luogo in cui si trova il Regione AWS cluster.
   - Per VPC, scegliere il VPC in cui si trova il cluster.
5. Scegliere Create Endpoint (Crea endpoint).

Per ulteriori informazioni sulla configurazione degli endpoint VPC, consulta Endpoint VPC.

Avvio di un flusso di attività di database

Per monitorare l'attività del database per tutte le istanze del cluster database Aurora, avvia un flusso di attività Aurora livello di cluster. Le eventuali istanze database aggiunte al cluster vengono anche monitorate automaticamente. Se utilizzi un database globale Aurora, avvia un flusso di attività del database su ciascun cluster di database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS.

Quando avvii un flusso di attività, ogni evento di attività del database configurato nella policy di audit, genera un evento di flusso di attività. Gli eventi di accesso vengono generati da comandi SQL quali `CONNECT` e `SELECT`. Gli eventi di modifica vengono generati da comandi SQL quali `CREATE` e `INSERT`.

Console

Come avviare un flusso di attività di database

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database per cui desideri abilitare un flusso di attività.
4. In Actions (Operazioni), scegliere Start activity stream (Avvia flusso di attività).

   Viene visualizzata la *name* finestra Avvia flusso di attività del database:, dove si *name* trova il cluster DB dell'.
5. Specificare le seguenti impostazioni:

   - Per AWS KMS key, scegli una chiave dall'elenco di AWS KMS keys.

     > **Note**
     >
     > Se il cluster Aurora MySQL non è in grado di accedere alle chiavi KMS, seguire le istruzioni riportate in Prerequisiti di rete per Database Activity Streams Aurora MySQL per attivare tale accesso.

     Aurora utilizza la chiave KMS per crittografare la chiave che a sua volta esegue la crittografia dell'attività del database. Scegliere una chiave KMS diversa dalla chiave di default. Per ulteriori informazioni sulle chiavi di crittografia e AWS KMS, vedi Cos'è AWS Key Management Service? nella Guida per gli AWS Key Management Service sviluppatori.
   - In Database activity stream mode (Modalità flusso di attività di database), scegliere Asynchronous (Asincrona) o Synchronous (Sincrona).

> **Note**
>
> Questa scelta si applica solo a Aurora PostgreSQL. Per Aurora MySQL, è possibile utilizzare solo la modalità asincrona.

- Scegliere Immediatamente.

  Selezionando Subito, il cluster database viene riavviata immediatamente. Se si sceglie Durante la finestra di manutenzione successiva, il cluster database non si riavvia subito. In questo caso, il flusso di attività del database non viene avviato fino alla finestra di manutenzione successiva.

6. Scegli Start database activity stream (Avvia flusso di attività di database).

   Lo stato del cluster di database mostra che il flusso di attività è in fase di avvio.

> **Note**
>
> Se ricevi l'errore `You can't start a database activity stream in this configuration`, controlla Classi di istanza database supportate per i flussi di attività di database per vedere se il cluster database utilizza una classe di istanza supportata.

AWS CLI

Per avviare i flussi di attività del database per un cluster di database (), configura il cluster di database () utilizzando il start-activity-stream AWS CLI comando.

- `--resource-arn` *`arn`*: specifica l'Amazon Resource Name (ARN) del cluster database.
- `--mode` *`sync-or-async`*: specifica la modalità sincrona (`sync`) o asincrona (`async`). Per Aurora PostgreSQL, è possibile scegliere uno dei due valori. Per Aurora MySQL, specifica `async`.
- `--kms-key-id` *`key`*: specifica l'identificatore della chiave KMS per la crittografia dei messaggi nel flusso di attività del database. L'identificatore della chiave AWS KMS è l'ARN della chiave, l'ID chiave, l'alias ARN o il nome alias per. AWS KMS key

L'esempio seguente avvia un flusso di attività del database per un cluster di database in modalità asincrona.

LinuxPermacOS, o: Unix

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-cluster-arn \
    --apply-immediately
```

Per Windows:

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-cluster-arn ^
    --apply-immediately
```

Amazon RDS API

Per avviare i flussi di attività del database per un', configura l' cluster utilizzando l'StartActivityStreamoperazione.

Richiamare l'operazione con i parametri seguenti:

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`

> **Note**
>
> Se viene visualizzato un errore che indica che non è possibile avviare un flusso di attività del database con la versione corrente del database Aurora PostgreSQL, applica la patch più recente per Aurora PostgreSQL prima di iniziare un flusso di attività del database. Per informazioni sull'aggiornamento del database Aurora PostgreSQL, consulta. Aggiornamento dei cluster Amazon Aurora DB

Di seguito sono riportate le versioni minime delle patch per avviare i flussi di attività del database con Aurora PostgreSQL.

- 3.4.15 (11.9.15), 11.21.10
- 12,9,15, 12,15,9, 12,16,10, 12,17,7, 12,18,5, 12,19,4, 12,20,3, 12,22,3
- 13,9,12, 13,11,9, 13,12,10, 13,13,7, 13,14,5, 13,15,4, 13,16,3, 13,18,3
- 14.6.12, 14.8.9, 14.9.10, 14.10.7, 14.11.5, 14.12.4, 14.13.3, 14.15.3
- 15.3.9, 15.4.10, 15.5.7, 15.6.5, 15.7.4, 15.8.3, 15.10.3
- 16.1.7, 16.2.5, 16.3.4, 16.4.3, 16.6.3

Recupero dello stato di un flusso di attività del database

Puoi recuperare lo stato di un flusso di attività tramite la console o la AWS CLI.

Console

Come recuperare lo stato di un flusso di attività del database

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Database e selezionare il link del cluster database.
3. Scegliere la scheda Configurazione e selezionare Flusso di attività di database per lo stato.

AWS CLI

È possibile ottenere la configurazione del flusso di attività per un cluster di database come risposta a una richiesta della CLI describe-db-clusters .

Nell'esempio seguente viene illustrato *my-cluster*.

```
aws rds --region my-region describe-db-clusters --db-cluster-identifier my-cluster
```

Il seguente esempio mostra una risposta in formato JSON: Sono visualizzati i seguenti campi:

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`
- `ActivityStreamStatus`

- `ActivityStreamMode`
-

Questi campi sono gli stessi per Aurora PostgreSQL e Aurora MySQL, tranne che `ActivityStreamMode` è sempre `async` per Aurora MySQL, mentre per Aurora PostgreSQL esso potrebbe essere `sync` o `async`.

```
{
    "DBClusters": [
        {
      "DBClusterIdentifier": "my-cluster",
            ...
            "ActivityStreamKinesisStreamName": "aws-rds-das-cluster-
A6TSYXITZCZXJHIRVFUBZ5LTWY",
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "12345678-abcd-efgh-ijkl-bd041f170262",
            "ActivityStreamMode": "async",
            "DbClusterResourceId": "cluster-ABCD123456"
            ...
        }
    ]
}
```

API RDS

È possibile ottenere la configurazione del flusso di attività per un cluster di database come risposta a un'operazione DescribeDBClusters .

Arresto di un flusso di attività di database

Puoi interrompere un flusso di attività utilizzando la console o AWS CLI.

Se elimini il cluster di database, il flusso di attività viene arrestato e il flusso Amazon Kinesis sottostante viene eliminato automaticamente.

Console

Per disattivare un flusso di attività

1. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere un cluster database per il quale si desidera interrompere il flusso di attività di database.
4. In Actions (Operazioni), scegliere Stop activity stream (Interrompi flusso di attività). Viene visualizzata la finestra Database Activity Stream (Flusso di attività di database).

    a. Scegliere Immediatamente.

    Selezionando Subito, il cluster database viene riavviata immediatamente. Se si sceglie Durante la finestra di manutenzione successiva, il cluster database non si riavvia subito. In questo caso, il flusso di attività del database non viene arrestato fino alla finestra di manutenzione successiva.

    b. Scegli Continue (Continua).

AWS CLI

Per interrompere i flussi di attività del database per il cluster DB del , configura l' utilizzando il AWS CLI comando stop-activity-stream. Identifica la regione AWS per il cluster database mediante il parametro `--region`. Il parametro `--apply-immediately` è facoltativo.

Per LinuxmacOS, oUnix:

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_CLUSTER_ARN \
    --apply-immediately
```

Per Windows:

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_CLUSTER_ARN ^
    --apply-immediately
```

API RDS

Per interrompere i flussi di attività del database per il cluster di database (), configura l' del cluster utilizzando l'StopActivityStreamoperazione. Identifica la regione AWS per il cluster database mediante il parametro `Region`. Il parametro `ApplyImmediately` è facoltativo.

Monitoraggio di flussi di attività di database

I flussi di attività di database monitorano e segnalano le attività. Il flusso di attività viene raccolto e trasmesso a Amazon Kinesis. Da Kinesis, è possibile monitorare il flusso di attività oppure altri servizi e applicazioni possono utilizzare il flusso di attività per ulteriori analisi. Puoi trovare il nome del flusso Kinesis sottostante utilizzando il AWS CLI comando `describe-db-clusters`o l'RDSAPI`DescribeDBClusters`operazione.

Aurora gestisce lo stream Kinesis per te nel modo seguente:

- Aurora crea automaticamente lo stream Kinesis con un periodo di conservazione di 24 ore.
- Aurora ridimensiona lo stream Kinesis, se necessario.
- Se interrompi il flusso di attività del database o elimini il cluster DB (), Aurora elimina il flusso Kinesis.

Le seguenti categorie di attività vengono monitorate e inserite nel log di controllo del flusso di attività:

- SQLcomandi: tutti i SQL comandi vengono controllati e anche le istruzioni preparate, le funzioni integrate e le funzioni in PL/. SQL Le chiamate alle procedure archiviate vengono controllate. Vengono inoltre controllate tutte SQL le dichiarazioni emesse all'interno di procedure o funzioni memorizzate.
- Altre informazioni sul database: l'attività monitorata include l'SQListruzione completa, il conteggio delle righe interessate dai DML comandi, gli oggetti a cui si accede e il nome univoco del database. Per Aurora PostgreSQL, i flussi di attività del database monitorano anche le variabili di associazione e i parametri delle stored procedure.

  > **Important**
  >
  > Il SQL testo completo di ogni dichiarazione è visibile nel registro di controllo del flusso di attività, inclusi tutti i dati sensibili. Tuttavia, le password degli utenti del database vengono oscurate se Aurora è in grado di determinarle dal contesto, come nella seguente dichiarazione. SQL
  >
  > ```
  > ALTER ROLE role-name WITH password
  > ```

- Informazioni di connessione – L'attività monitorata include informazioni di sessione e di rete, l'ID di processo del server e i codici di uscita.

Se un flusso di attività presenta un errore durante il monitoraggio dell'istanza DB, l'utente viene avvisato tramite eventi. RDS

Nelle sezioni seguenti, puoi accedere, controllare ed elaborare i flussi di attività del database.

Argomenti

- Accesso a un flusso di attività da Amazon Kinesis
- Contenuti ed esempi dei log di controllo per i flussi di attività del database
- databaseActivityEventElenca l'array JSON per i flussi di attività del database
- Elaborazione di un flusso di attività del database utilizzando AWS SDK

Accesso a un flusso di attività da Amazon Kinesis

Quando abiliti un flusso di attività per un cluster database, viene creato automaticamente un flusso Kinesis. Da Kinesis, puoi monitorare l'attività del database in tempo reale. Per analizzare ulteriormente l'attività del database, puoi connettere il flusso Kinesis ad applicazioni consumer. Puoi anche connettere lo stream ad applicazioni di gestione della conformità come Security Guardium o Security Guardium IBM di Imperva SecureSphere Database Audit and Protection IBM .

Puoi accedere allo stream Kinesis dalla RDS console o dalla console Kinesis.

Per accedere a un flusso di attività da Kinesis utilizzando la console RDS

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli l' cluster DB su cui hai avviato un flusso di attività.
4. Scegliere Configuration (Configurazione).
5. In Database activity stream (Flusso di attività del database), scegli il collegamento sotto Kinesis stream (Flusso Kinesis).
6. Nella console Kinesis, scegli Monitoring (Monitoraggio) per iniziare l'osservazione dell'attività del database.

Per accedere a un flusso di attività da Kinesis utilizzando la console Kinesis

1. Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com/kinesis.
2. Scegliere il flusso di attività dall'elenco di flussi Kinesis.

Il nome di un flusso di attività include il prefisso `aws-rds-das-cluster-` seguito dall'ID risorsa del cluster database. Di seguito è riportato un esempio.

```
aws-rds-das-cluster-NHVOV4PCLWHGF52NP
```

Per utilizzare la RDS console Amazon per trovare l'ID della risorsa per il cluster di , scegli l' dall'elenco dei database, quindi scegli la scheda Configurazione.

Per utilizzare AWS CLI per trovare il nome completo dello stream Kinesis per un flusso di attività, usa una describe-db-clustersCLIrichiesta e annota il valore di `ActivityStreamKinesisStreamName` nella risposta.

3. Scegliere Monitoring (Monitoraggio) per iniziare l'osservazione dell'attività di database.

Per ulteriori informazioni sull'utilizzo di Amazon Kinesis, consulta Che cos'è Amazon Kinesis Data Streams?.

Contenuti ed esempi dei log di controllo per i flussi di attività del database

Gli eventi monitorati sono rappresentati nel flusso di attività del database come stringhe JSON. La struttura è costituita da un oggetto JSON contenente un `DatabaseActivityMonitoringRecord`, che a sua volta contiene un array `databaseActivityEventList` di eventi attività.

> **Note**
>
> Per i flussi di attività del database, l'array `paramList` JSON non include valori nulli delle applicazioni Hibernate.

Argomenti

- Esempi di log di verifica per un flusso di attività
- DatabaseActivityMonitoringRecords oggetto JSON
- databaseActivityEvents Oggetto JSON

Esempi di log di verifica per un flusso di attività

Di seguito sono riportati registri di controllo JSON decrittografati di esempio di record di eventi attività.

Example Record di evento di attività di un'istruzione Aurora PostgreSQL CONNECT SQL

Il seguente record di evento di attività mostra un accesso con l'utilizzo di un'istruzione SQL `CONNECT` (`command`) mediante un client psql (`clientApplication`).


```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-10-30 00:39:49.940668+00",
          "logTime": "2019-10-30 00:39:49.990579+00",
          "statementId": 1,
          "substatementId": 1,
          "objectType": null,
          "command": "CONNECT",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "49804",
          "sessionId": "5ce5f7f0.474b",
          "rowCount": null,
          "commandText": null,
          "paramList": [],
          "pid": 18251,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "MISC",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
```

```
    },
   "key":"decryption-key"
}
```

Example Record di evento attività di un'istruzione SQL CONNECT Aurora MySQL

Il seguente record di evento di attività mostra un accesso con l'utilizzo di un'istruzione SQL `CONNECT` (`command`) mediante un client mysql (`clientApplication`).

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:13.267214+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"rdsadmin",
      "databaseName":"",
      "remoteHost":"localhost",
      "remotePort":"11053",
      "command":"CONNECT",
      "commandText":"",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"",
      "statementId":0,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725121",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:13.267207+00",
      "endTime":"2020-05-22 18:07:13.267213+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
```

```
      "class":"MAIN"
    }
  ]
}
```

Example Record di evento attività di Aurora PostgreSQL

Il seguente esempio mostra un evento `CREATE TABLE` per Aurora PostgreSQL.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:36:54.403455+00",
          "logTime": "2019-05-24 00:36:54.494235+00",
          "statementId": 2,
          "substatementId": 1,
          "objectType": null,
          "command": "CREATE TABLE",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": null,
          "commandText": "create table my_table (id serial primary key, name
 varchar(32));",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "DDL",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
```

```
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Record di evento attività di un'istruzione CREATE TABLE Aurora MySQL

Il seguente esempio mostra un'istruzione `CREATE TABLE` per Aurora MySQL. L'operazione è rappresentata come due record di eventi separati. Un evento ha `"class":"MAIN"`. L'altro evento ha `"class":"AUX"`. I messaggi potrebbero arrivare in qualsiasi ordine. Il campo `logTime` dell'evento `MAIN` è sempre precedente ai campi `logTime` di qualsiasi evento AUX corrispondente.

Nell'esempio seguente viene illustrato l'evento con un valore `class` pari a `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.250221+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"CREATE TABLE test1 (id INT)",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
```

```
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.250222+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

Nell'esempio seguente viene illustrato l'evento corrispondente con un valore `class` pari a AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.247182+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"CREATE",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
```

```
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.247182+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Example Record di evento attività di un'istruzione Aurora PostgreSQL SELECT

Il seguente esempio mostra un evento SELECT .

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:39:49.920564+00",
          "logTime": "2019-05-24 00:39:49.940668+00",
          "statementId": 6,
          "substatementId": 1,
          "objectType": "TABLE",
          "command": "SELECT",
          "objectName": "public.my_table",
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": 10,
          "commandText": "select * from my_table;",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
```

```
          "class": "READ",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
```

```
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x010500000000000515000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```


Example Record di evento attività di un'istruzione SELECT Aurora MySQL

Il seguente esempio mostra un evento `SELECT`.

Nell'esempio seguente viene illustrato l'evento con un valore `class` pari a `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986467+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"SELECT * FROM test1 WHERE id < 28",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"726571",
      "rowCount":2,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986467+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

Nell'esempio seguente viene illustrato l'evento corrispondente con un valore `class` pari a AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
```

```
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986399+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"READ",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"726571",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986399+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

DatabaseActivityMonitoringRecords oggetto JSON

I record di eventi dell'attività del database si trovano in un oggetto JSON che contiene le seguenti informazioni.

Campo JSON	Tipo di dati	Descrizione
`type`	stringa	Il tipo di record JSON. Il valore è `DatabaseActivityMonitoringRecords` .
`version`	stringa	La versione dei record di monitoraggio delle attività del database. La versione dei record di attività del database generati dipende dalla versione del motore del cluster database. • I record di attività del database versione 1.1 vengono generati per i cluster di database Aurora PostgreSQL che eseguono le versioni del motore 10.10 e versioni secondarie successive e versioni del motore 11.5 e successive. • I record di attività del database versione 1.0 vengono generati per i cluster di database Aurora PostgreSQL che eseguono le versioni del motore 10.7 e 11.4. Tutti i seguenti campi sono nella versione 1.0 e nella versione 1.1, tranne dove espressamente indicato.
databaseActivityEvents	string	Un oggetto JSON contenente gli eventi di attività.
key	string	Una chiave di crittografia utilizzata per decrittare databaseActivityEventElenca l'array JSON

databaseActivityEvents Oggetto JSON

L'oggetto JSON `databaseActivityEvents` contiene le seguenti informazioni.

Campi di primo livello nel record JSON

Ogni evento nel registro di controllo viene racchiuso in un record in formato JSON. Questo record contiene i seguenti campi.

type

Questo campo ha sempre il valore `DatabaseActivityMonitoringRecords`.

versione

Questo campo rappresenta la versione del protocollo o del contratto di dati del flusso di attività del database. Definisce quali campi sono disponibili.

La versione 1.0 rappresenta il supporto dei flussi di attività dati originali per le versioni Aurora PostgreSQL 10.7 e 11.4. La versione 1.1 rappresenta il supporto dei flussi di attività dati per le versioni Aurora PostgreSQL 10.10 e successive e Aurora PostgreSQL 11.5 e successive. La versione 1.1 include i campi aggiuntivi `errorMessage` e `startTime`. La versione 1.2 rappresenta il supporto dei flussi di attività dati per Aurora MySQL 2.08 e versioni successive. La versione 1.2 include i campi aggiuntivi `endTime` e `transactionId`.

databaseActivityEvents

Stringa crittografata che rappresenta uno o più eventi di attività. È rappresentato come un array di byte base64. Quando si decrittografa la stringa, il risultato è un record in formato JSON con campi come illustrato negli esempi di questa sezione.

key

Chiave dati crittografata utilizzata per crittografare la stringa `databaseActivityEvents`. È lo stesso AWS KMS key che hai fornito quando hai avviato il flusso di attività del database.

Nell'esempio seguente viene illustrato il formato di questo record.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
```

```
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

Per decrittografare il contenuto del campo `databaseActivityEvents`, procedere come segue:

1. Decrittare il valore nel campo `key` JSON utilizzando la chiave KMS fornita all'avvio del flusso di attività del database. In questo modo viene restituita la chiave di crittografia dei dati in testo non crittografato.
2. Decodificare il valore nel campo `databaseActivityEvents` JSON con base64 per ottenere il testo cifrato, in formato binario, del payload di controllo.
3. Decifrare il testo cifrato binario con la chiave di crittografia dei dati decodificata nel primo passaggio.
4. Decomprimere il payload decrittografato.
   - Il payload crittografato è nel campo `databaseActivityEvents`.
   - Il campo `databaseActivityEventList` contiene una matrice di record di controllo. I campi `type` nella matrice possono essere `record` o `heartbeat`.

Il record dell'evento attività registro di controllo è un oggetto JSON che contiene le seguenti informazioni.

Campo JSON	Tipo di dati	Descrizione
`type`	stringa	Il tipo di record JSON. Il valore è `DatabaseActivityMonitoringRecord` .
`clusterId`	stringa	Identificatore di risorsa cluster di database. Corrisponde all'attributo cluster database `DbClusterResourceId` .
`instanceId`	stringa	Identificatore della risorsa istanza database. Corrisponde all'attributo di istanza database `DbiResourceId` .
databaseActivityEventElenca l'array JSON	stringa	Matrice di record di controllo delle attività o messaggi heartbeat.

databaseActivityEventElenca l'array JSON per i flussi di attività del database

Il payload del registro di controllo è un array JSON `databaseActivityEventList` crittografato. Le tabelle riportano in ordine alfabetico i campi per ogni evento di attività nella matrice `DatabaseActivityEventList` decrittata di un log di verifica. I campi sono diversi a seconda che si utilizzi Aurora PostgreSQL o Aurora MySQL. Consultare la tabella che si applica al modulo di gestione di database.

> **Important**
>
> Tale struttura di eventi è soggetta a modifiche. Aurora potrebbe aggiungere nuovi campi agli eventi di attività in futuro. Nelle applicazioni che analizzano i dati JSON, assicurarsi che il codice possa ignorare o eseguire le azioni appropriate per i nomi di campo sconosciuti.

databaseActivityEventElenca i campi per Aurora PostgreSQL

Di seguito sono riportati `databaseActivityEventList` i campi per Aurora PostgreSQL.

Campo	Tipo di dati	Descrizione
`class`	stringa	La classe dell'evento attività. I valori validi per Aurora PostgreSQL sono i seguenti. • `ALL` • `CONNECT` – Un evento di connessione o disconnessione. • `DDL` – Un'istruzione DDL che non è inclusa nell'elenco di istruzioni per la classe `ROLE`. • `FUNCTION` – Una chiamata di funzione o un blocco `DO`. • `MISC` – Un comando vario quale `DISCARD`, `FETCH`, `CHECKPOINT` o `VACUUM`. • `NONE` • `READ` – Un'istruzione `SELECT` o `COPY` quando l'origine è una relazione o una query. • `ROLE` – Un'istruzione correlata ai ruoli e privilegi inclusi `GRANT`, `REVOKE` e `CREATE/ALTER/DROP ROLE`.

Campo	Tipo di dati	Descrizione
		• `WRITE` – Un'istruzione `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` o `COPY` quando la destinazione è una relazione.
`clientApplication`	stringa	L'applicazione utilizzata dal client per eseguire la connessione come segnalato dal client. Il client non deve fornire queste informazioni, pertanto il valore può essere nullo.
`command`	stringa	Il nome del comando SQL senza alcun dettaglio comando.
`commandText`	stringa	L'istruzione SQL effettiva passata dall'utente. Per Aurora PostgreSQL, il valore è identico all'istruzione SQL originale. Questo campo viene utilizzato per tutti i tipi di record tranne che i record di connessione e disconnessione, nel quale caso il valore è nullo. **Important** Il testo SQL completo di ogni istruzione è visibile nel registro di controllo del flusso di attività, inclusi eventuali dati sensibili. Tuttavia, le password degli utenti del database vengono omesse se Aurora può stabilirle dal contesto, come nell'istruzione SQL seguente. `ALTER ROLE role-name WITH password`
`databaseName`	stringa	Il database a cui è connesso l'utente.
`dbProtocol`	stringa	Il protocollo del database, ad esempio `Postgres 3.0`.
`dbUserName`	stringa	L'utente del database con cui il client è autenticato.

Campo	Tipo di dati	Descrizione
`errorMessage` (solo record di attività del database versione 1.1)	stringa	Se si verifica una errore qualsiasi, questo campo viene compilato con il messaggio di errore che verrebbe generato dal server di database. Il valore `errorMessage` è nullo per le istruzioni normali che non hanno generato un errore. Un errore è definito come qualsiasi attività che produce un evento del log degli errori PostgreSQL visibile al client in corrispondenza di un livello di gravità pari a `ERROR` o superiore. Per ulteriori informazioni, consulta la tabella dei livelli di gravità dei messaggi di PostgreSQL. Ad esempio, errori di sintassi e annullamenti delle query generano un messaggio di errore. Gli errori del server PostgreSQL interno, come gli errori del processo checkpointer in background non generano un messaggio di errore. Tuttavia, i record per tali eventi vengono comunque generati a prescindere dall'impostazione del livello di gravità del registro. Ciò impedisce agli aggressori di disattivare la registrazione per tentare di evitare il rilevamen to. Vedere anche il campo `exitCode`.
`exitCode`	int	Un valore usato per un record di chiusura sessione. In una clean exit, contiene il codice di chiusura. Un codice di chiusura può sempre essere ottenuto in alcuni scenari di errore. Esempi sono se PostgreSQL esegue un `exit()` o se un operatore esegue un comando quale `kill -9`. Se si verifica un errore qualsiasi, il campo `exitCode` mostra il codice di errore SQL, `SQLSTATE`, come elencato in PostgreSQL Error Codes. Vedere anche il campo `errorMessage` .

Campo	Tipo di dati	Descrizione
`logTime`	stringa	Una timestamp come registrato nel percorso del codice di controllo. Rappresenta l'ora di fine dell'esecuzione dell'istruzione SQL. Vedere anche il campo `startTime` .
`netProtocol`	stringa	Il protocollo di comunicazione di rete.
`objectName`	stringa	Il nome dell'oggetto di database se l'istruzione SQL viene eseguita su uno. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo.
`objectType`	stringa	Il tipo di oggetto di database, ad esempio tabella, indice, vista e così via. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo. I valori validi includono i seguenti: • `COMPOSITE TYPE` • `FOREIGN TABLE` • `FUNCTION` • `INDEX` • `MATERIALIZED VIEW` • `SEQUENCE` • `TABLE` • `TOAST TABLE` • `VIEW` • `UNKNOWN`
`paramList`	stringa	Un array di parametri separati da virgole passati all'istruzione SQL. Se l'istruzione SQL non dispone di parametri, questo valore è un array vuoto.

Campo	Tipo di dati	Descrizione
`pid`	int	L'ID di processo del processo back-end allocato per servire la connessione client.
`remoteHost`	stringa	L'indirizzo IP del client o il nome host. Per Aurora PostgreSQL, quale viene utilizzato dipende dall'impostazione dei parametri `log_hostname` del database. Il `remoteHost` valore include anche `[local]` e `localhost` che indicano l'attività dell'utente. `rdsadmin`
`remotePort`	string	Il numero di porta del client.
`rowCount`	int	Il numero di righe della tabella influenzate o recuperate dall'istruzione SQL. Questo campo viene utilizzato solo per istruzioni SQL che sono istruzioni DML (Data Manipulation Language). Se l'istruzione SQL non è un'istruzione DML, questo valore è nullo.
`serverHost`	string	L'indirizzo IP host del server di database. Il `serverHost` valore include anche `[local]` e `localhost` che indicano l'attività dell'`rdsadmin`utente.
`serverType`	string	Il tipo di server di database, ad esempio `PostgreSQL` .
`serverVersion`	stringa	La versione del server di database, ad esempio `2.3.1` per Aurora PostgreSQL.
`serviceName`	stringa	Il nome del servizio, ad esempi `Amazon Aurora PostgreSQL-Compatible edition` .
`sessionId`	int	Un identificatore di sessione pseudo-univoco.
`sessionId`	int	Un identificatore di sessione pseudo-univoco.

Campo	Tipo di dati	Descrizione
`startTime` (solo record di attività del database versione 1.1)	stringa	L'ora di inizio dell'esecuzione dell'istruzione SQL. Per calcolare il tempo di esecuzione approssimativo dell'istruzione SQL, utilizzar `logTime - startTime` . Vedere anche il campo `logTime`.
`statementId`	int	Identificatore per l'istruzione SQL del client. Il contatore è a livello di sessione e si incrementa con ogni istruzione SQL immessa dal client.
`substatementId`	int	Identificatore di una subistruzione SQL. Questo valore conteggia le istruzioni secondarie contenute per ogni istruzione SQL identificata dal campo `statementId` .
`type`	stringa	Il tipo di evento, I valori validi sono `record` e `heartbeat` .

databaseActivityEventElenca i campi per Aurora MySQL

Di seguito sono riportati `databaseActivityEventList` i campi per Aurora MySQL.

Campo	Tipo di dati	Descrizione
`class`	stringa	La classe dell'evento attività. I valori validi per Aurora MySQL sono i seguenti: • `MAIN` – L' evento primario che rappresenta un'istruzione SQL. • `AUX` – Un evento supplementare contenente ulteriori dettagli. Ad esempio, un'istruzione che rinomina un oggetto potrebbe avere un evento con classe `AUX` che riflette il nuovo nome. Per trovare gli eventi corrispondenti `MAIN` e `AUX` alla stessa istruzione, verificare la presenza di eventi diversi con gli

Campo	Tipo di dati	Descrizione
		stessi valori per il campo `pid` e per il campo `statement Id` .
`clientApplication`	stringa	L'applicazione utilizzata dal client per eseguire la connessione come segnalato dal client. Il client non deve fornire queste informazioni, pertanto il valore può essere nullo.
`command`	stringa	Categoria generale dell'istruzione SQL. I valori di questo campo dipendono dal valore di `class`. I valori quando `class` è `MAIN` includono quanto segue: • `CONNECT` – Quando una sessione client è connessa. • `QUERY` – Istruzione SQL Accompagnato da uno o più eventi con un valore di `class` pari a `AUX`. • `DISCONNECT` – Quando una sessione client viene disconnessa. • `FAILED_CONNECT` – Quando un client tenta di connettersi ma non è in grado di farlo. • `CHANGEUSER` – Un cambiamento di stato che fa parte del protocollo di rete MySQL, non da una dichiarazione rilasciata. I valori quando `class` è `AUX` includono quanto segue: • `READ` – Un'istruzione `SELECT` o `COPY` quando l'origine è una relazione o una query. • `WRITE` – Un'istruzione `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` o `COPY` quando la destinazione è una relazione. • `DROP` – Eliminazione di un oggetto • `CREATE` – Creazione di un oggetto. • `RENAME` – Ridenominazione di un oggetto • `ALTER` – Per cambiare le proprietà di un oggetto

Campo	Tipo di dati	Descrizione
`commandText`	stringa	Per gli eventi con un valore `class` pari a `MAIN`, questo campo rappresenta l'istruzione SQL effettiva passata dall'utente. Questo campo viene utilizzato per tutti i tipi di record tranne che i record di connessione e disconnessione, nel quale caso il valore è nullo. Per gli eventi con un valore `class` pari a `AUX`, questo campo contiene informazioni supplementari sugli oggetti coinvolti nell'evento. Per Aurora MySQL, i caratteri quali le virgolette sono preceduti da una barra rovesciata che rappresenta un carattere di escape. **Important** Il testo SQL completo di ogni istruzione è visibile nel registro di controllo, inclusi eventuali dati sensibili. Tuttavia, le password degli utenti del database vengono omesse se Aurora può stabilirle dal contesto, come nell'istruzione SQL seguente. `mysql> SET PASSWORD = 'my-password ';` **Note** Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.
`databaseName`	string	Il database a cui è connesso l'utente.
`dbProtocol`	stringa	Il protocollo di database. Attualmente, questo valore è sempre `MySQL` per Aurora MySQL.

Campo	Tipo di dati	Descrizione
`dbUserName`	stringa	L'utente del database con cui il client è autenticato.
`endTime` (solo record di attività del database versione 1.2)	stringa	L'ora di fine dell'esecuzione dell'istruzione SQL. È rappresentato in formato UTC (Coordinated Universal Time). Per calcolare il tempo di esecuzione dell'istruzione SQL, utilizzare `endTime - startTime` . Vedere anche il campo `startTime` .
`errorMessage` (solo record di attività del database versione 1.1)	stringa	Se si verifica una errore qualsiasi, questo campo viene compilato con il messaggio di errore che verrebbe generato dal server di database. Il valore `errorMessage` è nullo per le istruzioni normali che non hanno generato un errore. Un errore è definito come qualsiasi attività che produce un evento del log degli errori MySQL visibile al client in corrispondenza di un livello di gravità pari a `ERROR` o superiore. Per ulteriori informazioni, consulta il Log degli errori nel Manuale di riferimento di MySQL. Ad esempio, errori di sintassi e annullamenti delle query generano un messaggio di errore. Gli errori del server MySQL interno, come gli errori del processo checkpointer in background non generano un messaggio di errore. Tuttavia, i record per tali eventi vengono comunque generati a prescindere dall'impostazione del livello di gravità del registro. Ciò impedisce agli aggressori di disattivare la registrazione per tentare di evitare il rilevamento. Vedere anche il campo `exitCode`.
`exitCode`	int	Un valore usato per un record di chiusura sessione. In una clean exit, contiene il codice di chiusura. Un codice di chiusura può sempre essere ottenuto in alcuni scenari di errore. In questi casi, questo valore potrebbe essere zero o potrebbe essere vuoto.

Campo	Tipo di dati	Descrizione
`logTime`	stringa	Una timestamp come registrato nel percorso del codice di controllo. È rappresentato in formato UTC (Coordinated Universal Time). Per il modo più accurato per calcolare la durata dell'istruzione, vedere i campi `startTime` e `endTime`.
`netProtocol`	stringa	Il protocollo di comunicazione di rete. Attualmente, questo valore è sempre `TCP` per Aurora MySQL.
`objectName`	stringa	Il nome dell'oggetto di database se l'istruzione SQL viene eseguita su uno. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo. Per costruire il nome completo dell'oggetto, combinare `databaseName` e `objectName` . Se la query coinvolge più oggetti, questo campo può essere un elenco di nomi separati da virgole.
`objectType`	stringa	Il tipo di oggetto di database, ad esempio tabella, indice e così via. Questo campo viene utilizzato solo dove l'istruzione SQL è eseguita su un oggetto di database. Se l'istruzione SQL non è in esecuzione su un oggetto, questo valore è nullo. I valori validi per Aurora MySQL includono i seguenti: • `INDEX` • `TABLE` • `UNKNOWN`
`paramList`	stringa	Questo campo non viene utilizzato per Aurora MySQL ed è sempre nullo.

Campo	Tipo di dati	Descrizione
`pid`	int	L'ID di processo del processo back-end allocato per servire la connessione client. Quando il server di database viene riavviato, le modifiche `pid` e il contatore per il campo `statementId` ricomincia.
`remoteHost`	stringa	L'indirizzo IP o il nome host del client che ha emesso l'istruzione SQL. Per Aurora MySQL, quale viene utilizzato dipende dall'impostazione dei parametri `skip_name_resolve` del database. Il valore `localhost` indica l'attività dell'utente `rdsadmin` speciale.
`remotePort`	stringa	Il numero di porta del client.
`rowCount`	int	Numero di righe restituite dall'istruzione SQL. Ad esempio, se un'istruzione SELECT restituisce 10 righe, RowCount è 10. Per le istruzioni INSERT o UPDATE, RowCount è 0.
`serverHost`	string	Identificatore dell'istanza del server di database.
`serverType`	string	Il tipo di server di database, ad esempio `MySQL`.
`serverVersion`	stringa	La versione del server di database. Attualmente, questo valore è sempre `MySQL 5.7.12` per Aurora MySQL.
`serviceName`	stringa	Il nome del servizio Attualmente, questo valore è sempre `Amazon Aurora MySQL` per Aurora MySQL.
`sessionId`	int	Un identificatore di sessione pseudo-univoco.
`startTime` (solo record di attività del database versione 1.1)	stringa	L'ora di inizio dell'esecuzione dell'istruzione SQL. È rappresentato in formato UTC (Coordinated Universal Time). Per calcolare il tempo di esecuzione dell'istruzione SQL, utilizzare `endTime - startTime` . Vedere anche il campo `endTime`.

Campo	Tipo di dati	Descrizione
`statementId`	int	Identificatore per l'istruzione SQL del client. Il contatore aumenta con ogni istruzione SQL immessa dal client. Il contatore viene reimpostato quando viene riavviata l'istanza database.
`substatementId`	int	Identificatore di una subistruzione SQL. Questo valore è 1 per gli eventi con classe `MAIN` e 2 per gli eventi con classe `AUX`. Utilizzare il campo `statementId` per identificare tutti gli eventi generati dalla stessa istruzione.
`transactionId` (solo record di attività del database versione 1.2)	int	Identificatore di una transazione.
`type`	stringa	Il tipo di evento, I valori validi sono `record` e `heartbeat` .

Elaborazione di un flusso di attività del database utilizzando AWS SDK

È possibile elaborare a livello di codice un flusso di attività utilizzando. AWS SDK Di seguito sono riportati esempi Java e Python completamente funzionanti dell'elaborazione del flusso di dati Kinesis.

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.util.HashMap;
import java.util.List;
```

```
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {
```

```
    private static final String STREAM_NAME = "aws-rds-das-[cluster-external-
resource-id]";
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String DBC_RESOURCE_ID = "[cluster-external-resource-id]";
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
    private static final BasicAWSCredentials CREDENTIALS = new
 BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
    private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
 AWSStaticCredentialsProvider(CREDENTIALS);

    private static final AwsCrypto CRYPTO = new AwsCrypto();
    private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
            .withRegion(REGION_NAME)
            .withCredentials(CREDENTIALS_PROVIDER).build();

    class Activity {
        String type;
        String version;
        String databaseActivityEvents;
        String key;
    }

    class ActivityEvent {
        @SerializedName("class") String _class;
        String clientApplication;
        String command;
        String commandText;
        String databaseName;
        String dbProtocol;
        String dbUserName;
        String endTime;
        String errorMessage;
        String exitCode;
        String logTime;
        String netProtocol;
        String objectName;
        String objectType;
        List<String> paramList;
```

```
        String pid;
        String remoteHost;
        String remotePort;
        String rowCount;
        String serverHost;
        String serverType;
        String serverVersion;
        String serviceName;
        String sessionId;
        String startTime;
        String statementId;
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId;
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }

    static class RecordProcessorFactory implements IRecordProcessorFactory {
        @Override
        public IRecordProcessor createProcessor() {
            return new RecordProcessor();
        }
    }

    static class RecordProcessor implements IRecordProcessor {

        private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
        private static final int PROCESSING_RETRIES_MAX = 10;
        private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
        private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

        private static final Cipher CIPHER;
        static {
            Security.insertProviderAt(new BouncyCastleProvider(), 1);
            try {
                CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
```

```
            } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
                throw new ExceptionInInitializerError(e);
            }
        }

        private long nextCheckpointTimeInMillis;

        @Override
        public void initialize(String shardId) {
        }

        @Override
        public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
            for (final Record record : records) {
                processSingleBlob(record.getData());
            }

            if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
                checkpoint(checkpointer);
                nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
            }
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }

        private void processSingleBlob(final ByteBuffer bytes) {
            try {
                // JSON $Activity
                final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

                // Base64.Decode
                final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
                final byte[] decodedDataKey = Base64.decode(activity.key);
```

```
                Map<String, String> context = new HashMap<>();
                context.put("aws:rds:dbc-id", DBC_RESOURCE_ID);

                // Decrypt
                final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
                final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
                final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));

                // GZip Decompress
                final byte[] decompressed = decompress(decrypted);
                // JSON $ActivityRecords
                final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

                // Iterate throught $ActivityEvents
                for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                    System.out.println(GSON.toJson(event));
                }
            } catch (Exception e) {
                // Handle error.
                e.printStackTrace();
            }
        }

        private static byte[] decompress(final byte[] src) throws IOException {
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
                    break;
                } catch (ShutdownException se) {
```

```
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
            return out.toByteArray();
        }
```

```
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
 ":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
 CREDENTIALS_PROVIDER, workerId);

 kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
 APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
    }
}
```

Python

```
import base64
import json
import zlib
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
```

```
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # cluster-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-cluster-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL

class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                        wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)

    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)
```

```
def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],

 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:dbc-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

IAMesempi di policy per i flussi di attività del database

Queste operazioni sono incluse nel registro di controllo del flusso. Per le migliori pratiche di conformità, ti consigliamo di non fornire questi privilegi a. DBAs

L'accesso ai flussi di attività del database viene impostato utilizzando IAM le policy. Per ulteriori informazioni sull'RDSautenticazione Aurora , consulta. Gestione delle identità e degli accessi per Amazon Aurora Per ulteriori informazioni sulla creazione di IAM policy, consultaCreazione e utilizzo di una policy IAM per l'accesso al database IAM.

Example Policy per consentire la configurazione dei flussi di attività di database

Per offrire agli utenti un accesso granulare alla modifica dei flussi di attività, utilizza le chiavi di contesto operative specifiche del servizio e in una policy. `rds:StartActivityStream` `rds:StopActivityStream` IAM Il seguente esempio di IAM policy consente a un utente o a un ruolo di configurare i flussi di attività.

```
{
   "Version":"2012-10-17",
   "Statement":[
        {
            "Sid":"ConfigureActivityStreams",
            "Effect":"Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
            "Resource":"*",
          }
     ]
}
```

Example Policy per consentire l'avvio di flussi di attività di database

Il seguente esempio di IAM policy consente a un utente o a un ruolo di avviare flussi di attività.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
```

```
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
    ]
}
```

Example Policy per consentire l'interruzione di flussi di attività di database

Il seguente esempio di IAM policy consente a un utente o a un ruolo di interrompere i flussi di attività.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Policy per rifiutare l'avvio di flussi di attività di database

Il seguente esempio di IAM policy impedisce a un utente o a un ruolo di avviare flussi di attività.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example Policy per rifiutare l'interruzione di flussi di attività di database

Il seguente esempio di IAM policy impedisce a un utente o a un ruolo di interrompere i flussi di attività.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Monitoraggio delle minacce con Amazon GuardDuty RDS Protection per Amazon Aurora

Amazon GuardDuty è un servizio di rilevamento delle minacce che aiuta a proteggere account, contenitori, carichi di lavoro e dati all'interno del tuo AWS ambiente. Utilizzando modelli di machine learning (ML) e funzionalità di rilevamento di anomalie e minacce, monitora GuardDuty continuamente diverse fonti di log e attività di runtime per identificare e dare priorità ai potenziali rischi per la sicurezza e alle attività dannose nel tuo ambiente.

GuardDuty RDS Protection analizza e profila gli eventi di accesso per potenziali minacce di accesso ai tuoi database RDS. La protezione RDS monitora questi eventi e li profila per potenziali minacce interne o attori esterni.

Per ulteriori informazioni sull'attivazione di GuardDuty RDS Protection, consulta GuardDuty RDS Protection nella Amazon GuardDuty User Guide.

Quando RDS Protection rileva una potenziale minaccia, ad esempio uno schema insolito nei tentativi di accesso riusciti o falliti, GuardDuty genera una nuova scoperta con dettagli sul database potenzialmente compromesso. Puoi visualizzare i dettagli dei risultati nella sezione di riepilogo dei risultati nella GuardDuty console Amazon. I dettagli dell'esito variano in base al tipo di esito. I dettagli principali, il tipo di risorsa e il ruolo della risorsa determinano il tipo di informazioni disponibili per qualsiasi esito. Per ulteriori informazioni sui dettagli comunemente disponibili sui risultati e sui tipi di risultati, consulta rispettivamente i dettagli di ricerca e i tipi di risultati GuardDuty RDS Protection nella Amazon GuardDuty User Guide.

Puoi attivare o disattivare la funzionalità di protezione RDS Regione AWS ovunque sia disponibile. Account AWS Quando la protezione RDS non è abilitata, GuardDuty non rileva i database Aurora potenzialmente compromessi né fornisce dettagli sulla compromissione.

Un GuardDuty account esistente può abilitare RDS Protection con un periodo di prova di 30 giorni. Per un nuovo GuardDuty account, RDS Protection è già abilitato e incluso nel periodo di prova gratuito di 30 giorni. Per ulteriori informazioni, consulta la sezione Stima GuardDuty dei costi nella Amazon GuardDuty User Guide.

Per informazioni su Regioni AWS dove GuardDuty non supporta ancora la protezione RDS, consulta la disponibilità delle funzionalità specifiche per regione nella Amazon GuardDuty User Guide.

Per informazioni sulle versioni del database Aurora supportate da GuardDuty RDS Protection, consulta i database Amazon Aurora, Amazon RDS e Aurora Limitless supportati nella Amazon User Guide. GuardDuty

Utilizzo di Amazon Aurora MySQL

Amazon Aurora MySQL è un motore del database relazionale completamente gestito compatibile con MySQL che unisce la velocità e l'affidabilità dei database commerciali di fascia alta alla semplicità e al costo ridotto dei database open source. Aurora MySQL è un sostituto drop-in di MySQL e semplifica e rende più conveniente dal punto di vista dei costi la configurazione, l'utilizzo e il dimensionamento delle implementazioni MySQL nuove ed esistenti, consentendo di concentrarti sulle tue attività e applicazioni. Amazon RDS fornisce l'amministrazione di Aurora gestendo le attività di database di routine, come il provisioning, l'applicazione di patch, il backup, il ripristino, il rilevamento degli errori e la correzione. Amazon RDS fornisce strumenti di migrazione per convertire, con la semplice pressione di un pulsante, le applicazioni Amazon RDS for MySQL e Amazon RDS for PostgreSQL in Aurora MySQL.

Argomenti

- Panoramica di Amazon Aurora MySQL
- Sicurezza con Amazon Aurora MySQL
- Aggiornamento delle applicazioni per la connessione ai cluster database Aurora MySQL utilizzando nuovi certificati TLS
- Utilizzo dell'autenticazione Kerberos per Aurora My SQL
- Migrazione dei dati verso un cluster Amazon Aurora My DB SQL
- Gestione di Amazon Aurora MySQL
- Ottimizzazione di Aurora MySQL
- Interrogazione parallela per Amazon Aurora My SQL
- Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL
- Replica con Amazon Aurora MySQL
- Utilizzo dell'inoltro di scrittura locale in un cluster database Amazon Aurora MySQL
- Integrazione di Amazon Aurora SQL My con altri servizi AWS
- Modalità di laboratorio per Amazon Aurora MySQL
- Le migliori pratiche con Amazon Aurora My SQL
- Risoluzione dei problemi relativi alle prestazioni del database Amazon Aurora My SQL
- Amazon Aurora La mia referenza SQL
- Aggiornamenti del motore di database per Amazon Aurora My SQL

Panoramica di Amazon Aurora MySQL

Nelle sezioni seguenti viene offerta una panoramica di Amazon Aurora MySQL.

Argomenti

- Miglioramenti alle prestazioni di Amazon Aurora MySQL
- Amazon Aurora MySQL e dati spaziali
- Aurora My SQL versione 3 compatibile con My 8.0 SQL
- Aurora My SQL versione 2 compatibile con My 5.7 SQL

Miglioramenti alle prestazioni di Amazon Aurora MySQL

Amazon Aurora include opzioni che migliorano le prestazioni per supportare le differenti necessità dei database commerciali di fascia alta.

Inserimento rapido

L'inserimento rapido accelera gli inserimenti paralleli ordinati in base alla chiave primaria e si applica specificamente alle istruzioni `LOAD DATA` e `INSERT INTO ... SELECT ...`. L'inserimento rapido memorizza nella cache la posizione del cursore in un indice traversale mentre esegue l'istruzione. Evita l'attraversamento inutile nuovamente dell'indice.

L'inserimento rapido è abilitato solo per le normali tabelle InnoDB in Aurora MySQL versione 3.03.2 e successive. Questa ottimizzazione non funziona per le tabelle temporanee InnoDB. È disabilitato in Aurora MySQL versione 2 per tutte le versioni 2.11 e 2.12. L'ottimizzazione rapida degli inserti funziona solo se l'ottimizzazione dell'Adaptive Hash Index è disabilitata.

Puoi monitorare i seguenti parametri per determinare l'efficacia dell'inserimento rapido per il cluster di database:

- `aurora_fast_insert_cache_hits`: un contatore che viene aumentato quando il cursore memorizzato nella cache viene recuperato e verificato.
- `aurora_fast_insert_cache_misses`: un contatore che viene aumentato quando il cursore memorizzato non è più valido e Aurora esegue un attraversamento di indice normale.

Puoi recuperare il valore corrente dei parametri di inserimento rapido utilizzando il seguente comando:

```
mysql> show global status like 'Aurora_fast_insert%';
```

Otterrai un output simile al seguente:

```
+--------------------------------+-----------+
| Variable_name                  | Value     |
+--------------------------------+-----------+
| Aurora_fast_insert_cache_hits  | 3598300   |
| Aurora_fast_insert_cache_misses | 436401336 |
+--------------------------------+-----------+
```

Amazon Aurora MySQL e dati spaziali

Nell'elenco seguente vengono riepilogate le principali caratteristiche spaziali di Aurora MySQL e descritto come queste corrispondono alle caratteristiche spaziali in MySQL:

- Aurora MySQL versione 2 supporta gli stessi tipi di dati spaziali e funzioni di relazioni spaziali di MySQL 5.7. Per ulteriori informazioni su questi tipi di dati e funzioni, consulta Tipi di dati spaziali e Funzioni di relazione spaziale nella documentazione MySQL 5.7.
- Aurora MySQL versione 3 supporta gli stessi tipi di dati spaziali e funzioni di relazioni spaziali di MySQL 8.0. Per ulteriori informazioni su questi tipi di dati e funzioni, consulta Tipi di dati spaziali e Funzioni di relazione spaziale nella documentazione MySQL 8.0.
- Aurora MySQL supporta l'indicizzazione spaziale su tabelle InnoDB. L'indicizzazione spaziale consente di migliorare le prestazioni su set di dati di grandi dimensioni per le query sui dati spaziali. In MySQL, l'indicizzazione spaziale per le tabelle InnoDB è disponibile in MySQL 5.7 e 8.0.

  Aurora MySQL usa una diversa strategia di indicizzazione spaziale rispetto a MySQL per alte prestazioni con le query spaziali. L'implementazione dell'indice spaziale Aurora utilizza una curva di riempimento dello spazio in un albero B, finalizzata a fornire prestazioni più elevate per scansioni degli intervalli spaziali rispetto a un albero R.

  > **Note**
  >
  > In Aurora MySQL, una transazione su una tabella con un indice spaziale definito su una colonna con un identificatore di riferimento spaziale (SRID) non può essere inserita in un'area selezionata per l'aggiornamento da un'altra transazione.

Le seguenti istruzioni DDL sono supportate per la creazione di indici su colonne che utilizzano tipi di dati spaziali.

CREATE TABLE

Puoi utilizzare le parole chiave `SPATIAL INDEX` in un'istruzione `CREATE TABLE` per aggiungere un indice spaziale a una colonna in una nuova tabella. Di seguito è riportato un esempio.

```
CREATE TABLE test (shape POLYGON NOT NULL, SPATIAL INDEX(shape));
```

ALTER TABLE

Puoi utilizzare le parole chiave `SPATIAL INDEX` in un'istruzione `ALTER TABLE` per aggiungere un indice spaziale a una colonna in una tabella esistente. Di seguito è riportato un esempio.

```
ALTER TABLE test ADD SPATIAL INDEX(shape);
```

CREATE INDEX

Puoi utilizzare la parola chiave `SPATIAL` in un'istruzione `CREATE INDEX` per aggiungere un indice spaziale a una colonna in una tabella esistente. Di seguito è riportato un esempio.

```
CREATE SPATIAL INDEX shape_index ON test (shape);
```

Aurora My SQL versione 3 compatibile con My 8.0 SQL

Puoi usare Aurora My SQL versione 3 per ottenere le ultime funzionalità SQL compatibili con My, miglioramenti delle prestazioni e correzioni di bug. Di seguito, puoi scoprire Aurora My SQL versione 3, con compatibilità con My SQL 8.0. Puoi scoprire come aggiornare i tuoi cluster e le tue applicazioni alla versione 3 di Aurora SQL My.

Alcune funzionalità di Aurora, come Aurora Serverless v2, richiede Aurora SQL My versione 3.

Argomenti

- Funzionalità di My SQL 8.0 Community Edition
- Aurora My SQL versione 3: prerequisito per Aurora My Serverless v2 SQL
- Note di rilascio per Aurora My SQL versione 3
- Nuove ottimizzazioni delle query parallele

- Ottimizzazioni per ridurre i tempi di riavvio del database
- Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3
- Confronto tra Aurora My SQL versione 2 e Aurora My versione 3 SQL
- Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition
- Aggiornamento ad Aurora MySQL versione 3

Funzionalità di My SQL 8.0 Community Edition

La versione iniziale di Aurora My SQL versione 3 è compatibile con My SQL 8.0.23 Community Edition. My SQL 8.0 introduce diverse nuove funzionalità, tra cui:

- Supporto per Atomic Data Definition Language ()DDL. Per ulteriori informazioni, consulta Supporto per Atomic Data Definition Language () DDL.
- JSONfunzioni. Per informazioni sull'utilizzo, vedere JSONFunzioni nel manuale My SQL Reference.
- Funzioni finestra Per informazioni sull'utilizzo, vedere Window Functions nel My SQL Reference Manual.
- Espressioni di tabella comuni (CTEs), utilizzando la `WITH` clausola. Per informazioni sull'utilizzo, vedere WITH(Common Table Expressions) nel My SQL Reference Manual.
- Ottimizzazione `ADD COLUMN` e clausole `RENAME COLUMN` per l'istruzione `ALTER TABLE`. Queste ottimizzazioni sono chiamate «istantanee»DDL. Aurora My SQL version 3 è compatibile con la funzione My SQL instant DDL della community. La precedente DDL funzionalità Aurora fast non viene utilizzata. Per informazioni istantanee sull'utilizzoDDL, consultaDDL istantaneo (Aurora MySQL versione 3).
- Indici discendenti, funzionali e invisibili. Per informazioni sull'utilizzo, vedere Indici invisibili, Indici decrescenti e CREATEINDEXdichiarazione nel My Reference Manual. SQL
- Privilegi basati sui ruoli controllati tramite istruzioni. SQL Per ulteriori informazioni sulle modifiche al modello di privilegi, consulta Privilegio basato sui ruoli.
- clausole `NOWAIT` e `SKIP LOCKED` con istruzioni `SELECT ... FOR SHARE`. Queste clausole evitano di attendere che altre transazioni rilascino blocchi di riga. Per informazioni sull'utilizzo, vedere Locking Reads nel My Reference Manual. SQL
- Miglioramenti alla replica dei log binari (binlog). Per Aurora I miei SQL dati, vedi. Replica dei log binari In particolare, è possibile eseguire la replica filtrata. Per informazioni sull'utilizzo della replica filtrata, vedere How Servers Evaluate Replication Filtering Rules nel My Reference Manual. SQL

- Suggerimenti. Alcuni suggerimenti SQL compatibili con My 8.0 erano già stati trasferiti in Aurora My versione 2. SQL Per informazioni sull'uso dei suggerimenti con Aurora SQL My, consulta. Aurora I miei suggerimenti SQL Per l'elenco completo dei suggerimenti disponibili nella community My SQL 8.0, consulta Optimizer Hints nel My Reference Manual. SQL

Per l'elenco completo delle funzionalità aggiunte all'edizione comunitaria di My SQL 8.0, consulta il post sul blog L'elenco completo delle nuove funzionalità in My 8.0. SQL

Aurora My SQL versione 3 include anche modifiche alle parole chiave per un linguaggio inclusivo, supportate dalla community My 8.0.26. SQL Per i dettagli su tali modifiche, consultare Modifiche linguistiche incluse per Aurora My versione 3 SQL.

Aurora My SQL versione 3: prerequisito per Aurora My Serverless v2 SQL

Aurora My SQL versione 3 è un prerequisito per tutte le istanze DB in un cluster Aurora My Serverless v2. SQL Aurora My SQL Serverless v2 include il supporto per le istanze di lettura in un cluster DB e altre funzionalità di Aurora che non sono disponibili per Aurora My Serverless v1. SQL Ha anche una scalabilità più rapida e granulare rispetto ad Aurora My Serverless v1. SQL

Note di rilascio per Aurora My SQL versione 3

Per le note di rilascio per tutte le versioni di Aurora My SQL versione 3, consulta Aggiornamenti del motore di database per Amazon Aurora SQL My versione 3 nelle Note di rilascio per Aurora My. SQL

Nuove ottimizzazioni delle query parallele

L'ottimizzazione delle query parallele Aurora ora si applica a più SQL operazioni:

- La query parallela ora si applica alle tabelle contenenti i tipi di dati `TEXT`, `BLOB`, `JSON`, `GEOMETRY`, e `VARCHAR` e `CHAR` superiori a 768 byte.
- La query parallela può ottimizzare le query che coinvolgono tabelle partizionate.
- La query parallela può ottimizzare le query che coinvolgono chiamate di funzioni aggregate nell'elenco di selezione e nella clausola `HAVING`.

Per ulteriori informazioni su questi miglioramenti, consultare Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL. Per informazioni generali sulle query parallele Aurora, consultare Interrogazione parallela per Amazon Aurora My SQL.

Ottimizzazioni per ridurre i tempi di riavvio del database

Il cluster Aurora My SQL DB deve essere altamente disponibile durante le interruzioni pianificate e non pianificate.

Gli amministratori di database devono eseguire la manutenzione occasionale del database. Attività incluse nella manutenzione sono l'applicazione di patch al database, gli aggiornamenti, la modifica dei parametri del database che richiedono un riavvio manuale, l'esecuzione di un failover per ridurre il tempo impiegato da un'istanza per modificare la classe. Queste operazioni pianificate comportano tempi di inattività.

Tuttavia, i tempi di inattività possono essere causati anche da operazioni non pianificate, come un failover imprevisto dovuto a un guasto hardware sottostante o a una limitazione delle risorse del database. Tutte queste operazioni pianificate e non pianificate comportano il riavvio del database.

In Aurora My SQL versione 3.05 e successive, abbiamo introdotto ottimizzazioni che riducono il tempo di riavvio del database. Queste ottimizzazioni consentono di ridurre fino al 65% i tempi di inattività rispetto alle ottimizzazioni precedenti e di ridurre le interruzioni dei carichi di lavoro del database dopo un riavvio.

Durante l'avvio del database, vengono inizializzati molti componenti della memoria interna. Il più grande di questi è il buffer pool InnoDB, che in Aurora My SQL rappresenta per impostazione predefinita il 75% della dimensione della memoria dell'istanza. I nostri test hanno rilevato che il tempo di inizializzazione è proporzionale alla dimensione del pool di buffer InnoDB e che si dimensiona in base alla grandezza della classe dell'istanza del database. Durante questa fase di inizializzazione, il database non può accettare connessioni, il che causa tempi di inattività più lunghi durante i riavvii. La prima fase di Aurora My SQL fast restart ottimizza l'inizializzazione del buffer pool, riducendo il tempo di inizializzazione del database e quindi il tempo di riavvio complessivo.

Per maggiori dettagli, consulta il blog Riduci i tempi di inattività con Amazon Aurora Le ottimizzazioni del tempo di riavvio del SQL mio database.

Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3

Aurora MySQL versione 3 gestisce le tabelle temporanee in maniera diversa dalle versioni precedenti di Aurora MySQL. Questo nuovo comportamento è ereditato dalla edizione della comunità MySQL 8.0. Esistono due tipi di tabelle temporanee che possono essere create con Aurora MySQL versione 3:

- Tabelle temporanee interne (o implicite): create dal motore MySQL di Aurora per gestire operazioni come l'aggregazione di ordinamento, le tabelle derivate o le espressioni di tabella comuni (). CTEs
- Tabelle temporanee create dall'utente (o esplicite): create dal motore Aurora MySQL quando si utilizza l'istruzione `CREATE TEMPORARY TABLE`.

Ci sono ulteriori considerazioni per le tabelle temporanee interne e create dall'utente sulle istanze database di lettura Aurora. Queste modifiche vengono illustrate nelle sezioni seguenti.

Argomenti

- Motore di storage per tabelle temporanee (implicite) interne
- Limitazione delle dimensioni delle tabelle temporanee interne in memoria
- Mitigazione dei problemi di pienezza per le tabelle temporanee interne su repliche Aurora
- Ottimizzazione del parametro temptable_max_mmap sulle istanze Aurora MySQL DB
- Tabelle temporanee create dall'utente (esplicite) su istanze database di lettura
- Errori di creazione tabelle temporanee e mitigazione

Motore di storage per tabelle temporanee (implicite) interne

Quando si generano set di risultati intermedi, Aurora MySQL tenta inizialmente di scrivere nelle tabelle temporanee in memoria. Questa operazione potrebbe non riuscire, a causa di tipi di dati incompatibili o limiti configurati. In tal caso, la tabella temporanea viene convertita in una tabella temporanea su disco anziché essere conservata in memoria. Ulteriori informazioni sono disponibili in Utilizzo della tabella temporanea interna in MySQL nella documentazione MySQL.

In Aurora MySQL versione 3, il modo in cui funzionano le tabelle temporanee interne è diverso dalle versioni precedenti di Aurora MySQL. Invece di scegliere tra i motori di archiviazione InnoDB e MyISAM per tali tabelle temporanee, ora scegli tra i motori di archiviazione e. `TempTable` `MEMORY`

Con il motore di archiviazione `TempTable`, è possibile fare una scelta aggiuntiva su come gestire determinati dati. I dati interessati fanno traboccare il pool di memoria che contiene tutte le tabelle temporanee interne per l'istanza database.

Tali scelte possono influenzare le prestazioni delle query che generano volumi elevati di dati temporanei, ad esempio durante l'esecuzione di aggregazioni come `GROUP BY` su tabelle grandi.

> **Tip**
>
> Se il carico di lavoro include query che generano tabelle temporanee interne, verificare le prestazioni dell'applicazione con questa modifica eseguendo benchmark e monitorando le metriche relative alle prestazioni.
> In alcuni casi, la quantità di dati temporanei si inserisce nella pool di memoria `TempTable` o fa solo traboccare il pool di memoria in piccola misura. In questi casi, si consiglia di utilizzare l'impostazione `TempTable` per tabelle temporanee interne e file mappati in memoria per contenere i dati di overflow. Questa è l'impostazione di default.

Il motore di archiviazione di `TempTable` è il valore di default. `TempTable` utilizza un pool di memoria comune per tutte le tabelle temporanee che utilizzano questo motore, anziché un limite massimo di memoria per tabella. La dimensione di questo pool di memoria è specificata dal parametro temptable_max_ram. Il valore predefinito è 1 GiB su istanze database con 16 o più GiB di memoria e 16 MB su istanze database con meno di 16 GiB di memoria. La dimensione del pool di memoria influisce sul consumo di memoria a livello di sessione.

In alcuni casi, quando si utilizza il motore di archiviazione `TempTable` i dati temporanei potrebbero superare le dimensioni del pool di memoria. In tal caso, Aurora MySQL archivia i dati di overflow utilizzando un meccanismo secondario.

È possibile impostare il parametro temptable_max_mmap per scegliere se i dati eseguono l'overflow in file temporanei mappati in memoria o in tabelle temporanee interne di InnoDB su disco. I diversi formati di dati e i criteri di overflow di questi meccanismi di overflow possono influire sulle prestazioni delle query. Lo fanno influenzando la quantità di dati scritti su disco e la domanda sulla velocità effettiva di archiviazione su disco.

Aurora MySQL versione 3 archivia i dati di overflow nel modo seguente:

- Nell'istanza Writer DB, i dati che fuoriescono verso le tabelle temporanee interne di InnoDB o i file temporanei mappati in memoria risiedono nella memoria locale dell'istanza.
- Nelle istanze Reader DB, i dati di overflow risiedono sempre in file temporanei mappati in memoria nell'archiviazione locale.

  Le istanze di sola lettura non possono archiviare alcun dato sul volume del cluster Aurora.

I parametri di configurazione relativi alle tabelle temporanee interne si applicano in modo diverso alle istanze di scrittura e lettura nel cluster:

- Nelle istanze di lettura, Aurora MySQL utilizza sempre il motore di archiviazione `TempTable`.
- La dimensione per il valore di `temptable_max_mmap` predefinito è 1 GiB, per entrambe le istanze di scrittura e lettura, indipendentemente dalle dimensioni della memoria dell'istanza database. È possibile modificare questo valore sia nelle istanze di scrittura che in quelle di lettura.
- L'impostazione di `temptable_max_mmap` su `0` disattiva l'uso dei file temporanei mappati in memoria nelle istanze di scrittura.
- Non puoi impostare `temptable_max_mmap` su `0` nelle istanze di lettura.

> **Note**
>
> Ti consigliamo di non utilizzare il parametro temptable_use_mmap. È deprecato e il relativo supporto verrà rimosso in una versione futura di MySQL.

Limitazione delle dimensioni delle tabelle temporanee interne in memoria

Come discusso in Motore di storage per tabelle temporanee (implicite) interne, è possibile controllare le risorse temporanee delle tabelle a livello globale utilizzando le impostazioni temptable max_ram e temptable_max_mmap.

Inoltre, è possibile limitare le dimensioni di ogni singola tabella temporanea interna in memoria utilizzando il parametro database tmp_table_size. Questo limite serve a evitare che le singole query consumino una quantità eccessiva di risorse globali delle tabelle temporanee, che può influire sulle prestazioni delle query simultanee che richiedono tali risorse.

Il parametro `tmp_table_size` definisce le dimensioni massime delle tabelle temporanee create dal motore di storage `MEMORY` in Aurora MySQL versione 3.

In Aurora MySQL versione 3.04 e successive, il parametro `tmp_table_size` definisce anche le dimensioni massime delle tabelle temporanee create dal motore di storage `TempTable` quando il parametro database `aurora_tmptable_enable_per_table_limit` è impostato su `ON`. Questo comportamento è disabilitato per impostazione predefinita (`OFF`), che è identico al comportamento in Aurora MySQL versione 3.03 e precedenti.

- Quando `aurora_tmptable_enable_per_table_limit` è `OFF`, `tmp_table_size` non è considerato per le tabelle temporanee interne in memoria create dal motore di storage `TempTable`.

  Tuttavia, il limite di risorse `TempTable` globali rimane comunque in essere. Aurora MySQL presenta il seguente comportamento quando viene raggiunto il limite di risorse `TempTable` globale:

  - Istanze database di scrittura: Aurora MySQL converte automaticamente la tabella temporanea in memoria in una tabella temporanea su disco InnoDB.
  - Istanze database di lettura: la query termina con un errore.

  ```
  ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
  ```

- Quando `aurora_tmptable_enable_per_table_limit` è `ON`, Aurora MySQL presenta il seguente comportamento quando viene raggiunto il limite `tmp_table_size`:

  - Istanze database di scrittura: Aurora MySQL converte automaticamente la tabella temporanea in memoria in una tabella temporanea su disco InnoDB.
  - Istanze database di lettura: la query termina con un errore.

  ```
  ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
  ```

  Il limite di risorse `TempTable` globale e il limite per tabella si applicano entrambi in questo caso.

> **Note**
>
> Il parametro `aurora_tmptable_enable_per_table_limit` non ha effetto quando internal_tmp_mem_storage_engine è impostato su `MEMORY`. In questo caso, le dimensioni massime di una tabella temporanea in memoria sono definite dal valore tmp_table_size o dimensione max_heap_table_size, a seconda di quale sia il più piccolo.

Gli esempi seguenti mostrano il comportamento del parametro `aurora_tmptable_enable_per_table_limit`per le istanze database di scrittura e lettura.

Example dell'istanza database di scrittura con **`aurora_tmptable_enable_per_table_limit`** impostato su **`OFF`**

La tabella temporanea in memoria non viene convertita in una tabella temporanea su disco InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
|                  0 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (13.99 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
```

```
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example dell'istanza database di scrittura
con **aurora_tmptable_enable_per_table_limit** impostato su **ON**

La tabella temporanea in memoria viene convertita in una tabella temporanea su disco InnoDB.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+-----------------+-----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+-----------------+-----------------------------------------
+------------------+
|                  0 | 3.04.0          |                                        1 |
      16777216 |
+--------------------+-----------------+-----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
+---------+
| max(n)  |
```

```
+---------+
| 6000000 |
+---------+
1 row in set (4.10 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 1     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example dell'istanza database di lettura con **aurora_tmptable_enable_per_table_limit** impostato su **OFF**

La query termina senza errori perché `tmp_table_size` non è applicabile e il limite di risorse `TempTable` globale non è stato raggiunto.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
```

```
+----------+
| 60000000 |
+----------+
1 row in set (14.05 sec)
```

Example dell'istanza database di lettura con **aurora_tmptable_enable_per_table_limit** impostato su **OFF**

Questa query raggiunge il limite di TempTable risorse globali se impostata su OFF. `aurora_tmptable_enable_per_table_limit` La query termina con un errore su istanze di lettura.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+-----------------------------------------
+----------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+--------------------+------------------+-----------------------------------------
+----------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+--------------------+------------------+-----------------------------------------
+----------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.01 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 120000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_1586_2' is full
```

Example dell'istanza database di lettura con **aurora_tmptable_enable_per_table_limit** impostato su **ON**

La query termina con un errore quando viene raggiunto il limite `tmp_table_size`.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+------------------+----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+------------------+----------------------------------------
+------------------+
|                  1 | 3.04.0           |                                        1 |
      16777216 |
+--------------------+------------------+----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_8_2' is full
```

Mitigazione dei problemi di pienezza per le tabelle temporanee interne su repliche Aurora

Per evitare problemi di limitazione delle dimensioni per le tabelle temporanee, impostare i parametri `temptable_max_ram` e `temptable_max_mmap` su un valore combinato in grado di soddisfare i requisiti del carico di lavoro.

Presta attenzione durante l'impostazione del valore del parametro `temptable_max_ram`. L'impostazione di un valore troppo alto riduce la memoria disponibile sull'istanza del database, il che può causare una out-of-memory condizione. Monitora la memoria liberabile media sull'istanza database. Quindi determina un valore appropriato per `temptable_max_ram` in modo da avere ancora una quantità ragionevole di memoria libera sull'istanza. Per ulteriori informazioni, consulta Problemi di memoria liberabile in Amazon Aurora.

È inoltre importante monitorare le dimensioni dello storage locale e il consumo temporaneo di spazio della tabella. Puoi monitorare lo storage temporaneo disponibile per una specifica istanza DB con il CloudWatch parametro `FreeLocalStorage` Amazon, descritto in CloudWatch Parametri Amazon per Amazon Aurora.

> **Note**
>
> Questa procedura non funziona quando il parametro `aurora_tmptable_enable_per_table_limit` è impostato su `0N`. Per ulteriori informazioni, consulta Limitazione delle dimensioni delle tabelle temporanee interne in memoria.

Example 1

È noto che le tabelle temporanee raggiungono una dimensione cumulativa di 20 GiB. Desideri impostare tabelle temporanee in memoria su 2 GiB e aumentare fino a un massimo di 20 GiB su disco.

Imposta `temptable_max_ram` su **`2,147,483,648`** e `temptable_max_mmap` su **`21,474,836,480`**. Questi valori sono espressi in byte.

Queste impostazioni dei parametri garantiscono che le dimensioni delle tabelle temporanee possano aumentare fino a un totale cumulativo di 22 GiB.

Example 2

La dimensione istanza corrente è 16xlarge o superiore. Non conosci le dimensioni totali delle tabelle temporanee che potrebbero essere richieste. Vuoi poter utilizzare fino a 4 GiB in memoria e fino alla dimensione di storage massima disponibile su disco.

Imposta `temptable_max_ram` su **`4,294,967,296`** e `temptable_max_mmap` su **`1,099,511,627,776`**. Questi valori sono espressi in byte.

Qui stai impostando `temptable_max_mmap` su 1 TiB, che è inferiore allo storage locale massimo di 1,2 TiB su un'istanza database di Aurora 16xlarge.

Su una dimensione istanza più piccola, regola il valore di `temptable_max_mmap` in modo che non riempia lo storage locale disponibile. Ad esempio, un'istanza 2xlarge dispone solo di 160 GiB di storage locale disponibile. Pertanto, si consiglia di impostare su un valore inferiore a 160 GiB. Per ulteriori informazioni sullo storage locale disponibile per le dimensioni delle istanze database, consulta Limiti di storage temporaneo per Aurora MySQL.

Ottimizzazione del parametro temptable_max_mmap sulle istanze Aurora MySQL DB

Il `temptable_max_mmap` parametro in Aurora MySQL controlla la quantità massima di spazio su disco locale che può essere utilizzata dai file mappati in memoria prima di sovraccaricare le tabelle temporanee InnoDB su disco (sulle istanze DB di scrittura) o causare un errore (sulle istanze DB di lettura). L'impostazione corretta di questo parametro dell'istanza DB può aiutare a ottimizzare le prestazioni delle istanze DB.

Prerequisiti

1. Assicurati che lo schema delle prestazioni sia abilitato. Puoi verificarlo eseguendo il seguente comando SQL:

   ```
   SELECT @@performance_schema;
   ```

   Un valore di output di 1 indica che è abilitato.

2. Verificare che la strumentazione di memoria della tabella temporanea sia abilitata. È possibile verificarlo eseguendo il seguente comando SQL:

   ```
   SELECT name, enabled FROM performance_schema.setup_instruments WHERE name LIKE
    '%memory%temptable%';
   ```

   La `enabled` colonna mostra `YES` le voci pertinenti della strumentazione di memoria della tabella temporanea.

Monitoraggio dell'utilizzo temporaneo delle tabelle

Quando si imposta il valore iniziale per`temptable_max_mmap`, si consiglia di iniziare con l'80% della dimensione di archiviazione locale per la classe di istanza DB che si sta utilizzando. Ciò garantisce che le tabelle temporanee dispongano di spazio su disco sufficiente per funzionare in modo efficiente, lasciando spazio per altri utilizzi del disco sull'istanza.

Per trovare la dimensione di archiviazione locale per la tua classe di istanza DB, consultaLimiti di storage temporaneo per Aurora MySQL.

Ad esempio, se utilizzi la classe di istanze DB db.r5.large, la dimensione di archiviazione locale è di 32 GiB. In questo caso, inizialmente dovreste impostare il `temptable_max_mmap` parametro sull'80% di 32 GiB, ovvero 25,6 GiB.

Dopo aver impostato il `temptable_max_mmap` valore iniziale, esegui il carico di lavoro di picco sulle istanze Aurora MySQL. Monitora l'utilizzo attuale e temporaneo del disco della tabella utilizzando la seguente query SQL:

```
SELECT event_name, current_count, current_alloc, current_avg_alloc, high_count,
 high_alloc, high_avg_alloc
FROM sys.memory_global_by_current_bytes WHERE event_name LIKE 'memory/temptable/%';
```

Questa query recupera le seguenti informazioni:

- `event_name`— Il nome dell'evento temporaneo di utilizzo della memoria o del disco nella tabella.
- `current_count`— Il numero corrente di blocchi di memoria o disco temporanei allocati nella tabella.
- `current_alloc`— La quantità corrente di memoria o disco allocata per le tabelle temporanee.
- `current_avg_alloc`— La dimensione media attuale della memoria temporanea della tabella o dei blocchi di disco.
- `high_count`— Il numero massimo di blocchi di memoria o disco temporanei allocati nella tabella.
- `high_alloc`— La quantità massima di memoria o disco allocata per le tabelle temporanee.
- `high_avg_alloc`— La dimensione media più elevata della memoria della tabella temporanea o dei blocchi di disco.

Se le query hanno esito negativo con un errore Table is full utilizzando questa impostazione, significa che il carico di lavoro richiede più spazio su disco per le operazioni temporanee sulle tabelle. In questo caso, valuta la possibilità di aumentare la dimensione dell'istanza DB a un'istanza con più spazio di archiviazione locale.

Impostazione del **`temptable_max_mmap`** valore ottimale

Utilizzate la seguente procedura per monitorare e impostare la dimensione corretta per il `temptable_max_mmap` parametro.

1. Esaminate l'output della query precedente e identificate il picco di utilizzo del disco della tabella temporanea, come indicato dalla `high_alloc` colonna.
2. In base al picco di utilizzo del disco della tabella temporanea, regola il `temptable_max_mmap` parametro nel gruppo di parametri DB per le tue istanze DB Aurora MySQL.

Imposta il valore in modo che sia leggermente superiore al picco di utilizzo dei dischi tabellari temporanei per far fronte alle crescite future.

3. Applica le modifiche al gruppo di parametri alle tue istanze DB.
4. Monitora nuovamente l'utilizzo dei dischi delle tabelle temporanee durante il picco di carico di lavoro per assicurarti che il nuovo `temptable_max_mmap` valore sia appropriato.
5. Ripetete i passaggi precedenti se necessario per ottimizzare il `temptable_max_mmap` parametro.

Tabelle temporanee create dall'utente (esplicite) su istanze database di lettura

Puoi creare tabelle temporanee esplicite utilizzando la parola chiave `TEMPORARY` nell'istruzione `CREATE TABLE`. Le tabelle temporanee esplicite sono supportate nell'istanza database di scrittura in un cluster di database Aurora. Puoi inoltre utilizzare tabelle temporanee esplicite sulle istanze database di lettura, ma le tabelle non possono applicare l'uso del motore di storage InnoDB.

Per evitare errori durante la creazione di tabelle temporanee esplicite su istanze database di lettura di Aurora MySQL, assicurati che tutte le istruzioni `CREATE TEMPORARY TABLE` vengano eseguite in uno o entrambi i seguenti modi:

- Non specificare la clausola `ENGINE=InnoDB`.
- Non impostare la modalità SQL su `NO_ENGINE_SUBSTITUTION`.

Errori di creazione tabelle temporanee e mitigazione

L'errore che si riceve è diverso a seconda che si utilizzi una semplice istruzione `CREATE TEMPORARY TABLE` o la variazione `CREATE TEMPORARY TABLE AS SELECT`. Nell'esempio seguente vengono illustrati tipi diversi di errori.

Questo comportamento temporaneo della tabella si applica solo alle istanze di sola lettura. Questo primo esempio conferma che è il tipo di istanza a cui è connessa la sessione.

```
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
```

Per istruzioni semplici `CREATE TEMPORARY TABLE`, l'istruzione fallisce quando la modalità SQL `NO_ENGINE_SUBSTITUTION` è attivata. Quando `NO_ENGINE_SUBSTITUTION` è disattivato (impostazione predefinita), viene effettuata la sostituzione motore appropriata e la creazione della tabella temporanea ha esito positivo.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql>  CREATE TEMPORARY TABLE tt2 (id int) ENGINE=InnoDB;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> CREATE TEMPORARY TABLE tt4 (id int) ENGINE=InnoDB;

mysql> SHOW CREATE TABLE tt4\G
*************************** 1. row ***************************
       Table: tt4
Create Table: CREATE TEMPORARY TABLE `tt4` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Per le istruzioni `CREATE TEMPORARY TABLE AS SELECT`, l'istruzione ha esito negativo quando la modalità SQL `NO_ENGINE_SUBSTITUTION` è attivata. Quando `NO_ENGINE_SUBSTITUTION` è disattivato (impostazione predefinita), viene effettuata la sostituzione motore appropriata e la creazione della tabella temporanea ha esito positivo.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql> CREATE TEMPORARY TABLE tt1 ENGINE=InnoDB AS SELECT * FROM t1;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> show create table tt3;
+-------+-----------------------------------------------------------+
| Table | Create Table                                              |
+-------+-----------------------------------------------------------+
| tt3   | CREATE TEMPORARY TABLE `tt3` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------+
```

```
1 row in set (0.00 sec)
```

Per ulteriori informazioni sugli aspetti di storage e le implicazioni prestazionali delle tabelle temporanee in Aurora MySQL versione 3, consulta il post del blog Use the TempTable storage engine on Amazon RDS for MySQL e Amazon Aurora MySQL.

Confronto tra Aurora My SQL versione 2 e Aurora My versione 3 SQL

Usa quanto segue per conoscere le modifiche da tenere a mente quando aggiorni il cluster Aurora My SQL versione 2 alla versione 3.

Argomenti

- Supporto per Atomic Data Definition Language () DDL
- Differenze di funzionalità tra Aurora My SQL versione 2 e 3
- Supporto delle classi di istanza
- Modifiche ai parametri per Aurora My versione 3 SQL
- Variabili di stato
- Modifiche linguistiche incluse per Aurora My versione 3 SQL
- AUTO_ valori INCREMENT
- Replica dei log binari

Supporto per Atomic Data Definition Language () DDL

Una delle maggiori modifiche da My SQL 5.7 a 8.0 è l'introduzione dell'Atomic Data Dictionary. Prima di My SQL 8.0, il dizionario My SQL data utilizzava un approccio basato su file per archiviare metadati come definizioni di tabelle (.frm), trigger (.trg) e funzioni separatamente dai metadati del motore di archiviazione (come quelli di InnoDB). Ciò presentava alcuni problemi, tra cui il rischio che le tabelle diventassero «orfane» in caso di imprevisto durante un'DDLoperazione, che causava la non sincronizzazione dei metadati basati su file e del motore di archiviazione.

Per risolvere questo problema, My SQL 8.0 ha introdotto l'Atomic Data Dictionary, che memorizza tutti i metadati in un set di tabelle InnoDB interne nello schema. `mysql` Questa nuova architettura offre un modo transazionale e ACIDconforme per gestire i metadati del database, risolvendo il problema «atomico» del vecchio approccio basato su file. DDL Per ulteriori informazioni sull'Atomic Data Dictionary, consulta Rimozione dell'archiviazione di metadati basata su file e Supporto per le dichiarazioni di definizione dei dati Atomic nel My Reference Manual. SQL

A causa di questa modifica dell'architettura, è necessario considerare quanto segue quando si esegue l'aggiornamento da Aurora My SQL versione 2 alla versione 3:

- I metadati basati su file della versione 2 devono essere migrati nelle nuove tabelle del dizionario dei dati durante il processo di aggiornamento alla versione 3. A seconda del numero di oggetti del database migrati, questa operazione potrebbe richiedere del tempo.
- Le modifiche hanno inoltre introdotto alcune nuove incompatibilità che potrebbero dover essere risolte prima di poter eseguire l'aggiornamento da My SQL 5.7 a 8.0. Ad esempio, 8.0 include alcune nuove parole chiave riservate che potrebbero entrare in conflitto con i nomi degli oggetti del database esistenti.

Per aiutarti a identificare queste incompatibilità prima di aggiornare il motore, Aurora My SQL esegue una serie di controlli di compatibilità degli aggiornamenti (precontrolli) per determinare se ci sono oggetti incompatibili nel dizionario del database, prima di eseguire l'aggiornamento del dizionario dei dati. Per ulteriori informazioni sui precontrolli, consulta. Controlli preliminari di aggiornamento delle versioni principali per Aurora My SQL

Differenze di funzionalità tra Aurora My SQL versione 2 e 3

Le seguenti SQL funzionalità di Amazon Aurora My sono supportate in Aurora My SQL for My SQL 5.7, ma non sono supportate in Aurora My for My 8.0: SQL SQL

- Non puoi usare Aurora My SQL version 3 per Aurora Serverless cluster v1. Aurora La mia SQL versione 3 funziona con Aurora Serverless v2.
- La modalità Lab non si applica alla SQL versione 3 di Aurora My. Non ci sono funzionalità in modalità lab in Aurora My SQL versione 3. Instant DDL sostituisce la DDL funzionalità online veloce precedentemente disponibile in modalità lab. Per vedere un esempio, consulta DDL istantaneo (Aurora MySQL versione 3).
- La cache delle query viene rimossa dalla community My SQL 8.0 e anche da Aurora SQL My versione 3.
- Aurora My SQL version 3 è compatibile con la funzione My SQL hash join della community. L'implementazione specifica di Aurora degli hash join in Aurora My versione 2 non viene utilizzata. SQL Per informazioni sull'utilizzo di hash join con la query parallela Aurora, consultare Abilitazione dell'hash join per cluster di query parallele e Aurora I miei suggerimenti SQL. Per informazioni generali sull'utilizzo degli hash join, consulta Hash Join Optimization nel My Reference Manual. SQL

- La `mysql.lambda_async` stored procedure che era obsoleta in Aurora My SQL versione 2 viene rimossa nella versione 3. Per la versione 3, utilizzare invece la funzione asincrona `lambda_async`.
- Il set di caratteri predefinito in Aurora My SQL versione 3 è. `utf8mb4` In Aurora My SQL versione 2, il set di caratteri predefinito era. `latin1` Per informazioni su questo set di caratteri, consultate Il set di caratteri utf8mb4 (codifica Unicode a 4 byte UTF -8) nel My Reference Manual. SQL

Alcune SQL funzionalità di Aurora My sono disponibili per determinate combinazioni di AWS Region e versione del motore DB. Per dettagli, consultare Funzionalità supportate in Amazon Aurora by e Regione AWS Aurora DB engine.

Supporto delle classi di istanza

Aurora My SQL versione 3 supporta un set diverso di classi di istanze rispetto a Aurora My versione 2: SQL

- Per istanze più grandi, è possibile utilizzare le classi di istanza moderne come`db.r5`,`db.r6g`, `db.x2g`.
- Per istanze più piccole, è possibile utilizzare le classi di istanza moderne come `db.t3` e `db.t4g`.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Le seguenti classi di istanze di Aurora My SQL versione 2 non sono disponibili per Aurora My versione 3: SQL

- `db.r4`
- `db.r3`
- `db.t3.small`
- `db.t2`

Controlla gli script di amministrazione per eventuali CLI istruzioni che creano istanze Aurora SQL My DB. Nomi di classi di istanze hardcode che non sono disponibili per Aurora SQL My versione 3. Se necessario, modifica i nomi delle classi di istanza con quelli supportati da Aurora My SQL versione 3.

> **Tip**
>
> Per controllare le classi di istanza che puoi usare per una combinazione specifica di Aurora My SQL version e AWS Region, usa il `describe-orderable-db-instance-options` AWS CLI comando.

Per i dettagli sulle classi di istanza Aurora, consultare Classi di istanze DB Amazon Aurora.

Modifiche ai parametri per Aurora My versione 3 SQL

Aurora My SQL versione 3 include nuovi parametri di configurazione a livello di cluster e di istanza. Aurora My SQL versione 3 rimuove anche alcuni parametri presenti in Aurora My versione 2. SQL Alcuni nomi dei parametri vengono modificati a seguito dell'iniziativa per un linguaggio inclusivo. Per la compatibilità con le versioni precedenti, è comunque possibile recuperare i valori dei parametri utilizzando i vecchi nomi o i nuovi nomi. Tuttavia, è necessario utilizzare i nuovi nomi per specificare i valori dei parametri in un gruppo di parametri personalizzato.

In Aurora My SQL versione 3, il valore del `lower_case_table_names` parametro viene impostato in modo permanente al momento della creazione del cluster. Se utilizzi un valore non predefinito per questa opzione, configura il gruppo di parametri personalizzati Aurora SQL My versione 3 prima dell'aggiornamento. Quindi specificare il gruppo di parametri durante l'operazione di creazione del cluster o del ripristino dello snapshot.

> **Note**
>
> Con un database globale Aurora basato su Aurora MySQL, non è possibile eseguire un aggiornamento sul posto da Aurora SQL My versione 2 alla versione 3 se il parametro è attivato. `lower_case_table_names` Utilizzare invece il metodo di ripristino snapshot.

In Aurora My SQL versione 3, i `read_only` parametri `init_connect` and non si applicano agli utenti che dispongono del `CONNECTION_ADMIN` privilegio. incluso l'utente master Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.

Per l'elenco completo dei parametri del SQL cluster Aurora My, vedere. Parametri a livello di cluster La tabella riporta tutti i parametri di Aurora My SQL versione 2 e 3. La tabella include note che mostrano quali parametri sono nuovi in Aurora My SQL versione 3 o sono stati rimossi da Aurora My versione 3. SQL

Per l'elenco completo dei parametri dell'SQListanza Aurora My, consulta. Parametri a livello di istanza La tabella riporta tutti i parametri di Aurora My SQL versione 2 e 3. La tabella include note che mostrano quali parametri sono nuovi in Aurora My SQL versione 3 e quali parametri sono stati rimossi da Aurora My versione 3. SQL Include anche note che mostrano quali parametri erano modificabili nelle versioni precedenti ma non in Aurora My versione 3. SQL

Per informazioni sui nomi dei parametri modificati, consultare Modifiche linguistiche incluse per Aurora My versione 3 SQL.

Variabili di stato

Per informazioni sulle variabili di stato che non sono applicabili ad Aurora MySQL, consulta. Le mie variabili di SQL stato che non si applicano ad Aurora My SQL

Modifiche linguistiche incluse per Aurora My versione 3 SQL

Aurora My SQL version 3 è compatibile con la versione 8.0.23 dell'edizione My community. SQL Aurora My SQL versione 3 include anche modifiche rispetto a My SQL 8.0.26 relative alle parole chiave e agli schemi di sistema per un linguaggio inclusivo. Ad esempio, il comando `SHOW REPLICA STATUS` è ora preferito invece di `SHOW SLAVE STATUS`.

Le seguenti CloudWatch metriche di Amazon hanno nuovi nomi in Aurora SQL My versione 3.

In Aurora My SQL versione 3, sono disponibili solo i nuovi nomi delle metriche. Assicurati di aggiornare eventuali allarmi o altre automazioni che si basano sui nomi delle metriche quando esegui l'upgrade ad Aurora My versione 3. SQL

Vecchio nome	Nuovo nome	
`ForwardingMasterDMLLatency`	`ForwardingWriterDMLLatency`	
`ForwardingMasterOpenSessions`	`ForwardingWriterOpenSessions`	
`AuroraDMLRejectedMasterFull`	`AuroraDMLRejectedWriterFull`	
`ForwardingMasterDMLThroughput`	`ForwardingWriterDMLThroughput`	

Le seguenti variabili di stato hanno nuovi nomi in Aurora My SQL versione 3.

Per motivi di compatibilità, puoi usare entrambi i nomi nella versione iniziale di Aurora My SQL versione 3. I nomi delle variabili di stato precedenti devono essere rimossi in una versione futura.

Nome da rimuovere	Nome nuovo o preferito
`Aurora_fwd_master_dml_stmt_duration`	`Aurora_fwd_writer_dml_stmt_duration`
`Aurora_fwd_master_dml_stmt_count`	`Aurora_fwd_writer_dml_stmt_count`
`Aurora_fwd_master_select_stmt_duration`	`Aurora_fwd_writer_select_stmt_duration`
`Aurora_fwd_master_select_stmt_count`	`Aurora_fwd_writer_select_stmt_count`
`Aurora_fwd_master_errors_session_timeout`	`Aurora_fwd_writer_errors_session_timeout`
`Aurora_fwd_master_open_sessions`	`Aurora_fwd_writer_open_sessions`
`Aurora_fwd_master_errors_session_limit`	`Aurora_fwd_writer_errors_session_limit`
`Aurora_fwd_master_errors_rpc_timeout`	`Aurora_fwd_writer_errors_rpc_timeout`

I seguenti parametri di configurazione hanno nuovi nomi in Aurora My SQL versione 3.

Per motivi di compatibilità, è possibile verificare i valori dei parametri nel `mysql` client utilizzando uno dei due nomi nella versione iniziale di Aurora My SQL versione 3. Puoi utilizzare i nuovi nomi solo

quando modifichi i valori in un gruppo di parametri personalizzati. I nomi dei parametri precedenti devono essere rimossi in una versione futura.

Nome da rimuovere	Nome nuovo o preferito
aurora_fwd_master_idle_timeout	aurora_fwd_writer_idle_timeout
aurora_fwd_master_max_connections_pct	aurora_fwd_writer_max_connections_pct
master_verify_checksum	source_verify_checksum
sync_master_info	sync_source_info
init_slave	init_replica
rpl_stop_slave_timeout	rpl_stop_replica_timeout
log_slow_slave_statements	log_slow_replica_statements
slave_max_allowed_packet	replica_max_allowed_packet
slave_compressed_protocol	replica_compressed_protocol
slave_exec_mode	replica_exec_mode
slave_type_conversions	replica_type_conversions
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_parallel_type	replica_parallel_type

Nome da rimuovere	Nome nuovo o preferito
slave_preserve_commit_order	replica_preserve_commit_order
log_slave_updates	log_replica_updates
slave_allow_batching	replica_allow_batching
slave_load_tmpdir	replica_load_tmpdir
slave_net_timeout	replica_net_timeout
sql_slave_skip_counter	sql_replica_skip_counter
slave_skip_errors	replica_skip_errors
slave_checkpoint_period	replica_checkpoint_period
slave_checkpoint_group	replica_checkpoint_group
slave_transaction_retries	replica_transaction_retries
slave_parallel_workers	replica_parallel_workers
slave_pending_jobs_size_max	replica_pending_jobs_size_max
pseudo_slave_mode	pseudo_replica_mode

Le seguenti stored procedure hanno nuovi nomi in Aurora My SQL versione 3.

Per motivi di compatibilità, puoi usare entrambi i nomi nella versione iniziale di Aurora My SQL versione 3. I nomi delle procedure precedenti devono essere rimossi in una versione futura.

Nome da rimuovere	Nome nuovo o preferito
`mysql.rds_set_master_auto_position`	`mysql.rds_set_source_auto_position`
`mysql.rds_set_external_master`	`mysql.rds_set_external_source`
`mysql.rds_set_external_master_with_auto_position`	`mysql.rds_set_external_source_with_auto_position`
`mysql.rds_reset_external_master`	`mysql.rds_reset_external_source`
`mysql.rds_next_master_log`	`mysql.rds_next_source_log`

AUTO_ valori INCREMENT

In Aurora My SQL versione 3, Aurora conserva il `AUTO_INCREMENT` valore di ogni tabella al riavvio di ogni istanza DB. In Aurora My SQL versione 2, il `AUTO_INCREMENT` valore non veniva mantenuto dopo il riavvio.

Il `AUTO_INCREMENT` valore non viene mantenuto quando si configura un nuovo cluster eseguendo il ripristino da un'istantanea, eseguendo un point-in-time ripristino e clonando un cluster. In questi casi, il valore `AUTO_INCREMENT` viene inizializzato sul valore in base al valore di colonna più grande nella tabella al momento della creazione dello snapshot. Questo comportamento è diverso rispetto a My SQL 8.0, in RDS cui il `AUTO_INCREMENT` valore viene mantenuto durante queste operazioni.

Replica dei log binari

In My SQL 8.0 Community Edition, la replica dei log binari è attivata per impostazione predefinita. In Aurora My SQL versione 3, la replica dei log binari è disattivata per impostazione predefinita.

> **Tip**
>
> Se i requisiti di elevata disponibilità sono soddisfatti dalle funzionalità di replica incorporate di Aurora, è possibile lasciare disattivata la replica del log binario. In questo modo, è possibile

evitare il sovraccarico delle prestazioni della replica del log binario. È inoltre possibile evitare il monitoraggio e la risoluzione dei problemi associati necessari per gestire la replica dei log binari.

Aurora supporta la replica dei log binari da una fonte SQL compatibile con My 5.7 ad Aurora My versione 3. SQL Il sistema di origine può essere un cluster Aurora My SQL DB, un'istanza My DB RDS for My SQL DB o un'istanza My locale. SQL

Così come la community MySQL, Aurora My SQL supporta la replica da un'origine che esegue una versione specifica a una destinazione che esegue la stessa versione principale o una versione principale successiva. Ad esempio, la replica da un SQL sistema compatibile con My 5.6 ad Aurora My SQL versione 3 non è supportata. La replica da Aurora SQL My versione 3 a un sistema compatibile con SQL My 5.7 o SQL compatibile con My 5.6 non è supportata. Per i dettagli sull'utilizzo della replica dei log binari, consultare Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Aurora My SQL versione 3 include miglioramenti alla replica dei log binari nella community My SQL 8.0, come la replica filtrata. Per informazioni dettagliate sui miglioramenti apportati alla community da My SQL 8.0, consulta How Servers Evaluate Replication Filtering Rules nel My Reference Manual. SQL

Compressione delle transazioni per la replica dei registri binari

Per informazioni sull'utilizzo della compressione dei log binari, vedere Binary Log Transaction Compression nel My SQL Reference Manual.

Le seguenti limitazioni si applicano alla compressione dei log binari in Aurora My SQL versione 3:

- Le transazioni i cui dati di registro binario sono superiori alla dimensione massima consentita del pacchetto non vengono compresse. Ciò è vero indipendentemente dal fatto che l'impostazione di compressione del registro SQL binario Aurora My sia attivata. Tali transazioni vengono replicate senza essere compresse.
- Se utilizzi un connettore per Change Data Capture (CDC) che non supporta ancora My SQL 8.0, non puoi utilizzare questa funzionalità. Ti consigliamo di testare accuratamente tutti i connettori di terze parti con la compressione del registro binario. Inoltre, ti consigliamo di farlo prima di attivare la compressione binlog su sistemi che utilizzano la replica binlog per. CDC

Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition

È possibile utilizzare le seguenti informazioni per conoscere le modifiche di cui essere a conoscenza quando si converte da un sistema diverso compatibile con MySQL 8.0 a Aurora MySQL versione 3.

In generale, Aurora MySQL versione 3 supporta il set di funzionalità della community MySQL 8.0.23. Alcune nuove funzionalità dell'edizione della community di MySQL 8.0 non si applicano ad Aurora MySQL. Alcune di queste funzionalità non sono compatibili con alcuni aspetti di Aurora, come l'architettura di archiviazione Aurora. Non sono necessarie altre funzionalità perché il servizio di gestione Amazon RDS offre funzionalità equivalenti. Le seguenti funzionalità della community MySQL 8.0 non sono supportate o funzionano in modo diverso in Aurora MySQL versione 3.

Per le note di rilascio di tutte le release di Aurora MySQL versione 3, consultare Aggiornamenti del motore del database per Amazon Aurora MySQL versione 3 nelle Note di rilascio di Aurora MySQL.

Argomenti

- Funzionalità di MySQL 8.0 non disponibili in Aurora MySQL versione 3
- Privilegio basato sui ruoli
- Individuazione dell'ID del server del database
- Autenticazione

Funzionalità di MySQL 8.0 non disponibili in Aurora MySQL versione 3

Le seguenti funzionalità della community MySQL 8.0 non sono disponibili o funzionano in modo diverso in Aurora MySQL versione 3.

- I Resource groups e le istruzioni SQL associate non sono supportati in Aurora MySQL.
- Aurora MySQL non supporta le tablespace di annullamento definite dall'utente e le istruzioni SQL associate, come, e. `CREATE UNDO TABLESPACE` `ALTER UNDO TABLESPACE ... SET INACTIVE` `DROP UNDO TABLESPACE`
- Aurora MySQL non supporta il troncamento undo tablespace per le versioni di Aurora MySQL precedenti alla 3.06. In Aurora MySQL versione 3.06 e successive, è supportato il troncamento automatico dei tablespace di annullamento.
- Non è possibile modificare le impostazioni di plug-in MySQL.
- Il plugin X non è supportato.
- La replica da più origini non è supportata.

Privilegio basato sui ruoli

Con Aurora MySQL versione 3, non è possibile modificare le tabelle direttamente nel database `mysql`. In particolare, non è possibile configurare gli utenti inserendoli nella tabella `mysql.user`. Invece, si utilizzano istruzioni SQL per concedere privilegi basati sui ruoli. Inoltre, nel database `mysql`, non è possibile creare altri tipi di oggetti come le stored procedure. È comunque possibile interrogare le tabelle di `mysql`. Se si utilizza la replica del registro binario, le modifiche apportate direttamente alle tabelle di `mysql` sul cluster fonte non vengono replicate nel cluster di destinazione.

In alcuni casi, l'applicazione potrebbe utilizzare scorciatoie per creare utenti o altri oggetti inserendoli nelle tabelle di `mysql`. In tal caso, modifica il codice dell'applicazione per utilizzare le istruzioni corrispondenti come `CREATE USER`. Se l'applicazione crea stored procedure o altri oggetti nel database `mysql`, utilizzare un database di tipo diverso.

Per esportare i metadati per gli utenti del database durante la migrazione da un database MySQL esterno, è possibile utilizzare un comando MySQL Shell anziché. `mysqldump` Per ulteriori informazioni, vedere Instance Dump Utility, Schema Dump Utility e Table Dump Utility.

Per semplificare la gestione delle autorizzazioni per molti utenti o applicazioni, è possibile utilizzare l'istruzione `CREATE ROLE` per creare un ruolo con una serie di autorizzazioni. Puoi quindi utilizzare le istruzioni `GRANT` e `SET ROLE` e la funzione `current_role` per assegnare ruoli a utenti o applicazioni, cambiare il ruolo corrente e verificare quali ruoli sono in vigore. Per ulteriori informazioni sul sistema di autorizzazione basato sui ruoli in MySQL 8.0, consultare Utilizzo di ruoli nel Manuale di riferimento di MySQL.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

Argomenti

- rds_superuser_role
- Privilege controlla l'utente per la replica dei log binari
- Ruoli per l'accesso ad altri AWS servizi

rds_superuser_role

Aurora MySQL versione 3 include un ruolo speciale che ha tutti i seguenti privilegi. Il ruolo è denominato `rds_superuser_role`. L'utente amministrativo principale per ciascun cluster ha già concesso questo ruolo. Il ruolo `rds_superuser_role` include i seguenti privilegi per tutti gli oggetti del database:

- `ALTER`
- `APPLICATION_PASSWORD_ADMIN`
- `ALTER ROUTINE`
- `CONNECTION_ADMIN`
- `CREATE`
- `CREATE ROLE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`
- `DELETE`
- `DROP`
- `DROP ROLE`
- `EVENT`
- `EXECUTE`
- `FLUSH_OPTIMIZER_COSTS`(Aurora MySQL versione 3.09 e successive)
- `FLUSH_STATUS`(Aurora MySQL versione 3.09 e successive)
- `FLUSH_TABLES`(Aurora MySQL versione 3.09 e successive)
- `FLUSH_USER_RESOURCES`(Aurora MySQL versione 3.09 e successive)
- `INDEX`
- `INSERT`
- `LOCK TABLES`
- `PROCESS`
- `REFERENCES`
- `RELOAD`

- `REPLICATION CLIENT`
- `REPLICATION SLAVE`
- `ROLE_ADMIN`
- `SET_USER_ID`
- `SELECT`
- `SHOW DATABASES`
- `SHOW_ROUTINE` (Aurora MySQL versione 3.04 e successive)
- `SHOW VIEW`
- `TRIGGER`
- `UPDATE`
- `XA_RECOVER_ADMIN`

La definizione del ruolo include anche la `WITH GRANT OPTION` in modo che un utente amministrativo possa concedere tale ruolo ad altri utenti. In particolare, l'amministratore deve concedere tutti i privilegi necessari per eseguire la replica del log binario con il cluster Aurora MySQL come destinazione.

> **Tip**
>
> Per visualizzare i dettagli completi delle autorizzazioni, inserire le seguenti istruzioni.
>
> ```
> SHOW GRANTS FOR rds_superuser_role@'%';
> SHOW GRANTS FOR name_of_administrative_user_for_your_cluster@'%';
> ```

Privilege controlla l'utente per la replica dei log binari

Aurora MySQL versione 3 include un utente che controlla i privilegi per la replica dei log binari (binlog),. `rdsrepladmin_priv_checks_user` Oltre ai privilegi di, questo utente dispone del `rds_superuser_role` privilegio. `replication_applier`

Quando si attiva la replica binlog chiamando la `mysql.rds_start_replication` stored procedure, viene creata. `rdsrepladmin_priv_checks_user`

L'`rdsrepladmin_priv_checks_user@localhost`utente è un utente riservato. Non modificarlo.

Ruoli per l'accesso ad altri AWS servizi

Aurora MySQL versione 3 include ruoli che è possibile utilizzare per accedere ad altri servizi. AWS È possibile impostare molti di questi ruoli come alternativa alla concessione di privilegi. Ad esempio, specificare `GRANT AWS_LAMBDA_ACCESS TO user` anzichéGRANT `INVOKE LAMBDA ON *.* TO user`. Per le procedure di accesso ad altri AWS servizi, vedere. Integrazione di Amazon Aurora SQL My con altri servizi AWS Aurora MySQL versione 3 include i seguenti ruoli relativi all'accesso ad altri servizi: AWS

- `AWS_LAMBDA_ACCESS`— Un'alternativa al privilegio. `INVOKE LAMBDA` Per informazioni sull'utilizzo, consulta Richiamo di una funzione Lambda da un cluster Amazon Aurora My DB SQL.
- `AWS_LOAD_S3_ACCESS`— Un'alternativa al `LOAD FROM S3` privilegio. Per informazioni sull'utilizzo, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.
- `AWS_SELECT_S3_ACCESS`— Un'alternativa al `SELECT INTO S3` privilegio. Per informazioni sull'utilizzo, consulta Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3.
- `AWS_COMPREHEND_ACCESS`— Un'alternativa al `INVOKE COMPREHEND` privilegio. Per informazioni sull'utilizzo, consulta Concessione agli utenti del database dell'accesso a machine learning di Aurora.
- `AWS_SAGEMAKER_ACCESS`— Un'alternativa al `INVOKE SAGEMAKER` privilegio. Per informazioni sull'utilizzo, consulta Concessione agli utenti del database dell'accesso a machine learning di Aurora.
- `AWS_BEDROCK_ACCESS`— Non esiste un `INVOKE` privilegio analogo per Amazon Bedrock. Per informazioni sull'utilizzo, consulta Concessione agli utenti del database dell'accesso a machine learning di Aurora.

Quando si concede l'accesso utilizzando i ruoli in Aurora MySQL versione 3, si attiva anche il ruolo utilizzando l'istruzione `SET ROLE role_name` o `SET ROLE ALL`. L'esempio seguente mostra come. Sostituire il nome del ruolo appropriato per `AWS_SELECT_S3_ACCESS`.

```
# Grant role to user.

mysql> GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'

# Check the current roles for your user. In this case, the AWS_SELECT_S3_ACCESS role
 has not been activated.
```

```
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-----------------------------------------------------+
| CURRENT_ROLE()                                      |
+-----------------------------------------------------+
| `AWS_SELECT_S3_ACCESS`@`%`,`rds_superuser_role`@`%` |
+-----------------------------------------------------+
```

Individuazione dell'ID del server del database

L'ID del server di database (`server_id`) è necessario per la replica con registrazione binaria (binlog). Il modo per trovare l'ID del server è diverso in Aurora MySQL da Community MySQL.

In Community MySQL, l'ID del server è un numero, che si ottiene utilizzando la seguente sintassi mentre si è connessi al server:

```
mysql> select @@server_id;

+-------------+
| @@server_id |
+-------------+
| 2           |
+-------------+
1 row in set (0.00 sec)
```

In Aurora MySQL, l'ID server è l'ID dell'istanza DB, che si ottiene utilizzando la seguente sintassi mentre si accede all'istanza DB:

```
mysql> select @@aurora_server_id;

+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| mydbcluster-instance-2 |
+-------------------------+
1 row in set (0.00 sec)
```

Per ulteriori informazioni sulla replica binlog, vedere. Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari)

Autenticazione

Nella community MySQL 8.0, il plug-in di autenticazione di default è `caching_sha2_password`. Aurora MySQL versione 3 utilizza ancora il plug-in `mysql_native_password`. Non è possibile modificare l'impostazione `default_authentication_plugin`. Tuttavia, è possibile creare nuovi utenti e modificare gli utenti attuali, e le loro password individuali utilizzano il nuovo plug-in di autenticazione. Di seguito è riportato un esempio.

```
mysql> CREATE USER 'testnewsha'@'%' IDENTIFIED WITH caching_sha2_password BY
 'aNewShaPassword';
Query OK, 0 rows affected (0.74 sec)
```

Aggiornamento ad Aurora MySQL versione 3

Per informazioni sull'aggiornamento del database da Aurora MySQL versione 2 alla versione 3, consulta. Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL

Aurora My SQL versione 2 compatibile con My 5.7 SQL

Questo argomento descrive le differenze tra Aurora My SQL versione 2 e My SQL 5.7 Community Edition.

> **Important**
>
> Aurora My SQL version 2 ha raggiunto la fine del supporto standard il 31 ottobre 2024. Per ulteriori informazioni, consulta Preparazione per la fine del supporto standard di Amazon Aurora My SQL -Compatible Edition versione 2.

Funzionalità non supportate in Aurora My versione 2 SQL

Le seguenti funzionalità sono supportate in My SQL 5.7, ma attualmente non sono supportate in Aurora SQL My versione 2:

- `CREATE TABLESPACE`SQLdichiarazione
- Plugin replica gruppi
- Maggiori dimensioni pagina
- Caricamento buffer pool InnoDB all'avvio
- Plugin parser full-text InnoDB
- Replica multi-source
- Ridimensionamento buffer pool online
- Plugin di convalida della password: è possibile installare il plugin, ma non è supportato. Non è possibile personalizzare il plugin.
- Plugin riscrittura query
- Filtri replica
- Protocollo X

Per ulteriori informazioni su queste funzionalità, consulta la documentazione di My SQL 5.7.

Comportamento temporaneo del tablespace in Aurora My versione 2 SQL

In My SQL 5.7, il tablespace temporaneo si estende automaticamente e aumenta di dimensioni in base alle necessità per ospitare le tabelle temporanee su disco. Quando le tabelle temporanee vengono eliminate, lo spazio liberato può essere riutilizzato per nuove tabelle temporanee, ma il tablespace temporaneo rimane nella dimensione estesa e non si riduce. Il tablespace temporaneo viene eliminato e ricreato al riavvio del motore.

In Aurora My SQL versione 2, si applica il seguente comportamento:

- Per i nuovi cluster Aurora My SQL DB creati con la versione 2.10 e successive, il tablespace temporaneo viene rimosso e ricreato al riavvio del database. La funzione di ridimensionamento dinamico può in tal modo recuperare lo spazio di archiviazione.
- Per i cluster Aurora My SQL DB esistenti aggiornati a:

- Versione 2.10 o successive: il tablespace temporaneo viene rimosso e ricreato al riavvio del database. La funzione di ridimensionamento dinamico può in tal modo recuperare lo spazio di archiviazione.
- Versione 2.09: il tablespace temporaneo non viene rimosso al riavvio del database.

È possibile controllare la dimensione del tablespace temporaneo sul cluster DB Aurora My SQL versione 2 utilizzando la seguente query:

```
SELECT
    FILE_NAME,
    TABLESPACE_NAME,
    ROUND((TOTAL_EXTENTS * EXTENT_SIZE) / 1024 / 1024 / 1024, 4) AS SIZE
FROM
    INFORMATION_SCHEMA.FILES
WHERE
    TABLESPACE_NAME = 'innodb_temporary';
```

Per ulteriori informazioni, consulta The Temporary Tablespace nella documentazione personale. SQL

Motore di archiviazione per le tabelle temporanee su disco

Aurora My SQL versione 2 utilizza diversi motori di archiviazione per le tabelle temporanee interne su disco a seconda del ruolo dell'istanza.

- Nell'istanza di scrittura, le tabelle temporanee su disco utilizzano il motore di archiviazione InnoDB per impostazione predefinita. Sono archiviate nel tablespace temporaneo del volume del cluster Aurora.

  È possibile modificare questo comportamento nell'istanza di scrittura modificando il valore del parametro database `internal_tmp_disk_storage_engine`. Per ulteriori informazioni, consulta Parametri a livello di istanza.
- Nelle istanze Reader, le tabelle temporanee su disco utilizzano il motore di ISAM archiviazione My, che utilizza l'archiviazione locale. Questo perché le istanze di sola lettura non possono memorizzare dati sul volume del cluster Aurora.

Sicurezza con Amazon Aurora MySQL

La sicurezza di Amazon Aurora MySQL viene gestita su tre livelli:

- Per controllare chi può eseguire azioni di gestione di Amazon RDS su cluster e istanze DB MySQL Aurora, usi (IAM). AWS Identity and Access Management Quando ti connetti AWS utilizzando le credenziali IAM, il tuo AWS account deve disporre di policy IAM che concedano le autorizzazioni necessarie per eseguire le operazioni di gestione di Amazon RDS. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora

  Se utilizzi IAM per accedere alla console Amazon RDS, assicurati di accedere prima AWS Management Console con le tue credenziali utente IAM. Quindi vai alla console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

- I cluster di database Aurora MySQL devono essere creati in un cloud privato virtuale (VPC) utilizzando il servizio Amazon VPC. Per controllare quali dispositivi e EC2 istanze Amazon possono aprire connessioni all'endpoint e alla porta dell'istanza DB per i cluster Aurora MySQL DB in un VPC, utilizza un gruppo di sicurezza VPC. È possibile creare queste connessioni di endpoint e porta mediante Transport Layer Security (TLS). Le regole del firewall aziendale possono inoltre determinare se i dispositivi in esecuzione nell'azienda possono aprire connessioni a un'istanza database. Amazon VPC e AuroraPer ulteriori informazioni su, consulta. VPCs

  La tenancy VPC supportata dipende dalla classe di istanze database usata dai cluster DB di Aurora MySQL. Con la tenancy VPC `default`, VPC viene eseguito nell'hardware condiviso. Con la tenancy VPC `dedicated`, il VPC viene eseguito in un'istanza hardware dedicata. Le classi di istanze database espandibili supportano solo la locazione VPC di default. Le classi di istanza database con prestazioni espandibili includono le classi di istanza db.t2, db.t3 e db.t4g DB. Tutte le altre classi di istanza database di Aurora MySQL DB supportano sia il tenancy VPC di default che dedicato.

  > **Note**
  >
  > Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Per ulteriori informazioni sulle classi di istanza, consulta Classi di istanze DB Amazon Aurora. Per ulteriori informazioni sulle tenancy VPC `default` e `dedicated`, consulta Istanze dedicate nella Guida per l'utente di Amazon Elastic Compute Cloud.

- Per autenticare l'accesso e le autorizzazioni per un cluster DB Amazon Aurora MySQL puoi seguire uno degli approcci qui riportati oppure utilizzare una loro combinazione:
  - Puoi adottare lo stesso approccio utilizzato per un'istanza standalone di MySQL.

    I comandi come `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` e `SET PASSWORD` funzionano esattamente come nei database in locale, modificando direttamente le tabelle dello schema del database. Per ulteriori informazioni, consulta Access Control and Account Management nella documentazione MySQL.
  - Puoi anche utilizzare l'autenticazione database IAM.

    Questo metodo prevede l'autenticazione del database IAM nel cluster DB tramite un utente IAM oppure con un ruolo IAM e un token di autenticazione. Il token di autenticazione è un valore univoco, generato tramite il processo di firma Signature Version 4. Utilizzando l'autenticazione del database IAM, puoi utilizzare le stesse credenziali per controllare l'accesso alle tue AWS risorse e ai tuoi database. Per ulteriori informazioni, consulta Autenticazione del database IAM .

> **Note**
>
> Per ulteriori informazioni, consulta Sicurezza in Amazon Aurora.

Nelle sezioni seguenti, consulta le informazioni sulle autorizzazioni utente per le connessioni Aurora MySQL e TLS con i cluster Aurora MySQL DB.

Argomenti

- Privilegi dell'utente master con Amazon Aurora MySQL.
- Connessioni TLS ai cluster Aurora MySQL DB

Privilegi dell'utente master con Amazon Aurora MySQL.

Quando crei un'istanza database Amazon Aurora MySQL, l'utente master ha i seguenti privilegi predefiniti indicati in Privilegi dell'account utente master.

Per fornire i servizi di gestione per ogni cluster di database, vengono creati gli utenti `admin` e `rdsadmin` al momento della creazione del cluster di database. I tentativi di rimuovere, rinominare, cambiare la password o modificare i privilegi dell'account `rdsadmin` produrranno errori.

Nei cluster di database Aurora MySQL versione 2, gli utenti `admin` e `rdsadmin` vengono creati al momento della creazione del cluster di database. Nei cluster di database Aurora MySQL versione 3, vengono creati gli utenti `admin`, `rdsadmin` e `rds_superuser_role`.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

Per la gestione del cluster di database di Aurora MySQL, i comandi standard `kill` e `kill_query` sono stati limitati. Al loro posto, utilizza i comandi di Amazon RDS `rds_kill` e `rds_kill_query` per terminare le sessioni utente e le query nelle istanze database di Aurora MySQL.

> **Note**
>
> La crittografia di un'istanza database e delle snapshot non è supportata per la regione Cina (Ningxia).

Connessioni TLS ai cluster Aurora MySQL DB

I cluster database Amazon Aurora MySQL supportano le connessioni Transport Layer Security (TLS) da applicazioni che utilizzano lo stesso processo e la stessa chiave pubblica delle istanze database RDS per MySQL.

Amazon RDS crea un certificato TLS e installa il certificato nell'istanza database quando Amazon RDS effettua il provisioning dell'istanza. Questi certificati sono firmati da un'autorità di certificazione. Il certificato TLS include l'endpoint dell'istanza database come il nome comune (CN) per il certificato TLS per la protezione contro attacchi di spoofing. Di conseguenza, è possibile utilizzare l'endpoint del cluster DB per la connessione a un cluster DB tramite TLS solo se il client supporta i nomi alternativi dell'oggetto (Subject Alternative Names, SAN). In caso contrario, dovrai usare l'endpoint dell'istanza di un'istanza di scrittura.

Per ulteriori informazioni sul download dei certificati, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

Consigliamo il driver AWS JDBC come client che supporta SAN con TLS. Per ulteriori informazioni sul driver AWS JDBC e istruzioni complete per il suo utilizzo, consulta l'archivio dei driver JDBC di Amazon Web Services (AWS). GitHub

Argomenti

- Richiesta di una connessione TLS a un cluster DB Aurora MySQL
- Versioni TLS per Aurora MySQL
- Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL
- Crittografia delle connessioni a un cluster DB Aurora MySQL

Richiesta di una connessione TLS a un cluster DB Aurora MySQL

È possibile richiedere che tutte le connessioni utente al cluster database Aurora MySQL utilizzino TLS utilizzando il parametro cluster database `require_secure_transport`. Per impostazione predefinita, il parametro `require_secure_transport` è impostato su `OFF`. È possibile impostare il parametro `require_secure_transport` su `ON` per richiedere la crittografia TLS per le connessioni al cluster database.

Puoi impostare il valore del parametro `require_secure_transport` aggiornando il gruppo di parametri per il tuo cluster DB. Non è necessario riavviare il cluster DB affinché la modifica abbia effetto. Per ulteriori informazioni sui gruppi di parametri, consulta .

> **Note**
>
> Il parametro `require_secure_transport` è disponibile per Aurora MySQL versione 2 e 3. È possibile impostare questo parametro in un gruppo di parametri del cluster database personalizzato. Il parametro non è disponibile nei gruppi di parametri di istanza database.

Quando il parametro `require_secure_transport` è impostato su `ON` per un cluster DB, un client di database può connettersi ad esso se è in grado di stabilire una connessione crittografata. In caso contrario, viene restituito al client un messaggio di errore simile al seguente:

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

Versioni TLS per Aurora MySQL

Aurora MySQL supporta Transport Layer Security (TLS) versione 1.0, 1.1, 1.2 e 1.3. A partire da Aurora MySQL versione 3.04.0 e successive, è possibile utilizzare il protocollo TLS 1.3 per proteggere le connessioni. La tabella seguente mostra il supporto TLS per le versioni di Aurora MySQL.

Versione Aurora MySQL	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Predefinita
Aurora MySQL versione 2	Supportato	Supportato	Supportato	Non supportato	Tutte le versioni TLS supportate
Aurora MySQL versione 3 (precedente alla 3.04.0)	Supportato	Supportato	Supportato	Non supportato	Tutte le versioni TLS supportate
Aurora MySQL versione 3 (3.04.0 e successive)	Non supportato	Non supportato	Supportato	Supportato	Tutte le versioni TLS supportate

Important

Se utilizzi gruppi di parametri personalizzati per i cluster Aurora MySQL con versione 2 e versioni precedenti alla 3.04.0, ti consigliamo di utilizzare TLS 1.2 perché TLS 1.0 e 1.1 sono meno sicuri. L'edizione community di MySQL 8.0.26 e Aurora MySQL 3.03 e le relative versioni minori hanno reso obsoleto il supporto per le versioni TLS 1.1 e 1.0.

L'edizione community di MySQL 8.0.28 e le versioni compatibili di Aurora MySQL 3.04.0 e successive non supportano TLS 1.1 e TLS 1.0. Se utilizzi Aurora MySQL versione 3.04.0 o successiva, non impostare il protocollo TLS su 1.0 e 1.1 nel gruppo di parametri personalizzato.
Per Aurora MySQL versione 3.04.0 e successive, l'impostazione predefinita è TLS 1.3 e TLS 1.2.

È possibile utilizzare il parametro del cluster database `tls_version` per indicare le versioni del protocollo consentite. Parametri client simili esistono per la maggior parte degli strumenti client o dei driver di database. Alcuni client meno recenti potrebbero non supportare versioni TLS più recenti. Per impostazione predefinita, il cluster DB tenta di utilizzare la versione del protocollo TLS più alta consentita dalla configurazione del server e del client.

Impostare il parametro del cluster `tls_version` DB su uno dei seguenti valori:

- `TLSv1.3`
- `TLSv1.2`
- `TLSv1.1`
- `TLSv1`

È anche possibile impostare il parametro `tls_version` come un stringa di elenco separato da virgole. Se si desidera utilizzare entrambi i protocolli TLS 1.2 e TLS 1.0, il parametro `tls_version` deve includere tutti i protocolli da quello più basso a quello più alto. In questo caso, `tls_version` è impostato come:

```
tls_version=TLSv1,TLSv1.1,TLSv1.2
```

Per informazioni sulla modifica di parametri in un gruppo di parametri del cluster database, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora. Se si utilizza il AWS CLI per modificare il parametro del cluster `tls_version` DB, deve essere impostato su. `ApplyMethod` `pending-reboot` Quando il metodo di applicazione è `pending-reboot`, le modifiche ai parametri vengono applicate dopo l'arresto e il riavvio dei cluster database associati al gruppo di parametri.

Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera abilitare per proteggere le connessioni TLS client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. In tal modo si previene l'uso di crittografie obsolete o non sicure.

Le suite di cifratura configurabili sono supportate in Aurora MySQL versione 3 e Aurora MySQL versione 2. Per specificare l'elenco di cifrature TLS 1.2, TLS 1.1, TLS 1.0 consentite per la crittografia delle connessioni, modifica il parametro del cluster `ssl_cipher`. Imposta il parametro `ssl_cipher` in un gruppo di parametri cluster utilizzando la AWS Management Console, la AWS CLI o l'API RDS.

Imposta il parametro `ssl_cipher` su una stringa di valori di cifratura separati da virgole per la versione TLS. Per l'applicazione client, puoi specificare le cifrature da utilizzare per le connessioni crittografate utilizzando l'opzione `--ssl-cipher` durante la connessione al database. Per ulteriori informazioni sulla connessione al database, consulta Connessione a un cluster di database Amazon Aurora MySQL.

A partire da Aurora MySQL versione 3.04.0 e successive, è possibile specificare le suite di crittografia TLS 1.3. Per specificare le suite di crittografia TLS 1.3 consentite, modifica il parametro `tls_ciphersuites` nel gruppo di parametri. TLS 1.3 ha ridotto il numero di suite di crittografia disponibili a causa delle modifiche alla convenzione di denominazione che rimuove il meccanismo di scambio delle chiavi e il certificato utilizzati. Imposta il parametro `tls_ciphersuites` su una stringa di valori di cifratura separati da virgole per TLS 1.3.

La tabella seguente mostra le cifrature supportate insieme al protocollo di crittografia TLS e alle versioni valide del motore Aurora MySQL per ogni cifratura.

Crittografia	Protocollo di crittografia	Versioni di Aurora MySQL supportate
`ECDHE-RSA-AES128-SHA`	TLS 1.0	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-RSA-AES128-SHA256`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive

Crittografia	Protocollo di crittografia	Versioni di Aurora MySQL supportate
`ECDHE-RSA-AES128-GCM-SHA256`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-RSA-AES256-SHA`	TLS 1.0	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-RSA-AES256-GCM-SHA384`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-ECDSA-AES128-GCM-SHA256`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-ECDSA-AES256-GCM-SHA384`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
`ECDHE-ECDSA-CHACHA20-POLY1305`	TLS 1.2	3.04.0 e versioni successive, 2.11.0 e versioni successive
`TLS_AES_128_GCM_SHA256`	TLS 1.3	3.04.0 e versioni successive
`TLS_AES_256_GCM_SHA384`	TLS 1.3	3.04.0 e versioni successive
`TLS_CHACHA20_POLY1305_SHA256`	TLS 1.3	3.04.0 e versioni successive

Important

`DHE-RSA`i codici non sono supportati dalle versioni di Aurora MySQL a partire dalla 2.11.0. in Aurora MySQL 2 e a partire dalla 3.04.0 in Aurora MySQL 3.

Per informazioni sulla modifica di parametri in un gruppo di parametri del cluster database, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora. Se usi l'interfaccia

a riga di comando per modificare il parametro cluster di database `ssl_cipher`, assicurati di impostare `ApplyMethod` su `pending-reboot`. Quando il metodo di applicazione è `pending-reboot`, le modifiche ai parametri vengono applicate dopo l'arresto e il riavvio dei cluster database associati al gruppo di parametri.

È inoltre possibile utilizzare il comando CLI describe-engine-default-cluster-parameters per determinare quali suite di crittografia sono attualmente supportate per una famiglia di gruppi di parametri specifica. L'esempio seguente mostra come ottenere i valori consentiti per il parametro del cluster `ssl_cipher` per Aurora MySQL versione 2.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
mysql5.7

       ...some output truncated...
 {
  "ParameterName": "ssl_cipher",
  "ParameterValue": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-
AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-
SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-SHA384",
  "Description": "The list of permissible ciphers for connection encryption.",
  "Source": "system",
  "ApplyType": "static",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,ECDHE-
RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-RSA-AES256-
SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-SHA384",
  "IsModifiable": true,
  "SupportedEngineModes": [
   "provisioned"
  ]
 },
       ...some output truncated...
```

Per ulteriori informazioni sulle cifrature, consulta la variabile ssl_ciphers nella documentazione di MySQL. Per ulteriori informazioni sui formati di suite di cifratura, consulta la documentazione per il formato elenco openssl-ciphers e il formato stringa openssl-ciphers sul sito Web OpenSSL.

Crittografia delle connessioni a un cluster DB Aurora MySQL

Per crittografare le connessioni con il client `mysql` di default, avvia il client mysql utilizzando il parametro `--ssl-ca` per fare riferimento alla chiave pubblica, ad esempio:

Per MySQL 5.7 e 8.0:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-mode=VERIFY_IDENTITY
```

Per MySQL 5.6:

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-verify-server-cert
```

*`full_path_to_CA_certificate`*Sostituiscilo con il percorso completo del certificato di Certificate Authority (CA). Per ulteriori informazioni sul download di un certificato, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

È possibile richiedere connessioni TLS per account utente specifici. Ad esempio, in base alla versione di MySQL, è possibile utilizzare una delle seguenti istruzioni per richiedere connessioni TLS per l'account utente `encrypted_user`.

Per MySQL 5.7 e 8.0:

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

Per MySQL 5.6:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

Quando utilizzi un proxy RDS, esegui la connessione all'endpoint del proxy anziché al normale endpoint del cluster. Puoi rendere SSL/TLS obbligatorio o facoltativo per le connessioni al proxy, allo stesso modo delle connessioni dirette al cluster DB Aurora. Per informazioni sull'utilizzo del proxy RDS, consulta Proxy Amazon RDS per Aurora.

> **Note**
>
> Per ulteriori informazioni sulle connessioni TLS con MySQL, consulta la documentazione di MySQL.

Aggiornamento delle applicazioni per la connessione ai cluster database Aurora MySQL utilizzando nuovi certificati TLS

A partire dal 13 gennaio 2023, sono stati pubblicati da Amazon RDS nuovi certificati dell'autorità di certificazione (CA) per la connessione ai cluster database Aurora utilizzando Transport Layer Security (TLS). Di seguito sono disponibili le informazioni sull'aggiornamento delle applicazioni per utilizzare i nuovi certificati.

Le informazioni in questo argomento possono aiutarti a stabilire se le applicazioni client utilizzano TLS per connettersi ai cluster database. In caso affermativo, puoi determinare anche se le applicazioni richiedono la verifica del certificato per la connessione.

> **Note**
>
> Alcune applicazioni sono configurate per connettersi ai cluster DB Aurora MySQL solo se sono in grado di verificare il certificato del server.
> Per queste applicazioni, è necessario aggiornare gli archivi di trust delle applicazioni client per includere i nuovi certificati CA.

Dopo aver aggiornato i certificati CA negli archivi attendibilità delle applicazioni client, puoi ruotare i certificati nei cluster DB. Consigliamo vivamente di testare queste procedure in un ambiente di sviluppo o di gestione temporanea prima di implementarle negli ambienti di produzione.

Per ulteriori informazioni sulla rotazione dei certificati, consulta Rotazione del certificato SSL/TLS. Per ulteriori informazioni sul download, consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Per informazioni sull'utilizzo di TLS con i cluster database Aurora MySQL, consulta Connessioni TLS ai cluster Aurora MySQL DB.

Argomenti

- Determinazione se un'applicazione si connette al cluster databse Aurora MySQL utilizzando SSL
- Determinare se un client richiede la verifica del certificato per la connessione
- Aggiornare l'archivio di trust delle applicazioni
- Codice Java di esempio per stabilire connessioni TLS

Determinazione se un'applicazione si connette al cluster databse Aurora MySQL utilizzando SSL

Se utilizzi Aurora MySQL versione 2 (compatibile con MySQL 5.7) e lo schema delle prestazioni è abilitato, esegui la query indicata di seguito per verificare se le connessioni utilizzano TLS. Per informazioni sull'abilitazione dello schema delle prestazioni, consulta l'argomento relativo alla guida rapida per lo schema delle prestazioni nella documentazione di MySQL.

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

In questo output di esempio, puoi vedere che la tua sessione (`admin`) e un'applicazione collegata come `webapp1` stanno entrambe usando TLS.

```
+----+-----------------+------------------+-----------------+
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

Determinare se un client richiede la verifica del certificato per la connessione

Puoi verificare se i client JDBC e MySQL richiedono la verifica del certificato per la connessione.

JDBC

L'esempio seguente con MySQL Connector/J 8.0 mostra un modo per verificare le proprietà della connessione JDBC di un'applicazione per determinare se le connessioni riuscite richiedono un certificato valido. Per ulteriori informazioni su tutte le opzioni di connessione JDBC per MySQL, consulta l'argomento relativo alle proprietà di configurazione nella documentazione di MySQL.

Quando utilizzi MySQL Connector/J 8.0, la connessione TLS richiede la verifica del certificato CA del server se nelle proprietà di connessione `sslMode` è impostato su `VERIFY_CA` o `VERIFY_IDENTITY`, come nell'esempio seguente.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> Se si utilizza MySQL Java Connector v5.1.38 o versione successiva o MySQL Java Connector v8.0.9 o versione successiva per connettersi ai database, anche se non sono state configurate esplicitamente le applicazioni per l'utilizzo di TLS durante la connessione ai database, questi driver client utilizzano automaticamente TLS. Inoltre, quando utilizzano TLS, eseguono la verifica parziale del certificato e non riescono a connettersi se il certificato del server di database è scaduto.

MySQL

I seguenti esempi con il client MySQL mostrano due modi per verificare la connessione MySQL di uno script per determinare se le connessioni riuscite richiedono un certificato valido. Per ulteriori informazioni su tutte le opzioni di connessione con il client MySQL, consulta l'argomento relativo alla configurazione lato client delle connessioni crittografate nella documentazione di MySQL.

Quando utilizzi il client MySQL 5.7 o MySQL 8.0, la connessione TLS richiede la verifica del certificato CA del server se per l'opzione `--ssl-mode` viene specificato `VERIFY_CA` o `VERIFY_IDENTITY`, come nell'esempio seguente.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

Quando utilizzi il client MySQL 5.6, la connessione SSL richiede la verifica del certificato CA del server se viene specificata l'opzione `--ssl-verify-server-cert`, come nell'esempio seguente.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

Aggiornare l'archivio di trust delle applicazioni

Per informazioni sull'aggiornamento dell'archivio attendibilità per le applicazioni MySQL, consulta l'argomento relativo all'installazione dei certificati SSL nella documentazione di MySQL.

> **Note**
>
> Quando aggiorni l'archivio di trust puoi conservare i certificati meno recenti oltre ad aggiungere i nuovi certificati.

Aggiornare l'archivio di trust delle applicazioni per JDBC

Puoi aggiornare l'archivio di trust delle applicazioni che utilizzano JDBC per le connessioni TLS.

Per ulteriori informazioni sul download del certificato root, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

Per gli script di esempio che importano i certificati, consulta Script di esempio per l'importazione di certificati nel tuo archivio di trust.

Se utilizzi il driver mysql JDBC in un'applicazione, imposta le seguenti proprietà nell'applicazione.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

> **Note**
>
> Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

Quando avvii l'applicazione, imposta le seguenti proprietà.

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

Codice Java di esempio per stabilire connessioni TLS

L'esempio di codice seguente mostra come configurare la connessione SSL che convalida il certificato del server utilizzando JDBC.

```
public class MySQLSSLTest {

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void test(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");


        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.setProperty("sslMode", "VERIFY_IDENTITY");
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);

        Connection connection = DriverManager.getConnection("jdbc:mysql://jagdeeps-ssl-
test.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306",properties);
        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
    }
}
```

Important

Dopo aver stabilito che le connessioni al database utilizzano TLS e aver aggiornato l'archivio attendibile dell'applicazione, è possibile aggiornare il database per utilizzare i certificati rds-ca-rsa 2048-g1. Per istruzioni, consulta la fase 3 in Aggiornamento del certificato CA modificando l'istanza il cluster di database.

Utilizzo dell'autenticazione Kerberos per Aurora My SQL

È possibile utilizzare l'autenticazione Kerberos per autenticare gli utenti quando si connettono al cluster Aurora My DB. SQL A tale scopo, configura il cluster DB da utilizzare per l' AWS Directory Service for Microsoft Active Directory autenticazione Kerberos. AWS Directory Service for Microsoft Active Directory viene anche chiamato. AWS Managed Microsoft ADÈ una funzionalità disponibile con AWS Directory Service. Per ulteriori informazioni, vedi Cos'è AWS Directory Service? nella Guida all'AWS Directory Service amministrazione.

Per iniziare, crea una AWS Managed Microsoft AD directory per memorizzare le credenziali dell'utente. Quindi, fornisci il dominio di Active Directory e altre informazioni al tuo cluster Aurora My SQL DB. Quando gli utenti si autenticano con il cluster Aurora SQL My DB, le richieste di autenticazione vengono inoltrate alla directory. AWS Managed Microsoft AD

Mantenere tutte le credenziali nella stessa directory consente di ridurre il tempo e l'impegno. Con questo approccio, hai una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database. L'uso di una directory può inoltre migliorare il profilo di sicurezza complessivo.

Puoi inoltre accedere alle credenziali da Microsoft Active Directory on-premise. A tale scopo, crea una relazione di dominio trusting in modo che la directory AWS Managed Microsoft AD consideri attendibile Microsoft Active Directory on-premise. In questo modo, gli utenti possono accedere ai cluster Aurora My SQL DB con la stessa esperienza Windows single sign-on (SSO) di quando accedono ai carichi di lavoro nella rete locale.

Un database può utilizzare Kerberos, AWS Identity and Access Management () o entrambi Kerberos e l'autenticazioneIAM. IAM Tuttavia, poiché Kerberos e IAM l'autenticazione offrono metodi di autenticazione diversi, un utente specifico può accedere a un database utilizzando solo l'uno o l'altro metodo di autenticazione, ma non entrambi. Per ulteriori informazioni sull'autenticazione IAM, consulta Autenticazione del database IAM .

Indice

- Panoramica dell'autenticazione Kerberos per i cluster Aurora My DB SQL
- Limitazioni dell'autenticazione Kerberos per Aurora My SQL
- Configurazione dell'autenticazione Kerberos per i cluster Aurora My DB SQL
  - Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD
  - Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise
  - Fase 3: creare un IAM ruolo da utilizzare con Amazon Aurora

- Fase 4: creazione e configurazione di utenti
- Fase 5: Creare o modificare un cluster Aurora My DB SQL
- Passaggio 6: Creare SQL gli utenti Aurora My che utilizzano l'autenticazione Kerberos
  - Modificare un account SQL Aurora My esistente
- Passaggio 7: configurare un client My SQL
- Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole
- Connessione ad Aurora MySQL con l'autenticazione Kerberos
  - Utilizzo dell'accesso Kerberos di Aurora MySQL per la connessione al cluster di database
  - Autenticazione Kerberos con i database globali Aurora
  - Migrazione da RDS per MySQL ad Aurora MySQL
  - Memorizzazione del ticket nella cache impedita
  - Registrazione per l'autenticazione Kerberos
- Gestione di un cluster di database in un dominio
  - Appartenenza al dominio

Panoramica dell'autenticazione Kerberos per i cluster Aurora My DB SQL

Per configurare l'autenticazione Kerberos per un cluster Aurora My SQL DB, completare i seguenti passaggi generali. che vengono descritte dettagliatamente più avanti.

1. Utilizzare per AWS Managed Microsoft AD creare una directory. AWS Managed Microsoft AD È possibile utilizzare il AWS Management Console AWS CLI, il o il AWS Directory Service per creare la directory. Per istruzioni dettagliate, consulta Create your AWS Managed Microsoft AD directory nella AWS Directory Service Administration Guide.
2. Crea un ruolo AWS Identity and Access Management (IAM) che utilizzi la IAM policy gestita`AmazonRDSDirectoryServiceAccess`. Il ruolo consente ad Amazon Aurora di effettuare chiamate alla tua directory.

   Affinché il ruolo consenta l'accesso, l'endpoint AWS Security Token Service (AWS STS) deve essere attivato nel campo Regione AWS per il tuo AWS account. AWS STS Gli endpoint sono tutti Regioni AWS attivi per impostazione predefinita e puoi utilizzarli senza ulteriori azioni. Per ulteriori informazioni, consulta Attivazione e disattivazione AWS STSRegione AWS in un capitolo della Guida per l'utente. IAM

3. Crea e configura gli utenti nella AWS Managed Microsoft AD directory utilizzando gli strumenti di Microsoft Active Directory. Per ulteriori informazioni sulla creazione di utenti in Active Directory, vedere Gestire utenti e gruppi in Microsoft AD AWS gestito nella Guida all'AWS Directory Service amministrazione.
4. Crea o modifica un cluster Aurora My SQL DB. Se utilizzi CLI o RDS API nella richiesta di creazione, specifica un identificatore di dominio con il `Domain` parametro. Utilizzate l'`d-*`identificatore generato quando avete creato la directory e il nome del IAM ruolo che avete creato.

   Se modifichi un cluster Aurora My SQL DB esistente per utilizzare l'autenticazione Kerberos, imposta i parametri di dominio e IAM ruolo per il cluster DB. Individua il cluster DB nella VPC stessa directory del dominio.
5. Utilizza le credenziali utente RDS primarie di Amazon per connetterti al cluster Aurora SQL My DB. Crea l'utente del database in Aurora My SQL utilizzando le istruzioni in. Passaggio 6: Creare SQL gli utenti Aurora My che utilizzano l'autenticazione Kerberos

   Gli utenti creati in questo modo possono accedere al cluster Aurora My SQL DB utilizzando l'autenticazione Kerberos. Per ulteriori informazioni, consulta Connessione ad Aurora MySQL con l'autenticazione Kerberos.

Per usare l'autenticazione Kerberos con una Microsoft Active Directory on-premise o auto ospitato, crea un trust tra foreste. Un trust tra foreste è una relazione di trust tra due gruppi di domini. La fiducia può essere a senso unico o bidirezionale. Per ulteriori informazioni sulla configurazione dei trust forestali utilizzando AWS Directory Service, consulta When to create a trust relationship nella Administration Guide.AWS Directory Service

Limitazioni dell'autenticazione Kerberos per Aurora My SQL

Le seguenti limitazioni si applicano all'autenticazione Kerberos per Aurora My: SQL

- L'autenticazione Kerberos è supportata per Aurora My SQL versione 3.03 e successive.

  Per informazioni sul supporto, consulta Regione AWS . Autenticazione Kerberos con Aurora MySQL
- Per utilizzare l'autenticazione Kerberos con Aurora MySQL, il SQL client o connettore My deve utilizzare la versione 8.0.26 o successiva sulle piattaforme Unix, 8.0.27 o successiva su Windows. In caso contrario, il plugin `authentication_kerberos_client` lato client non è disponibile e non puoi autenticarti.

- AWS Managed Microsoft AD È supportato solo su Aurora My. SQL Tuttavia, puoi unire i cluster Aurora My SQL DB a domini Microsoft AD gestiti condivisi di proprietà di account diversi all'interno dello stesso. Regione AWS

  Inoltre puoi utilizzare la tua Active Directory on-premise. Per ulteriori informazioni, consulta Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise.
- Quando si utilizza Kerberos per autenticare un utente che si connette al SQL cluster Aurora My dai miei SQL client o dai driver sul sistema operativo Windows, per impostazione predefinita la maiuscola e minuscola del nome utente del database deve corrispondere a quella dell'utente in Active Directory. Ad esempio, se l'utente in Active Directory è `Admin`, il nome utente del database deve essere `Admin`.

  Tuttavia, ora puoi utilizzare il confronto dei nomi utente senza distinzione tra maiuscole e minuscole con il plugin `authentication_kerberos`. Per ulteriori informazioni, consulta Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole.
- È necessario riavviare le istanze database di lettura dopo aver attivato la funzionalità per installare il plugin `authentication_kerberos`.
- La replica su istanze database che non supportano il plugin `authentication_kerberos` può causare un errore di replica.
- Affinché i database globali Aurora utilizzino l'autenticazione Kerberos, è necessario configurarla per ogni cluster del database globale.
- Il nome di dominio non deve contenere più di 62 caratteri.
- Non modificare la porta del cluster di database dopo aver abilitato l'autenticazione Kerberos. Se si modifica la porta, l'autenticazione Kerberos non funzionerà.

Configurazione dell'autenticazione Kerberos per i cluster Aurora My DB SQL

AWS Managed Microsoft AD Da utilizzare per configurare l'autenticazione Kerberos per un cluster Aurora My DB. SQL Per configurare l'autenticazione Kerberos, completa la procedura seguente.

Argomenti

- Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD
- Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise

- Fase 3: creare un IAM ruolo da utilizzare con Amazon Aurora
- Fase 4: creazione e configurazione di utenti
- Fase 5: Creare o modificare un cluster Aurora My DB SQL
- Passaggio 6: Creare SQL gli utenti Aurora My che utilizzano l'autenticazione Kerberos
- Passaggio 7: configurare un client My SQL
- Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole

Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD

AWS Directory Service crea una Active Directory completamente gestita nel AWS cloud. Quando crei una AWS Managed Microsoft AD directory, AWS Directory Service crea due controller di dominio e server Domain Name System (DNS) per tuo conto. I server di directory vengono creati in diverse sottoreti in un. VPC Questa ridondanza assicura che la directory rimanga accessibile anche se si verifica un errore.

Quando crei una AWS Managed Microsoft AD directory, AWS Directory Service esegue le seguenti attività per tuo conto:

- Configura un Active Directory all'interno diVPC.
- Crea un account amministratore della directory con il nome utente `Admin` e la password specificata. Puoi utilizzare questo account per gestire le directory.

  > **Note**
  >
  > Assicurati di salvare questa password. AWS Directory Service non la memorizza. È possibile reimpostarla ma non recuperarla.

- Crea un gruppo di sicurezza per i controller della directory.

Quando si avvia un AWS Managed Microsoft AD, AWS crea un'unità organizzativa (OU) che contiene tutti gli oggetti della directory. Questa unità organizzativa ha il BIOS nome di rete immesso al momento della creazione della directory. Si trova nella radice del dominio, che è di proprietà e gestita da AWS.

L'`Admin`account creato con la AWS Managed Microsoft AD directory dispone delle autorizzazioni per le attività amministrative più comuni dell'unità organizzativa, tra cui:

- Creazione, aggiornamento o eliminazione di utenti
- Aggiunta di risorse al tuo dominio, come file o server di stampa, quindi assegnazione delle autorizzazioni per tali risorse a utenti e dell'UO
- Crea contenitori aggiuntivi OUs e
- Delega dell'autorità
- Ripristino degli oggetti eliminati dal cestino di Active Directory
- Esegui i PowerShell moduli AD e DNS Windows sul servizio Web Active Directory

L'account `Admin` dispone anche dei diritti per eseguire queste attività in tutto il dominio:

- Gestione DNS delle configurazioni (aggiunta, rimozione o aggiornamento di record, zone e server d'inoltro)
- Visualizza i registri degli eventi DNS
- visualizzazione di log di eventi di sicurezza.

Per creare una directory con AWS Managed Microsoft AD

1. Accedi a AWS Management Console e apri la AWS Directory Service console all'indirizzo https://console.aws.amazon.com/directoryservicev2/.
2. Nel riquadro di navigazione, seleziona Directories (Directory) e quindi Set up directory (Configura la directory).
3. Scegli AWS Managed Microsoft AD. AWS Managed Microsoft AD è l'unica opzione attualmente utilizzabile con AmazonRDS.
4. Immetti le seguenti informazioni:

   DNSNome della directory

   Il nome completo della directory, ad esempio **`corp.example.com`**.

   BIOSNome della directory Net

   Nome breve per la directory, ad esempio **`CORP`**.

   Descrizione della directory

   (Opzionale) Una descrizione della directory.

Password amministratore

La password dell'amministratore della directory. Con il processo di creazione della directory viene generato un account amministratore con il nome utente Admin e questa password.

La password dell'amministratore della directory e non può includere il termine "admin". La password distingue tra maiuscole e minuscole e la lunghezza deve essere compresa tra 8 e 64 caratteri. Deve anche contenere un carattere di almeno tre delle seguenti quattro categorie:

- Lettere minuscole (a–z)
- Lettere maiuscole (A–Z)
- Numeri (0–9)
- Caratteri non alfanumerici (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password (Conferma password)

La password dell'amministratore immessa nuovamente.

5. Scegli Next (Successivo).
6. Immettere le seguenti informazioni nella sezione Networking (Rete) e quindi scegliere Next (Avanti):

   VPC

   Il VPC per la directory. Crea il cluster Aurora My SQL DB in questo stesso ambiente. VPC

   Sottoreti

   Sottoreti per i server di directory. Le due sottoreti devono trovarsi in diverse zone di disponibilità.
7. Esaminare le informazioni relative alla directory e apportare eventuali modifiche. Quando le informazioni sono corrette, scegli Create Directory (Crea directory).

Per creare la directory sono necessari alcuni minuti. Una volta creata correttamente la directory, il valore Status (Stato) viene modificato in Active (Attivo).

Per consultare le informazioni sulla directory, selezionare il nome della directory nell'elenco di directory. Nota il valore Directory ID perché ti serve quando crei o modifichi il tuo cluster Aurora My SQL DB.

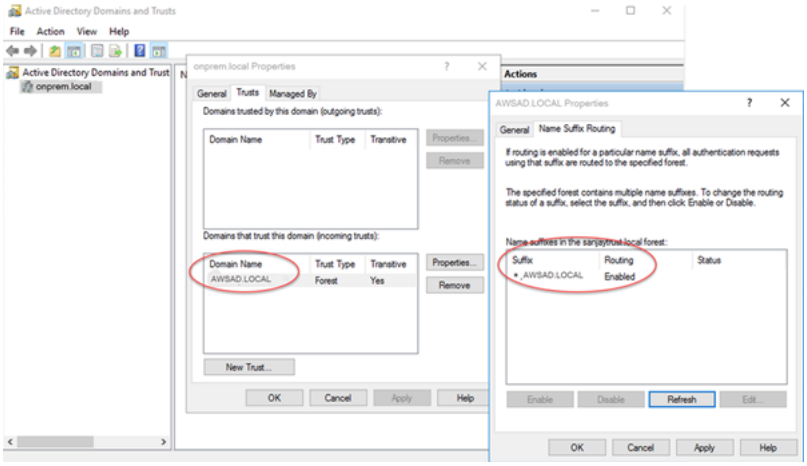
Fase 2: (facoltativa) creazione di un trust per una Active Directory on-premise

Se non prevedi di utilizzare Microsoft Active Directory locale, passa a Fase 3: creare un IAM ruolo da utilizzare con Amazon Aurora.

Per utilizzare l'autenticazione Kerberos con l'Active Directory locale, è necessario creare una relazione di dominio affidabile utilizzando un trust di foresta tra Microsoft Active Directory locale e la AWS Managed Microsoft AD directory (creata in). Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD Il trust può essere unidirezionale, in cui la directory AWS Managed Microsoft AD considera attendibile Microsoft Active Directory locale. Il trust può anche essere bidirezionale, in cui entrambe le Active Directory si considerano reciprocamente attendibili. Per ulteriori informazioni sulla configurazione dei trust utilizzando AWS Directory Service, consulta When to create a trust relationship nella Administration Guide.AWS Directory Service

> **Note**
>
> Se utilizzi una Microsoft Active Directory locale:
>
> - I client Windows non possono connettersi con endpoint personalizzati Aurora. Per ulteriori informazioni, consulta Connessioni endpoint Amazon Aurora.
> - Per i database globali:
>   - I client Windows possono connettersi solo con endpoint di istanza o cluster nella Regione AWS primaria del database globale.
>   - I client Windows non possono connettersi utilizzando gli endpoint del cluster in modalità secondaria. Regioni AWS

Assicurati che il nome di dominio Microsoft Active Directory locale includa un routing dei DNS suffissi che corrisponda alla relazione di trust appena creata. Il risultato è mostrato nella screenshot seguente.



Fase 3: creare un IAM ruolo da utilizzare con Amazon Aurora

Affinché Amazon Aurora possa AWS Directory Service chiamarti, hai bisogno di un ruolo AWS Identity and Access Management (IAM) che utilizzi la policy gestitaIAM. `AmazonRDSDirectoryServiceAccess` Questo ruolo permette ad Aurora di effettuare chiamate ad AWS Directory Service.

Quando crei un cluster DB utilizzando e disponi AWS Management Console dell'`iam:CreateRole`autorizzazione, la console crea questo ruolo automaticamente. In questo caso, il nome del ruolo è `rds-directoryservice-kerberos-access-role`. Altrimenti, è necessario creare il IAM ruolo manualmente. Quando create questo IAM ruolo`Directory Service`, scegliete e allegate `AmazonRDSDirectoryServiceAccess` ad esso la politica AWS gestita.

Per ulteriori informazioni sulla creazione di IAM ruoli per un servizio, consulta Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella Guida per l'IAMutente.

Facoltativamente, puoi creare policy con le autorizzazioni richieste anziché utilizzare la policy IAM gestita `AmazonRDSDirectoryServiceAccess`. In questo caso, il IAM ruolo deve avere la seguente politica di IAM fiducia.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
```

```
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Il ruolo deve anche disporre della seguente policy del ruolo IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Fase 4: creazione e configurazione di utenti

È possibile creare utenti con lo strumento Utenti Active Directory e computer. Questo strumento fa parte degli strumenti Active Directory Domain Services e Active Directory Lightweight Directory Services. Gli utenti possono essere individui singoli o entità che hanno accesso alla tua directory.

Per creare utenti in una AWS Directory Service directory, usi un'EC2istanza locale o Amazon basata su Microsoft Windows che viene aggiunta alla tua AWS Directory Service directory. Devi anche essere connesso all'istanza come utente che dispone dei privilegi per creare utenti. Per ulteriori informazioni, consulta Gestione di utenti e gruppi in AWS Managed Microsoft AD nella AWS Directory Service - Guida di amministrazione.

Fase 5: Creare o modificare un cluster Aurora My DB SQL

Crea o modifica un cluster Aurora My SQL DB da utilizzare con la tua directory. È possibile utilizzare la console o associare un cluster DB RDS API a una directory. AWS CLI Questa operazione può essere eseguita in uno dei seguenti modi:

- Crea un nuovo cluster Aurora My SQL DB utilizzando la console, il create-db-clusterCLIcomando o l'operazione C. reateDBCluster RDS API

  Per istruzioni, consulta Creazione di un cluster database Amazon Aurora.
- Modifica un cluster Aurora My SQL DB esistente utilizzando la console, il modify-db-clusterCLIcomando o l'operazione M. odifyDBCluster RDS API

  Per istruzioni, consulta Modifica di un cluster database Amazon Aurora.
- Ripristina un cluster Aurora My SQL DB da un'istantanea DB utilizzando la console, il comando restore-db-cluster-from-snapshot CLI o l'operazione R. estoreDBCluster FromSnapshot RDS API

  Per istruzioni, consulta Ripristino da uno snapshot cluster database.
- Ripristina un cluster Aurora My SQL DB point-in-time utilizzando la console, il point-in-time CLI comando restore-db-cluster-to- o l'operazione R. estoreDBCluster ToPointInTime RDS API

  Per istruzioni, consulta Ripristino di un cluster di database a un determinato momento.

L'autenticazione Kerberos è supportata solo per i cluster Aurora My SQL DB in a. VPC Il cluster DB può trovarsi nella VPC stessa directory o in un'altra. VPC I cluster DB VPC devono disporre di un gruppo di VPC sicurezza che consenta la comunicazione in uscita verso la directory dell'utente.

Console

Quando utilizzi la console per creare, modificare o ripristinare un cluster di database, scegli Autenticazione Kerberos nella sezione Autenticazione database. Scegli Browse Directory (Sfoglia directory) quindi seleziona la directory oppure scegli Create a new directory (Crea una nuova directory).

AWS CLI

Quando usi AWS CLI o RDSAPI, associa un cluster DB a una directory. Per consentire al cluster di database di utilizzare la directory del dominio che hai creato sono necessari i seguenti parametri:

- Per il parametro `--domain`, utilizza l'identificatore di dominio (identificatore "d-*") generato durante la creazione della directory.
- Per il `--domain-iam-role-name` parametro, utilizzate il ruolo creato che utilizza la IAM policy gestita`AmazonRDSDirectoryServiceAccess`.

Ad esempio, il CLI comando seguente modifica un cluster DB per utilizzare una directory.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --domain d-ID \
    --domain-iam-role-name role-name
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Se modifichi un cluster di database per abilitare l'autenticazione Kerberos, riavvia le istanze database di lettura dopo aver apportato la modifica.

Passaggio 6: Creare SQL gli utenti Aurora My che utilizzano l'autenticazione Kerberos

Il cluster DB viene aggiunto al dominio. AWS Managed Microsoft AD Pertanto, puoi creare utenti Aurora My SQL dagli utenti di Active Directory del tuo dominio. Le autorizzazioni del database sono gestite tramite le autorizzazioni Aurora SQL My standard concesse e revocate a questi utenti.

È possibile consentire a un utente di Active Directory di autenticarsi con SQL Aurora My. A tale scopo, utilizza innanzitutto le credenziali utente RDS primarie di Amazon per connetterti al cluster Aurora SQL My DB come con qualsiasi altro cluster DB. Dopo aver effettuato l'accesso, crea un

utente autenticato esternamente con autenticazione Kerberos in Aurora My come illustrato di seguito: SQL

```
CREATE USER user_name@'host_name' IDENTIFIED WITH 'authentication_kerberos' BY
 'realm_name';
```

- Sostituisci *user_name* con il nome utente. Gli utenti (persone e applicazioni) del tuo dominio ora possono connettersi al cluster di database da un computer client associato al dominio utilizzando l'autenticazione Kerberos.
- Sostituisci *host_name* con il nome host. È possibile utilizzare `%` come carattere jolly. È inoltre possibile usare indirizzi IP specifici per il nome host.
- Sostituisci *realm_name* con il nome dell'area di autenticazione della directory del dominio. Il nome del realm è in genere lo stesso del nome di DNS dominio in lettere maiuscole, ad esempio. `CORP.EXAMPLE.COM` Un realm è un gruppo di sistemi che utilizzano lo stesso Kerberos Key Distribution Center.

L'esempio seguente crea un utente del database con il nome `Admin` che esegue l'autenticazione in Active Directory con il nome del realm `MYSQL.LOCAL`.

```
CREATE USER Admin@'%' IDENTIFIED WITH 'authentication_kerberos' BY 'MYSQL.LOCAL';
```

Modificare un account SQL Aurora My esistente

È inoltre possibile modificare un SQL accesso Aurora My esistente per utilizzare l'autenticazione Kerberos utilizzando la seguente sintassi:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Passaggio 7: configurare un client My SQL

Per configurare un My SQL client, procedi nel seguente modo:

1. Crea un file `krb5.conf` (o equivalente) che punti al dominio.
2. Verifica che il traffico possa fluire tra l'host del client e AWS Directory Service. Utilizza un'utilità di rete come Netcat per le operazioni seguenti:
   - Verifica il traffico verso DNS la porta 53.

- Verifica il traffico suTCP/UDPper la porta 53 e per Kerberos, che include le porte 88 e 464 per. AWS Directory Service

3. Verifica che il traffico scorra senza problemi tra l'host client e l'istanza database sulla porta del database. Ad esempio, utilizza `mysql` per connetterti e accedere al database.

Di seguito è riportato un esempio `krb5.conf` di contenuto per. AWS Managed Microsoft AD

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Di seguito è riportato un esempio di contenuto `krb5.conf` per una Microsoft Active Directory on-premise.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Fase 8: (facoltativo) configurazione il confronto dei nomi utente senza distinzione tra maiuscole e minuscole

Per impostazione predefinita, le maiuscole e minuscole del nome utente My SQL database devono corrispondere a quelle del nome di accesso ad Active Directory. Tuttavia, ora puoi utilizzare il confronto dei nomi utente senza distinzione tra maiuscole e minuscole con il plugin `authentication_kerberos`. A tale scopo, devi impostare il parametro `authentication_kerberos_caseins_cmp` del cluster database su `true`.

Utilizzo del confronto dei nomi utente senza distinzione tra maiuscole e minuscole

1. Crea un gruppo di parametri per il cluster database personalizzato. Seguire le procedure indicate in Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.
2. Modifica il nuovo gruppo di parametri per impostare il valore di `authentication_kerberos_caseins_cmp` su `true`. Seguire le procedure indicate in Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.
3. Associa il gruppo di parametri del cluster DB al tuo cluster Aurora My SQL DB. Seguire le procedure indicate in Associazione di un gruppo di parametri del cluster DB a un cluster DB in Amazon Aurora.
4. Riavviare il cluster database.

Connessione ad Aurora MySQL con l'autenticazione Kerberos

Per evitare errori, usa un client MySQL versione 8.0.26 o successiva su piattaforme Unix e 8.0.27 o successiva su Windows.

Utilizzo dell'accesso Kerberos di Aurora MySQL per la connessione al cluster di database

Per connetterti ad Aurora MySQL con l'autenticazione Kerberos, accedi come utente del database creato utilizzando le istruzioni riportate in Passaggio 6: Creare SQL gli utenti Aurora My che utilizzano l'autenticazione Kerberos.

Al prompt dei comandi, connettiti a uno degli endpoint associati al cluster di database Aurora MySQL. Quando viene richiesta la password, immetti la password Kerberos associata al nome utente.

Quando ti autentichi con Kerberos, viene generato un ticket di concessione (TGT) se non ne esiste già uno. Il plugin `authentication_kerberos` utilizza il TGT per ottenere un ticket di servizio, che viene poi inviato al server di database Aurora MySQL.

Puoi utilizzare il client MySQL per connetterti ad Aurora MySQL con l'autenticazione Kerberos usando Windows o Unix.

Unix

Puoi connetterti usando i seguenti metodi:

- Ottieni il TGT manualmente. In questo caso, non è necessario fornire la password al client MySQL.
- Fornisci la password per l'accesso ad Active Directory direttamente al client MySQL.

Il plugin lato client è supportato nelle piattaforme Unix per i client MySQL versione 8.0.26 e successive.

Per connettersi ottenendo il TGT manualmente

1. Nell'interfaccia della linea di comando, utilizza il seguente comando per ottenere il TGT.

   ```
   kinit user_name
   ```

2. Usa il comando `mysql` seguente per accedere all'endpoint dell'istanza database del cluster di database.

   ```
   mysql -h DB_instance_endpoint -P 3306 -u user_name -p
   ```

   > **Note**
   >
   > L'autenticazione non riesce se il keytab viene ruotato sull'istanza database. In questo caso, ottieni un nuovo TGT eseguendo nuovamente `kinit`.

Per connettersi direttamente

1. Nell'interfaccia della linea di comando usa il comando `mysql` seguente per accedere all'endpoint dell'istanza database del cluster di database.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name -p
```

2. Immetti la password per l'utente di Active Directory.

Windows

In Windows, l'autenticazione viene solitamente eseguita al momento dell'accesso, quindi non è necessario ottenere il TGT manualmente per connettersi al cluster di database Aurora MySQL. Le maiuscole e minuscole del nome utente del database devono corrispondere alle maiuscole e minuscole dell'utente in Active Directory. Ad esempio, se l'utente in Active Directory è `Admin`, il nome utente del database deve essere `Admin`.

Il plug-in lato client è supportato su Windows per i client MySQL versione 8.0.27 e successive.

Per connettersi direttamente

- Nell'interfaccia della linea di comando usa il comando `mysql` seguente per accedere all'endpoint dell'istanza database del cluster di database.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name
```

Autenticazione Kerberos con i database globali Aurora

L'autenticazione Kerberos per Aurora MySQL è supportata per i database globali Aurora. Per autenticare gli utenti sul cluster di database secondario utilizzando l'Active Directory del cluster di database primario, replica l'Active Directory sulla Regione AWS secondaria. L'autenticazione Kerberos viene attivata sul cluster secondario utilizzando lo stesso ID di dominio del cluster primario. La replica AWS Managed Microsoft AD è supportata solo con la versione Enterprise di Active Directory. Per ulteriori informazioni, consulta Replica multiregionale nella Guida per l'amministrazione di AWS Directory Service.

Migrazione da RDS per MySQL ad Aurora MySQL

Dopo aver eseguito la migrazione da RDS per MySQL ad Aurora MySQL con l'autenticazione Kerberos abilitata, modifica gli utenti creati con il plugin `auth_pam` per utilizzare il plugin `authentication_kerberos`. Ad esempio:

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Memorizzazione del ticket nella cache impedita

Se non esiste un TGT valido all'avvio, l'applicazione client MySQL può ottenerlo e memorizzarlo nella cache. Se vuoi evitare che il TGT venga memorizzato nella cache, imposta un parametro di configurazione nel file `/etc/krb5.conf`.

> **Note**
>
> Questa configurazione si applica solo agli host client che eseguono Unix, non Windows.

Per impedire la memorizzazione del TGT nella cache

- Aggiungi una sezione `[appdefaults]` a `/etc/krb5.conf` come segue:

```
[appdefaults]
  mysql = {
    destroy_tickets = true
  }
```

Registrazione per l'autenticazione Kerberos

La variabile di ambiente `AUTHENTICATION_KERBEROS_CLIENT_LOG` imposta il livello di registrazione per l'autenticazione Kerberos. È possibile utilizzare i log per il debug sul lato client.

I valori consentiti sono compresi tra 1 e 5. I messaggi del log vengono scritti nell'output di errore standard. La seguente tabella descrive ciascun livello di registrazione.

Livello di logging	Descrizione
1 o non impostato	Nessuna registrazione
2	Messaggi di errore
3	Messaggi di errore e avviso
4	Messaggi di errore, avviso e informazioni

Livello di logging	Descrizione
5	Messaggi di errore, avviso, informazioni e debug

Gestione di un cluster di database in un dominio

Puoi utilizzare la AWS CLI o l'API RDS per gestire il cluster di database e la sua relazione con la tua Active Directory gestita. Ad esempio, è possibile associare un'autenticazione di Active Directory per Kerberos e annullare l'associazione di una Active Directory per disattivare l'autenticazione Kerberos. Puoi anche spostare un cluster di database affinché venga autenticato esternamente da una Microsoft Active Directory a un'altra.

Ad esempio, puoi utilizzare l'API Amazon RDS per effettuare quanto segue:

- Per tentare di attivare nuovamente l'autenticazione Kerberos per un'appartenenza non riuscita, utilizza l'operazione API `ModifyDBInstance` e specifica l'ID della directory dell'appartenenza corrente.
- Per aggiornare il nome del ruolo IAM dell'appartenenza, utilizza l'operazione API `ModifyDBInstance` e specifica l'ID della directory dell'appartenenza corrente e il nuovo ruolo IAM.
- Per disattivare l'autenticazione Kerberos in un cluster di database, utilizza l'operazione API `ModifyDBInstance` e specifica `none` come parametro di dominio.
- Per spostare un cluster di database da un dominio a un altro, utilizza l'operazione API `ModifyDBInstance` e specifica l'identificatore del nuovo dominio come parametro di dominio.
- Per elencare l'appartenenza per ogni cluster di database, utilizza l'operazione API `DescribeDBInstances`.

Appartenenza al dominio

Quando il cluster di database viene creato o modificato diventa membro del dominio. Puoi visualizzare lo stato dell'appartenenza al dominio per il cluster di database eseguendo il comando CLI describe-db-clusters. Lo stato del cluster di database può essere uno dei seguenti:

- `kerberos-enabled`: il cluster di database ha l'autenticazione Kerberos attivata.
- `enabling-kerberos`: AWS sta attivando l'autenticazione Kerberos sul cluster di database.

- `pending-enable-kerberos`: l'attivazione dell'autenticazione Kerberos è in sospeso su questo cluster di database.
- `pending-maintenance-enable-kerberos`: AWS proverà ad attivare l'autenticazione Kerberos sul cluster di database durante la prossima finestra di manutenzione pianificata.
- `pending-disable-kerberos`: la disattivazione dell'autenticazione Kerberos è in sospeso su questo cluster di database.
- `pending-maintenance-disable-kerberos`: AWS proverà a disattivare l'autenticazione Kerberos sul cluster di database durante la prossima finestra di manutenzione pianificata.
- `enable-kerberos-failed`: un problema di configurazione ha impedito ad AWS di attivare l'autenticazione Kerberos sul cluster di database. Verifica e correggi la configurazione prima di eseguire nuovamente il comando di modifica del cluster di database.
- `disabling-kerberos`: AWS sta disattivando l'autenticazione Kerberos sul cluster di database.

Una richiesta per attivare l'autenticazione Kerberos potrebbe non andare a buon fine a causa di un problema di connettività di rete o un ruolo IAM non corretto. Ad esempio, supponi di creare un cluster di database o di modificare un cluster di database esistente e il tentativo di attivare l'autenticazione Kerberos non riesce. In questo caso, esegui nuovamente il comando di modifica o modifica il cluster di database appena creato per l'aggiunta al dominio.

Migrazione dei dati verso un cluster Amazon Aurora My DB SQL

Sono disponibili diverse opzioni per migrare i dati dal database esistente a un cluster Amazon Aurora SQL My DB. Le opzioni di migrazione dipendono anche dal database da cui esegui la migrazione e dalle dimensioni dei dati sottoposti a migrazione.

Ci sono due diversi tipi di migrazione: fisica e logica. La migrazione fisica significa che le copie fisiche dei file di database vengono utilizzate per migrare il database. La migrazione logica significa che la migrazione viene effettuata applicando modifiche di database logiche, come inserimenti, aggiornamenti ed eliminazioni.

La migrazione fisica ha i vantaggi seguenti:

- La migrazione fisica è più veloce della migrazione logica, specialmente per database di grandi dimensioni.
- Le prestazioni del database non ne risentono quando un backup viene utilizzato per la migrazione fisica.
- La migrazione fisica può migrare tutto nel database origine, comprese componenti di database complesse.

La migrazione fisica ha i limiti seguenti:

- Il parametro `innodb_page_size` deve essere impostato al valore predefinito (`16KB`).
- Il parametro `innodb_data_file_path` deve essere configurato con un solo file di dati che utilizza il nome di file di dati predefinito `"ibdata1:12M:autoextend"`. I database con due file di dati o con un file di dati con un nome diverso non possono essere migrati utilizzando questo metodo.

  Di seguito sono riportati esempi di nomi di file che non sono permessi: `"innodb_data_file_path=ibdata1:50M; ibdata2:50M:autoextend"` e `"innodb_data_file_path=ibdata01:50M:autoextend"`.
- Il parametro `innodb_log_files_in_group` deve essere impostato al valore predefinito (`2`).

La migrazione logica ha i vantaggi seguenti:

- Puoi migrare i sottoinsiemi del database, come tabelle specifiche o parti di una tabella.
- I dati possono essere migrati indipendentemente dalla struttura fisica dello storage.

La migrazione logica ha i limiti seguenti:

- La migrazione logica solitamente è più lenta della migrazione fisica.
- I componenti di database complessi possono rallentare il processo di migrazione logica. In alcuni casi, i componenti di database complessi possono anche bloccare la migrazione logica.

La tabella seguente descrive le opzioni e il tipo di migrazione per ogni opzione.

Migrazione da	Tipo di migrazione	Soluzione
E RDS per l'istanza My DB SQL	Fisica	È possibile migrare da un'istanza RDS for My SQL DB creando prima una replica Aurora SQL My read di un'SQListanza My DB. Quando il ritardo della replica tra l'istanza My SQL DB e la replica Aurora My SQL read è 0, è possibile indirizzare le applicazioni client a leggere dalla replica di lettura Aurora e quindi interrompere la replica per rendere la replica Aurora My read un cluster Aurora My DB autonomo per la lettura e la SQL scrittura. SQL Per informazioni dettagliate, consultare Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora.
E per My DB snapshot RDS SQL	Fisica	Puoi migrare i dati direttamente da uno snapshot RDS for My SQL DB a un cluster Amazon SQL Aurora My DB. Per informazioni dettagliate, consultar e Migrazione di una snapshot RDS for MySQL a Aurora.
A: Il mio SQL database esterno ad Amazon RDS	Logica	Puoi creare un dump dei tuoi dati utilizzando l'`mysqldump` utilità e quindi importare tali dati in un cluster Amazon Aurora SQL My DB esistente . Per informazioni dettagliate, consultar e Migrazione logica da My SQL ad Amazon Aurora My SQL utilizzando mysqldump.

Migrazione da	Tipo di migrazione	Soluzione
		Per esportare i metadati per gli utenti del database durante la migrazione da un SQL database My esterno, puoi anche utilizzare un comando My SQL Shell anziché. `mysqldump` Per ulteriori informazioni, vedere Instance Dump Utility, Schema Dump Utility e Table Dump Utility. **Note** L'utilità mysqlpump è obsoleta a partire da My 8.0.34. SQL
A: Il mio SQL database esterno ad Amazon RDS	Fisica	Puoi copiare i file di backup dal tuo database in un bucket Amazon Simple Storage Service (Amazon S3) e quindi ripristinare un cluster Amazon SQL Aurora My DB da tali file. Questa opzione può essere molto più rapida rispetto alla migrazione dei dati con `mysqldump` . Per informazioni dettaglia te, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3.

Migrazione da	Tipo di migrazione	Soluzione
A: Il mio SQL database esterno ad Amazon RDS	Logica	Puoi salvare i dati dal database come file di testo e copiare tali file in un bucket Amazon S3. È quindi possibile caricare tali dati in un cluster Aurora My SQL DB esistente utilizzando il comando `LOAD DATA FROM S3` MySQL. Per ulteriori informazioni, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.
Un database che non è compatibile con My SQL	Logica	Puoi usare AWS Database Migration Service (AWS DMS) per migrare i dati da un database che non è compatibile con MySQL. Per ulteriori informazioni su AWS DMS, consulta Cos'è il servizio di migrazione AWS del database?

> **Note**
>
> Se stai migrando un SQL database My esterno ad AmazonRDS, le opzioni di migrazione descritte nella tabella sono supportate solo se il database supporta i tablespace InnoDB o ISAM My.
> Se il SQL database Il mio che stai migrando su Aurora SQL My `memcached` utilizza, `memcached` rimuovilo prima di migrarlo.
> Non è possibile migrare ad Aurora SQL My versione 3.05 e successive da alcune versioni precedenti di SQL My 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Ti consigliamo di eseguire l'aggiornamento alla versione My 8.0.28 prima della migrazione. SQL

La migrazione di dati da un database MySQL esterno a un cluster database Amazon Aurora MySQL

Se il database supporta gli spazi tabelle InnoDB o MyISAM, hai queste opzioni per migrare i dati a un cluster database Amazon Aurora MySQL:

- Puoi creare un dump dei dati utilizzando l'utilità `mysqldump` e importare i dati in un cluster di database Amazon Aurora MySQL esistente. Per ulteriori informazioni, consulta Migrazione logica da My SQL ad Amazon Aurora My SQL utilizzando mysqldump.
- Puoi copiare i file di backup completi e incrementali dal database a un bucket Amazon S3 ed eseguire il ripristino su un cluster database Amazon Aurora MySQL da tali file. Questa opzione può essere molto più rapida rispetto alla migrazione dei dati con `mysqldump`. Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3.

Argomenti

- Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3
- Migrazione logica da My SQL ad Amazon Aurora My SQL utilizzando mysqldump

Migrazione fisica da MySQL utilizzando XtraBackup Percona e Amazon S3

Puoi copiare i file di backup completi e incrementali dal database MySQL versione 5.7 o 8.0 di origine a un bucket S3. Puoi quindi eseguire il ripristino in un cluster database Amazon Aurora MySQL con la stessa versione principale del motore d9 datahase da tali file.

Questa opzione può essere molto più veloce rispetto alla migrazione dei dati utilizzando `mysqldump`, in quanto l'utilizzo di `mysqldump` riproduce tutti i comandi per ricreare lo schema e i dati dal database di origine nel cluster di database Aurora MySQL. Copiando i file di dati MySQL origine, Aurora MySQL può immediatamente utilizzare quei file come dati per un cluster di database Aurora MySQL.

Puoi anche ridurre al minimo i tempi di inattività utilizzando la replica dei log binari durante il processo di migrazione. Se utilizzi la replica dei log binari, il database MySQL esterno rimane aperto per transazioni mentre i dati sono in fase di migrazione verso il cluster database Aurora MySQL. Dopo la creazione del cluster di database Aurora MySQL utilizza la replica dei log binari per sincronizzare il cluster di database Aurora MySQL con le transazioni avvenute dopo il backup. Quando il cluster di database Aurora MySQL è aggiornato con il database MySQL, la migrazione viene terminata passando completamente al cluster di database Aurora MySQL per le nuove transazioni. Per ulteriori

informazioni, consulta Sincronizzazione del cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica.

Indice

- Considerazioni e limitazioni
- Prima di iniziare
  - Installazione di Percona XtraBackup
  - Autorizzazioni richieste
  - Creazione del ruolo del servizio IAM
- Esecuzione del backup di file da ripristinare come un cluster database Amazon Aurora MySQL
  - Creazione di un backup completo con Percona XtraBackup
  - Utilizzo di backup incrementali con Percona XtraBackup
  - Considerazioni sul backup
- Ripristino di un cluster di database Amazon Aurora MySQL da un bucket Amazon S3
- Sincronizzazione del cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica
  - Configurazione del database MySQL esterno e del cluster database Aurora MySQL per la replica crittografata
  - Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno
- Riduzione dei tempi di migrazione fisica ad Amazon Aurora MySQL
  - Tipi di tabella non supportati
  - Account utente con privilegi non supportati
  - Privilegi dinamici in Aurora MySQL versione 3
  - Oggetti archiviati con 'rdsadmin'@'localhost' come definer

Considerazioni e limitazioni

Le seguenti limitazioni e considerazioni si applicano al ripristino su un cluster database Amazon Aurora MySQL da un bucket Amazon S3:

- Puoi eseguire la migrazione dei tuoi dati solo a un nuovo cluster database e non a un cluster database esistente.

- È necessario utilizzare Percona per eseguire il backup dei dati XtraBackup su S3. Per ulteriori informazioni, consulta Installazione di Percona XtraBackup.
- Il bucket Amazon S3 e il cluster Aurora MySQL DB devono trovarsi nella stessa regione. AWS
- Non puoi eseguire il ripristino dai seguenti elementi:
  - Esportazione di uno snapshot cluster database in Amazon S3. Non è possibile eseguire la migrazione dei dati da un'esportazione snapshot del cluster database al bucket S3.
  - Un database di origine crittografato, ma puoi crittografare i dati migrati. Durante il processo di migrazione puoi anche lasciare i dati non crittografati.
  - Un database MySQL 5.5 o 5.6
- Percona Server for MySQL non è supportato come database di origine, perché può `compression_dictionary*` contenere tabelle nello schema. `mysql`
- Non è possibile eseguire il ripristino da un cluster database Aurora Serverless.
- La migrazione alle versioni precedenti non è supportata per le versioni principali e secondarie. Ad esempio, non puoi eseguire la migrazione da MySQL versione 8.0 ad Aurora MySQL versione 2 (compatibile con MySQL 5.7), né da MySQL versione 8.0.32 ad Aurora MySQL versione 3.03, che è compatibile con la versione 8.0.26 della community MySQL.
- Non è possibile migrare ad Aurora MySQL 3.05 e versioni successive da alcune versioni precedenti di MySQL 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Si consiglia di eseguire l'aggiornamento a MySQL 8.0.28 prima della migrazione.
- L'importazione da Amazon S3 non è supportata sulla classe di istanza database db.t2.micro. Tuttavia, puoi eseguire il ripristino in una classe istanza database diversa e modificare la classe di istanza in seguito. Per altre informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora.
- Amazon S3 limita la dimensione del file caricato in un bucket S3 a 5 TB. Se un file di backup supera i 5 TB, devi dividerlo in file più piccoli.
- Amazon RDS limita il numero di file caricati in un bucket S3 a 1 milione. Se i dati di backup del database, inclusi tutti i backup completi e incrementali, superano 1 milione di file, utilizza un file Gzip (.gz), tar (.tar.gz) o Percona xbstream (.xbstream) per archiviare i file dei backup completi e incrementali nel bucket S3. Percona XtraBackup 8.0 supporta solo Percona xbstream per la compressione.
- Per fornire servizi di gestione per ogni cluster database, viene creato l'utente `rdsadmin` al momento della creazione del cluster database. Poiché si tratta di un utente riservato in RDS, si applicano le seguenti limitazioni:

  - Le funzioni, le procedure, le viste, gli eventi e i trigger con il definer `'rdsadmin'@'localhost'` non vengono importati. Per ulteriori informazioni, consultare Oggetti archiviati con 'rdsadmin'@'localhost' come definer e Privilegi dell'utente master con Amazon Aurora MySQL..
  - Quando viene creato il cluster database Aurora MySQL, viene creato un utente master con i privilegi massimi supportati. Durante il ripristino dal backup, tutti i privilegi non supportati assegnati agli utenti importati vengono rimossi automaticamente durante l'importazione.

    Per identificare gli utenti che potrebbero essere interessati da questo problema, consulta Account utente con privilegi non supportati. Per ulteriori informazioni sui privilegi supportati in Aurora MySQL, consulta Privilegio basato sui ruoli.
- Per la versione 3 di Aurora MySQL, i privilegi dinamici non vengono importati. I privilegi dinamici supportati da Aurora possono essere importati dopo la migrazione. Per ulteriori informazioni, consulta Privilegi dinamici in Aurora MySQL versione 3.
- Le tabelle create dall'utente nello schema `mysql` non viene migrato.
- Il parametro `innodb_data_file_path` deve essere configurato con un solo file di dati che utilizza il nome di file di dati predefinito `ibdata1:12M:autoextend`. I database con due file di dati o con un file di dati con un nome diverso non possono essere migrati utilizzando questo metodo.

  Di seguito sono riportati esempi di nomi di file non consentiti: `innodb_data_file_path=ibdata1:50M`, `ibdata2:50M:autoextend` e `innodb_data_file_path=ibdata01:50M:autoextend`.
- Non puoi eseguire la migrazione da un database di origine con tabelle definite all'esterno della directory dei dati MySQL predefinita.
- La dimensione massima supportata per i backup non compressi che utilizzano questo metodo è attualmente limitata a 64 TiB. Per i backup compressi, questo limite viene ridotto per tenere conto dei requisiti dello spazio di decompressione. In questi casi, la dimensione massima di backup supportata sarebbe è (`64 TiB - compressed backup size`).
- Aurora MySQL non supporta l'importazione di MySQL e di altri componenti e plugin esterni.
- Aurora MySQL non ripristina tutto dal database. È consigliabile salvare lo schema e i valori del database per i seguenti elementi dal database MySQL di origine e quindi aggiungerli al cluster database Aurora MySQL ripristinato dopo che è stato creato:
  - Account utenti
  - Funzioni
  - Procedure archiviate

- Informazioni fuso orario. Le informazioni sul fuso orario vengono caricate dal sistema operativo locale per il cluster database Aurora MySQL. Per ulteriori informazioni, consulta Fuso orario locale per i cluster DB Amazon Aurora.

Prima di iniziare

Prima di poter copiare i dati in un bucket Amazon S3 ed eseguire il ripristino in un cluster database da tali file, devi eseguire quanto segue:

- Installa Percona sul tuo server locale XtraBackup .
- Permetti a Aurora MySQL di accedere al bucket Amazon S3 per tuo conto.

Installazione di Percona XtraBackup

Amazon Aurora può ripristinare un cluster DB da file creati utilizzando Percona. XtraBackup Puoi installare Percona XtraBackup da Software Downloads - Percona.

Per la migrazione a MySQL 5.7, usa Percona 2.4. XtraBackup

Per la migrazione a MySQL 8.0, usa Percona 8.0. XtraBackup Assicurati che la versione di Percona sia compatibile con la XtraBackup versione del motore del tuo database di origine.

Autorizzazioni richieste

Per migrare i dati MySQL in un cluster database Amazon Aurora MySQL, sono necessarie molte autorizzazioni:

- L'utente che richiede ad Aurora di creare un nuovo cluster da un bucket Amazon S3 deve avere l'autorizzazione per elencare i bucket per il proprio account. AWS Concedi all'utente questa autorizzazione utilizzando una AWS Identity and Access Management policy (IAM).
- Aurora richiede l'autorizzazione di agire per tuo conto per accedere al bucket Amazon S3 dove archivi i file utilizzati per creare il cluster di database Amazon Aurora MySQL. Concedi ad Aurora le autorizzazioni necessarie utilizzando un ruolo di servizio IAM.
- L'utente che effettua la richiesta deve avere anche l'autorizzazione di elencare i ruoli IAM per l'account AWS .
- Se l'utente che effettua la richiesta deve creare un ruolo di servizio IAM o richiedere che Aurora crei un ruolo di servizio IAM (utilizzando la console), l'utente deve avere l'autorizzazione di creare un ruolo IAM per l'account AWS .

- Se prevedi di crittografare i dati durante il processo di migrazione, aggiorna la policy IAM dell'utente che eseguirà la migrazione per concedere l'accesso RDS all'utente AWS KMS keys utilizzato per la crittografia dei backup. Per istruzioni, consultare Creazione di una policy IAM per l'accesso alle risorseAWS KMS.

Ad esempio, la seguente policy IAM concede all'utente le autorizzazioni minime necessarie per utilizzare la console per elencare ruoli IAM, creare un ruolo IAM, elencare i bucket Amazon S3 per l'account ed elencare le chiavi KMS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "s3:ListBucket",
                "kms:ListKeys"
            ],
            "Resource": "*"
        }
    ]
}
```

Inoltre, affinché un utente associ un ruolo IAM a un bucket Amazon S3, l'utente IAM deve avere l'autorizzazione `iam:PassRole` per quel ruolo IAM. Quest'autorizzazione permette a un amministratore di limitare quali ruoli IAM un utente può associare a un bucket Amazon S3.

Ad esempio, la policy IAM permette a un utente di associare il ruolo denominato `S3Access` a un bucket Amazon S3.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
```

```
            "Resource":"arn:aws:iam::123456789012:role/S3Access"
        }
    ]
}
```

Per ulteriori informazioni sulle autorizzazioni utente IAM, consulta Gestione dell'accesso con policy.

Creazione del ruolo del servizio IAM

Puoi AWS Management Console creare un ruolo per te scegliendo l'opzione Crea un nuovo ruolo (illustrata più avanti in questo argomento). Se selezioni questa opzione e specifichi un nome per il nuovo ruolo, Aurora crea il ruolo del servizio IAM necessario affinché Aurora acceda al bucket Amazon S3 con il nome che fornisci.

In alternativa, puoi creare manualmente il ruolo utilizzando la procedura seguente.

Per creare un ruolo IAM affinché Aurora possa accedere a Amazon S3

1. Completa le fasi descritte in Creazione di una policy IAM per l'accesso alle risorse Amazon S3.
2. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Completa le fasi descritte in Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

Esecuzione del backup di file da ripristinare come un cluster database Amazon Aurora MySQL

Puoi creare un backup completo dei file del tuo database MySQL utilizzando XtraBackup Percona e caricare i file di backup su un bucket Amazon S3. In alternativa, se utilizzi già Percona XtraBackup per eseguire il backup dei file del tuo database MySQL, puoi caricare le directory e i file di backup completi e incrementali esistenti su un bucket Amazon S3.

Argomenti

- Creazione di un backup completo con Percona XtraBackup
- Utilizzo di backup incrementali con Percona XtraBackup
- Considerazioni sul backup

Creazione di un backup completo con Percona XtraBackup

Per creare un backup completo dei file del database MySQL che possono essere ripristinati da Amazon S3 per creare un cluster Aurora MySQL DB, utilizza l'utilità Percona () per eseguire il backup del database. XtraBackup `xtrabackup`

Ad esempio, il comando seguente consente di creare un backup di un database MySQL e memorizzare i file nella cartella `/on-premises/s3-restore/backup`.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/
s3-restore/backup>
```

Se desideri comprimere il backup in un singolo file (che può essere diviso, se necessario), puoi utilizzare l'opzione `--stream` per salvare il backup in uno dei seguenti formati:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

Il comando seguente consente di creare un backup del database MySQL diviso in più file Gzip.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar.gz
```

Il comando seguente consente di creare un backup del database MySQL diviso in più file tar.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar
```

Il comando seguente consente di creare un backup del database MySQL diviso in più file xbstream.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.xbstream
```

> **Note**
>
> Se viene visualizzato il seguente errore, potrebbe essere causato dalla combinazione di formati di file nel comando:
>
> ```
> ERROR:/bin/tar: This does not look like a tar archive
> ```

Dopo aver eseguito il backup del database MySQL utilizzando l'utilità XtraBackup Percona, puoi copiare le directory e i file di backup in un bucket Amazon S3.

Per informazioni sulla creazione e il caricamento di un file in un bucket Amazon S3, consulta Nozioni di base su Amazon Simple Storage Service nella Guida alle operazioni di base di Amazon S3.

Utilizzo di backup incrementali con Percona XtraBackup

Amazon Aurora MySQL supporta backup completi e incrementali creati con Percona. XtraBackup Se utilizzi già Percona XtraBackup per eseguire backup completi e incrementali dei file del tuo database MySQL, non è necessario creare un backup completo e caricare i file di backup su Amazon S3. Puoi, invece, risparmiare tempo copiando le directory e i file di backup esistenti per i tuoi backup completi e incrementali in un bucket Amazon S3. Per ulteriori informazioni, consulta Create an incremental backup (Creazione di un backup incrementale) sul sito Web di Percona.

Durante la copia dei file del backup completo e incrementale in un bucket Amazon S3, devi copiare in modo ricorsivo i contenuti della directory di base. Questi contenuti includono il backup completo e anche tutte le directory e i file del backup incrementale. Questa copia deve mantenere la struttura di directory nel bucket Amazon S3. Aurora esegue l'iterazione di tutti i file e le directory. Aurora usa il file `xtrabackup-checkpoints` incluso con ogni backup incrementale per identificare la directory di base e ordinare i backup incrementali in base all'intervallo dei numeri di sequenza log (LSN).

Per informazioni sulla creazione e il caricamento di un file in un bucket Amazon S3, consulta Nozioni di base su Amazon Simple Storage Service nella Guida alle operazioni di base di Amazon S3.

Considerazioni sul backup

Aurora non supporta i backup parziali creati utilizzando Percona. XtraBackup Non puoi usare le opzioni seguenti per creare un backup parziale quando esegui il backup dei file di origine per il database: `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude` o `--databases-file`.

Per ulteriori informazioni sul backup del database con Percona XtraBackup, consulta Percona XtraBackup - Documentation and Work with binary logs sul sito Web di Percona.

Aurora supporta i backup incrementali creati utilizzando Percona. XtraBackup Per ulteriori informazioni, consulta Create an incremental backup (Creazione di un backup incrementale) sul sito Web di Percona.

Aurora consuma i file di backup in base al nome del file. Assicurati di assegnare un nome ai file di backup con l'estensione file appropriata in base al formato file,— ad esempio, `.xbstream` per i file archiviati utilizzando il formato Percona xbstream.

Aurora consuma i file di backup in ordine alfabetico e anche in ordine numerico naturale. Utilizza sempre l'opzione `split` quando invii il comando `xtrabackup` per assicurarti che i file di backup vengano scritti e denominati nell'ordine corretto.

Amazon S3 limita la dimensione del file caricato in un bucket Amazon S3 a 5 TB. Se i dati di backup per il database eccedono 5 TB, utilizza il comando `split` per suddividere i file di backup in file multipli che hanno meno di 5 TB ciascuno.

Aurora limita il numero di file di origine caricati in un bucket Amazon S3 a 1 milione di file. In alcuni casi, i dati di backup per il database, compresi i backup completi e incrementali, possono ammontare a un numero alto di file. In questi casi, utilizza un file tarball (.tar.gz) per archiviare i file di backup completi e incrementali in un bucket Amazon S3.

Quando carichi un file in un bucket Amazon S3, puoi utilizzare la crittografia lato server per crittografare i dati. Puoi ripristinare un cluster database Amazon Aurora MySQL da quei file crittografati. Amazon Aurora MySQL può ripristinare un cluster di database con file crittografati utilizzando i seguenti tipi di crittografia lato server:

- Crittografia lato server con chiavi gestite da Amazon S3–, (SSE-S3): –ogni oggetto viene crittografato con una chiave unica che utilizza una crittografia avanzata a più fattori.
- Crittografia lato server con chiavi AWS KMS gestite (SSE-KMS): simile a SSE-S3, ma è possibile creare e gestire autonomamente le chiavi di crittografia e altre differenze.

Per informazioni sull'utilizzo della crittografia lato server durante il caricamento di file in un bucket Amazon S3, consulta Protezione dei dati con la crittografia lato server nella Guida per sviluppatori Amazon S3.

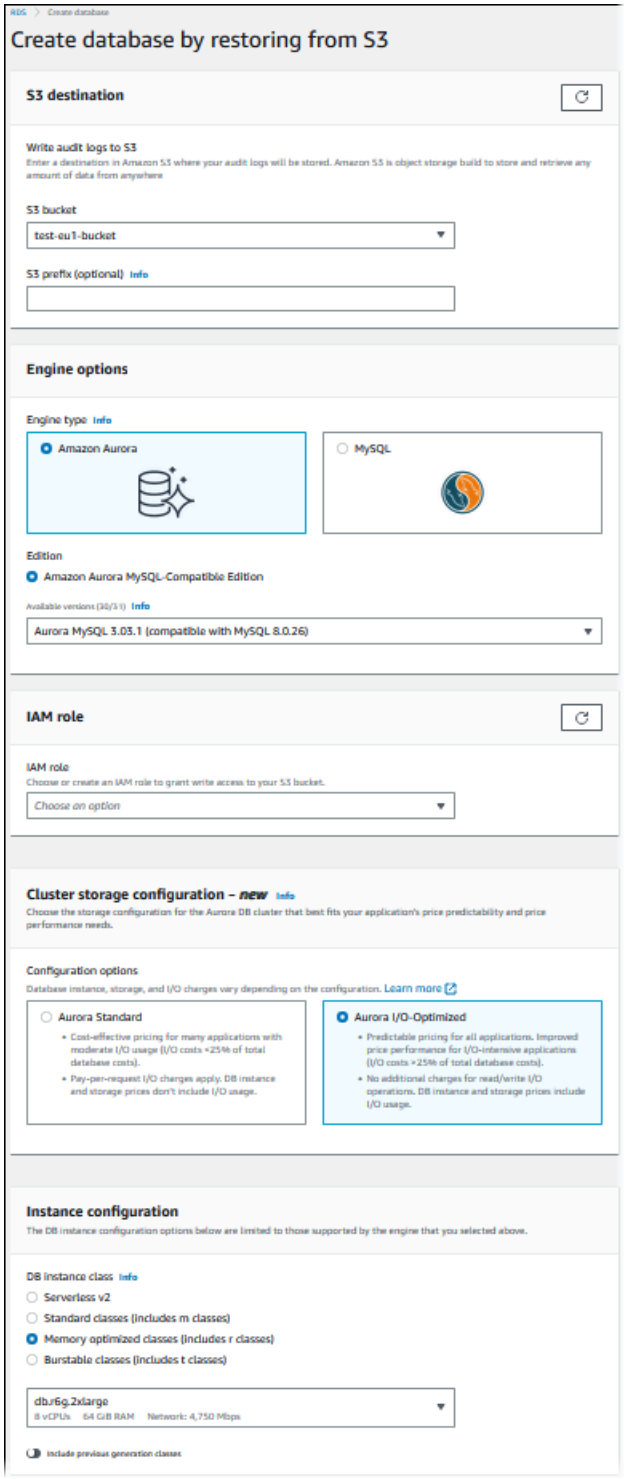
Ripristino di un cluster di database Amazon Aurora MySQL da un bucket Amazon S3

Puoi ripristinare i file di backup dal bucket Amazon S3 per creare un nuovo cluster di database Amazon Aurora MySQL utilizzando la console Amazon RDS.

Per ripristinare un cluster di database Amazon Aurora MySQL da file in un bucket Amazon S3

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra della console Amazon RDS, scegli la AWS regione in cui creare il tuo cluster DB. Scegli la stessa AWS regione del bucket Amazon S3 che contiene il backup del database.
3. Nel riquadro di navigazione, scegliere Databases (Database) e Restore from S3 (Ripristina da S3).
4. Seleziona Ripristina da S3.

   Sarà visualizzata la pagina Crea database ripristinando da S3 .



5. In Destinazione S3:

   a. Selezionare il bucket S3 che contiene i file di backup.

   b. (Facoltativo) Per S3 folder path prefix (Prefisso percorso cartella S3) inserire un prefisso del percorso per i file archiviati nel bucket Amazon S3.

Se non si specifica un prefisso, RDS crea l'istanza database utilizzando tutti i file e le cartelle nella cartella root del bucket S3. Se si specifica un prefisso, RDS crea l'istanza database utilizzando tutti i file e le cartelle nel bucket S3 in cui il percorso del file inizia con il prefisso specificato.

Ad esempio, si supponga di archiviare i file di backup su S3 in una sottocartella denominata backups e di disporre di più set di file di backup, ciascuno nella sua directory (gzip_backup1, gzip_backup2 e così via). In questo caso, specificare un prefisso di backups/gzip_backup1 per eseguire il ripristino dai file nella cartella gzip_backup1.

6. In Opzioni motore:

   a. Per Engine type (Tipo di motore), seleziona Amazon Aurora.

   b. Per Version (Versione), scegliere la versione del motore Aurora MySQL per l'istanza database ripristinata.

7. Per Ruolo IAM, puoi scegliere un ruolo IAM esistente.

8. (Facoltativo) Puoi anche creare un nuovo ruolo IAM selezionando Crea un nuovo ruolo. In tal caso:

   a. Specifica il nome del ruolo IAM.

   b. Specifica se consentire l'accesso alla chiave KMS:

      - Se non hai crittografato i file di backup, seleziona No.
      - Se hai crittografato i file di backup con AES-256 (SSE-S3) quando li hai caricati su Amazon S3, seleziona No. In questo caso, i dati vengono decrittati automaticamente.
      - Se hai crittografato i file di backup con la crittografia lato server AWS KMS (SSE-KMS) quando li hai caricati su Amazon S3, scegli Sì. In seguito, scegli la chiave corretta per AWS KMS key.

        AWS Management Console Crea una policy IAM che consente ad Aurora di decrittografare i dati.

   Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server nella Guida per sviluppatori Amazon S3.

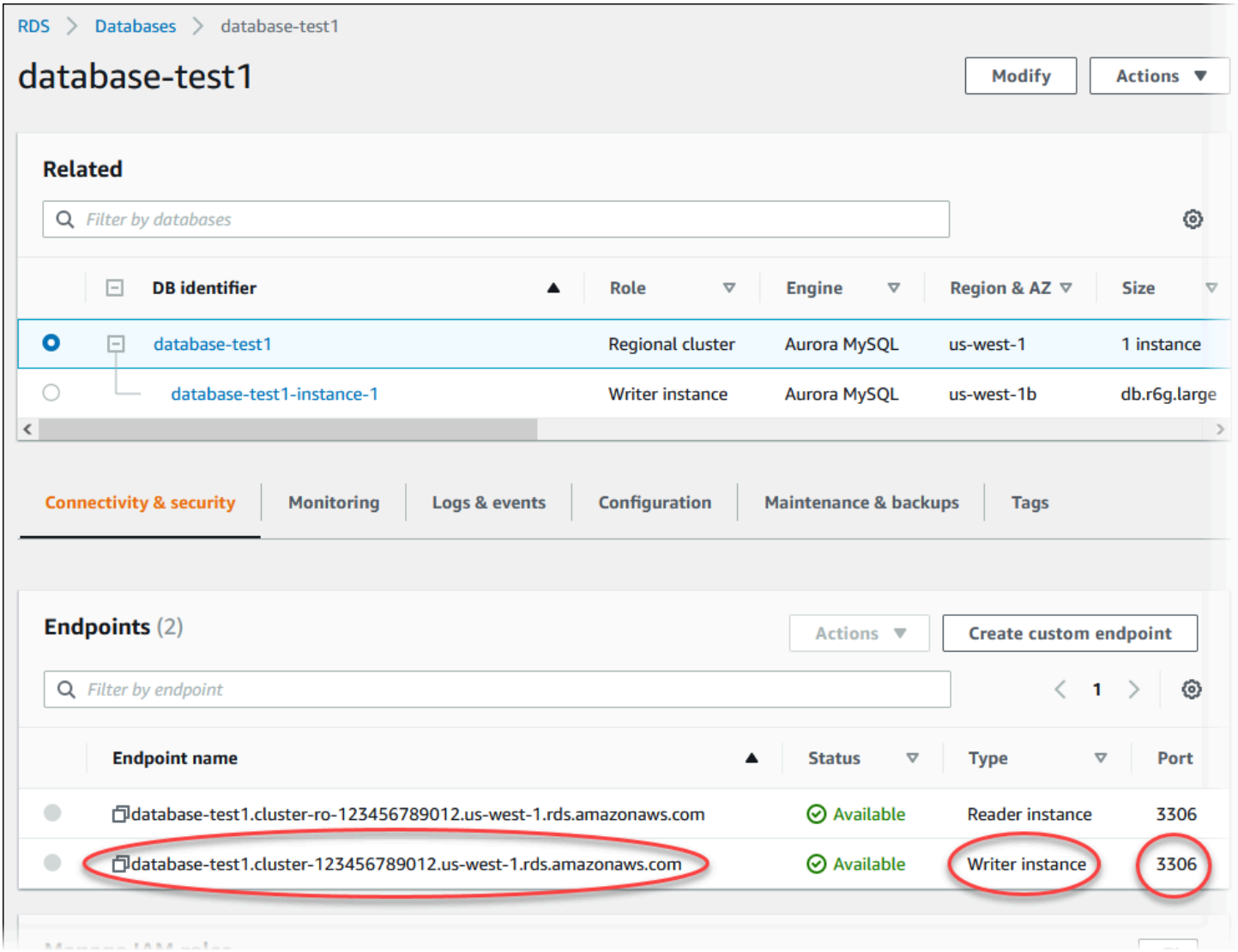
9. Scegliere le impostazioni per il cluster database, ad esempio la configurazione dell'archiviazione del cluster database, la classe di istanza database, l'identificatore del cluster database e le

credenziali di accesso. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

10. Personalizzare le impostazioni aggiuntive per il cluster database Aurora MySQL in base alle esigenze.
11. Scegliere Create database (Crea database) per avviare l'istanza database Aurora.

Nella console Amazon RDS la nuova istanza database viene visualizzata nell'elenco delle istanze database. L'istanza database rimane nello stato creating (creazione in corso) fino a quando non è stata creata e non è pronta per l'uso. Quando lo stato passa su available (disponibile), puoi connetterti all'istanza principale per il cluster database. A seconda della classe di istanza database e dello store allocato, potrebbero trascorrere diversi minuti prima che la nuova istanza sia disponibile.

Per visualizzare il cluster appena creato, seleziona la visualizzazione Databases (Database) nella console Amazon RDS quindi seleziona il cluster di database. Per ulteriori informazioni, consulta Visualizzazione di un cluster di database Amazon Aurora.



Prendi nota della porta e dell'endpoint di scrittura del cluster database. Utilizza la porta e l'endpoint di scrittura del cluster database nelle stringhe di connessione JDBC e ODBC per un'applicazione che esegue operazioni di scrittura o lettura.

Sincronizzazione del cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica

Per fare in modo che ci sia poco o nessun tempo di inattività durante la migrazione, puoi replicare le transazioni eseguite sul database MySQL nel cluster database Aurora MySQL. La replica permette al cluster database di essere aggiornato con le transazioni nel database MySQL avvenute durante la migrazione. Quando il cluster database è completamente aggiornato, puoi fermare la replica e terminare la migrazione su Aurora MySQL.

Argomenti

- Configurazione del database MySQL esterno e del cluster database Aurora MySQL per la replica crittografata
- Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno

Configurazione del database MySQL esterno e del cluster database Aurora MySQL per la replica crittografata

Per replicare i dati in maniera sicura, puoi utilizzare la replica crittografata.

> **Note**
>
> Se non hai bisogno di utilizzare la replica crittografata, puoi saltare queste fasi e passare alle istruzioni in Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno.

Seguono i prerequisiti per l'utilizzo della replica crittografata:

- Secure Sockets Layer (SSL) deve essere abilitato su un database primario MySQL esterno.
- Una chiave e un certificato client devono essere preparati per il cluster database Aurora MySQL.

Durante la replica crittografata, il cluster di database Aurora MySQL agisce come un client per il server di database MySQL. I certificati e le chiavi per il client Aurora MySQL sono in file in formato .pem.

Per configurare il database MySQL esterno e il cluster database Aurora MySQL per la replica crittografata

1. Assicurati di essere preparato per la replica crittografata:
   - Se SSL non è abilitato sul database MySQL primario esterno e non disponi di una chiave client e di un certificato client preparato, abilita SSL sul server di database MySQL e genera la chiave client e il certificato client necessari.
   - Se SSL è abilitato sul primario esterno, fornisci un certificato e una chiave client per il cluster database Aurora MySQL. Se non disponi di questi elementi, genera una nuova chiave e un nuovo certificato per il cluster di database Aurora MySQL. Per firmare il certificato client, devi avere la chiave autorità certificato che hai utilizzato per configurare SSL nel database primario esterno MySQL.

Per ulteriori informazioni, consulta Creating SSL Certificates and Keys Using openssl nella documentazione MySQL.

Hai bisogno del certificato autorità certificato, della chiave client e del certificato client.

2. Esegui la connessione al cluster database Aurora MySQL come primario utilizzando SSL.

   Per informazioni sulla connessione a un cluster di database Aurora MySQL con SSL, consulta Connessioni TLS ai cluster Aurora MySQL DB.

3. Esegui la procedura archiviata mysql.rds_import_binlog_ssl_material per importare le informazioni SSL nel cluster di database Aurora MySQL.

   Per il parametro `ssl_material_value` inserisci le informazioni dai file in formato .pem per il cluster di database Aurora MySQL nel payload JSON corretto.

   L'esempio seguente importa le informazioni SSL in un cluster di database Aurora MySQL. Nei file in formato .pem, il codice del corpo è in genere più lungo del codice del corpo riportato nell'esempio.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Per ulteriori informazioni, consultare mysql.rds_import_binlog_ssl_material e Connessioni TLS ai cluster Aurora MySQL DB.

> **Note**
>
> Dopo aver eseguito la procedura, i segreti vengono archiviati in file. Per eliminare i file in un secondo momento, puoi eseguire la stored procedure mysql.rds_remove_binlog_ssl_material.

Sincronizza il cluster database Amazon Aurora MySQL con il database MySQL esterno

Puoi sincronizzare i cluster database Amazon Aurora MySQL con il database MySQL utilizzando la replica.

Per sincronizzare il cluster database Aurora MySQL con il database MySQL utilizzando la replica

1. Assicurati che il the /etc/my file.cnf per il database MySQL esterno contenga le voci pertinenti.

   Se la replica crittografata non è necessaria, assicurati che il database MySQL esterno venga avviato con i bin binari (binlog) abilitati e SSL disabilitato. Di seguito sono riportate le voci pertinenti nel the /etc/my file.cnf per i dati non crittografati.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

   Se la replica crittografata è necessaria, assicurati che il database MySQL esterno venga avviato con SSL e i binlog abilitati. Le voci nel the /etc/my file.cnf includono le posizioni dei file.pem per il server di database MySQL.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

```
# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

Puoi verificare l'abilitazione di SSL utilizzando il seguente comando.

```
mysql> show variables like 'have_ssl';
```

L'output deve essere simile a quanto segue.

```
+-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-+-~-~-~-~-~-~-~-+
| Variable_name | Value |
+-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-+-~-~-~-~-~-~-~-+
| have_ssl      | YES   |
+-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-+-~-~-~-~-~-~-~-+
1 row in set (0.00 sec)
```

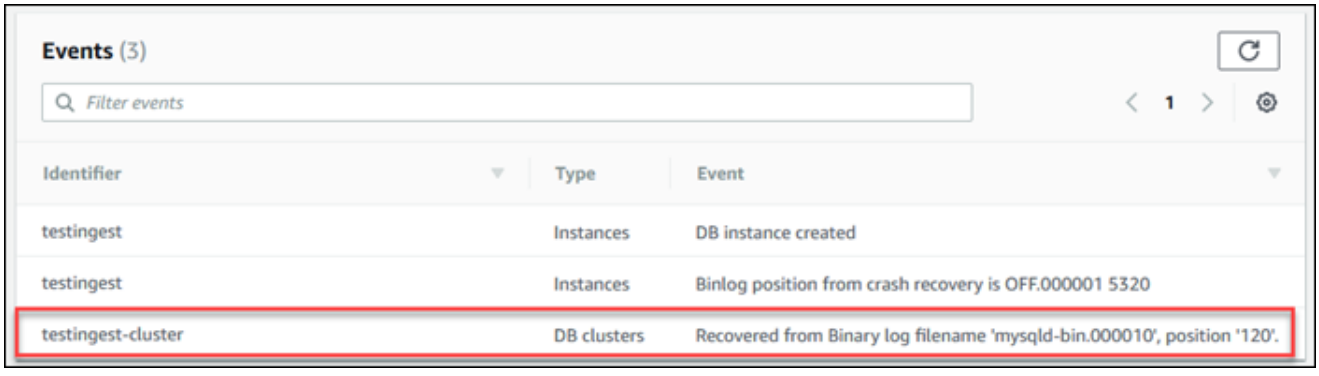
2. Determina la posizione iniziale del log binario per la replica: Specifichi la posizione per avviare la replica in una fase successiva.

   Utilizzando il AWS Management Console

   a. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   b. Nel riquadro di navigazione selezionare Events (Eventi).

   c. Nell'elenco Events (Eventi), annota la posizione nell'evento Recovered from Binary log filename (Recuperato dal nome di file del log binario).



Utilizzando il AWS CLI

È inoltre possibile ottenere il nome e la posizione del file binlog utilizzando il comando AWS CLI describe-events. Di seguito viene illustrato un esempio del comando `describe-events`.

```
PROMPT> aws rds describe-events
```

Nell'output, identifica l'evento che mostra la posizione del binlog.

3. Durante la connessione al database esterno MySQL, crea un utente da utilizzare per la replica. Questo account viene utilizzato unicamente per la replica e deve essere limitato al dominio personale per aumentare la sicurezza. Di seguito è riportato un esempio.

```
mysql> CREATE USER '<user_name>'@'<domain_name>' IDENTIFIED BY '<password>';
```

L'utente richiede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE`. Concedi questi privilegi all'utente.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO
 '<user_name>'@'<domain_name>';
```

Se non hai bisogno di utilizzare la replica crittografata, richiedi le connessioni SSL per l'utente replica. Ad esempio, puoi usare la seguente istruzione per richiedere connessioni SSL per l'account utente *<user_name>*.

```
GRANT USAGE ON *.* TO '<user_name>'@'<domain_name>' REQUIRE SSL;
```

> **Note**
>
> Se `REQUIRE SSL` non è incluso, la connessione di replica potrebbe ridiventare una connessione non crittografata.

4. Nella console Amazon RDS, aggiungi l'indirizzo IP del server che ospita il database MySQL esterno al gruppo di sicurezza VPC per il cluster di database Aurora MySQL. Per ulteriori

informazioni sulla modifica di un gruppo di sicurezza VPC, consulta Gruppi di sicurezza per il VPC nella Guida per l'utente di Amazon Virtual Private Cloud.

Potrebbe anche essere necessario configurare la rete locale per consentire le connessioni dall'indirizzo IP del cluster database Aurora MySQL, affinché possa comunicare con il database MySQL esterno. Per individuare l'indirizzo IP del cluster di database Aurora MySQL, utilizzare il comando `host`.

```
host <db_cluster_endpoint>
```

Il nome host è il nome DNS dell'endpoint del cluster di database Aurora MySQL.

5. Abilita la replica del log binario eseguendo la stored procedure mysql.rds_reset_external_master Aurora My versione 2) SQL SQL o mysql.rds_reset_external_source SQL. Questa procedura archiviata ha la seguente sintassi.

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);

CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Per informazioni sui parametri, consulta mysql.rds_reset_external_master Aurora My versione 2) SQL SQL e mysql.rds_reset_external_source SQL.

Per `mysql_binary_log_file_name` e `mysql_binary_log_file_location`, utilizza la posizione nell'evento Recovered from Binary log filename (Recuperato dal nome di file del log binario) che hai annotato in precedenza.

Se i dati nel cluster di database Aurora MySQL non sono crittografati, il parametro `ssl_encryption` deve essere impostato su `0`. Se i dati sono crittografati, il parametro `ssl_encryption`, deve essere impostato su `1`.

L'esempio seguente esegue la procedura per un cluster database Aurora MySQL che ha dati crittografati.

```
CALL mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);

CALL mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);
```

Questa procedura archiviata imposta i parametri che il cluster database Aurora MySQL utilizza per la connessione al database MySQL esterno e per leggere il log binario. Se i dati sono crittografati, scarica anche il certificato autorità certificato SSL, il certificato client e la chiave client nel disco locale.

6. Avvia la replica del log binario eseguendo la procedura archiviata mysql.rds_start_replication.

```
CALL mysql.rds_start_replication;
```

7. Monitora quanto è indietro il cluster database Aurora MySQL rispetto al database primario MySQL di replica. A questo scopo, esegui la connessione al cluster database Aurora MySQL ed esegui il comando seguente.

```
Aurora MySQL version 2:
SHOW SLAVE STATUS;

Aurora MySQL version 3:
SHOW REPLICA STATUS;
```

   Nell'output del comando, il campo `Seconds Behind Master` indica quanto è indietro il cluster database Aurora MySQL rispetto al primario MySQL. Quando questo valore corrisponde a `0` (zero), il cluster database Aurora MySQL è aggiornato rispetto al primario e puoi passare alla fase successiva per interrompere la replica.

8. Esegui la connessione al database primario MySQL di replica e interrompi la replica. Per eseguire questa operazione, chiama la stored procedure mysql.rds_stop_replication.

```
CALL mysql.rds_stop_replication;
```

Riduzione dei tempi di migrazione fisica ad Amazon Aurora MySQL

Puoi apportare le seguenti modifiche al database per velocizzare il processo di migrazione di un database ad Amazon Aurora MySQL.

> **Important**
>
> Assicurati di eseguire questi aggiornamenti su una copia di un database di produzione, anziché sul database di produzione stesso. Puoi eseguire il backup della copia e quindi ripristinarlo nel cluster database Aurora MySQL per evitare interruzioni del servizio nel database di produzione.

Tipi di tabella non supportati

Aurora MySQL supporta solo il motore InnoDB per le tabelle di database. Se il database include tabelle MyISAM, queste devono essere convertite prima della loro migrazione ad Aurora MySQL. Il

processo di conversione richiede ulteriore spazio per la conversione da MyISAM a InnoDB durante la procedura di migrazione.

Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, converti tutte le tabelle MyISAM in tabelle InnoDB prima di migrarle. Le dimensioni della tabella InnoDB risultante è equivalente alle dimensioni necessarie da Aurora MySQL per quella tabella. Per convertire una tabella MyISAM in InnoDB, esegui il comando seguente:

```
ALTER TABLE schema.table_name engine=innodb, algorithm=copy;
```

Aurora MySQL non supporta tabelle o pagine compresse, ovvero tabelle create con o. `ROW_FORMAT=COMPRESSED` `COMPRESSION = {"zlib"|"lz4"}`

Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, espandi le tabelle compresse impostando `ROW_FORMAT` su `DEFAULT`, `COMPACT`, `DYNAMIC`, o `REDUNDANT`. Per le pagine compresse, imposta. `COMPRESSION="none"`

Per ulteriori informazioni, consulta i formati di riga InnoDB e la compressione di tabelle e pagine InnoDB nella documentazione MySQL.

Puoi utilizzare il seguente script SQL nell'istanza database MySQL esistente per elencare le tabelle nel database che sono tabelle MyISAM o compresse.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Aurora MySQL.
-- It must be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
```

```
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Account utente con privilegi non supportati

Gli account utente con privilegi non supportati da Aurora MySQL vengono importati senza i privilegi non supportati. Per l'elenco dei privilegi supportati, consulta Privilegio basato sui ruoli.

Puoi eseguire la seguente query SQL sul database di origine per elencare gli account utente con privilegi non supportati.

```
SELECT
    user,
    host
FROM
    mysql.user
WHERE
    Shutdown_priv = 'y'
    OR File_priv = 'y'
    OR Super_priv = 'y'
    OR Create_tablespace_priv = 'y';
```

Privilegi dinamici in Aurora MySQL versione 3

I privilegi dinamici non vengono importati. Aurora MySQL versione 3 supporta i privilegi dinamici seguenti.

```
'APPLICATION_PASSWORD_ADMIN',
'CONNECTION_ADMIN',
'REPLICATION_APPLIER',
'ROLE_ADMIN',
'SESSION_VARIABLES_ADMIN',
'SET_USER_ID',
'XA_RECOVER_ADMIN'
```

Lo script di esempio seguente concede i privilegi dinamici supportati agli account utente nel cluster database Aurora MySQL.

```
-- This script finds the user accounts that have Aurora MySQL supported dynamic
 privileges
-- and grants them to corresponding user accounts in the Aurora MySQL DB cluster.

/home/ec2-user/opt/mysql/8.0.26/bin/mysql -uusername -pxxxxx -P8026 -h127.0.0.1 -BNe
 "SELECT
  CONCAT('GRANT ', GRANTS, ' ON *.* TO ', GRANTEE ,';') AS grant_statement
  FROM (select GRANTEE, group_concat(privilege_type) AS GRANTS FROM
 information_schema.user_privileges
      WHERE privilege_type IN (
        'APPLICATION_PASSWORD_ADMIN',
        'CONNECTION_ADMIN',
        'REPLICATION_APPLIER',
        'ROLE_ADMIN',
        'SESSION_VARIABLES_ADMIN',
```

```
        'SET_USER_ID',
        'XA_RECOVER_ADMIN')
      AND GRANTEE NOT IN (\"'mysql.session'@'localhost'\",
\"'mysql.infoschema'@'localhost'\",\"'mysql.sys'@'localhost'\") GROUP BY GRANTEE)
      AS PRIVGRANTS; " | /home/ec2-user/opt/mysql/8.0.26/bin/mysql -u master_username -
p master_password -h DB_cluster_endpoint
```

Oggetti archiviati con 'rdsadmin'@'localhost' come definer

Le funzioni, le procedure, le viste, gli eventi e i trigger con `'rdsadmin'@'localhost'` come definer non vengono importati.

Puoi usare il seguente script SQL nel database MySQL di origine per elencare gli oggetti archiviati con il definire non supportato.

```
-- This SQL query lists routines with `rdsadmin`@`localhost` as the definer.

SELECT
    ROUTINE_SCHEMA,
    ROUTINE_NAME
FROM
    information_schema.routines
WHERE
    definer = 'rdsadmin@localhost';

-- This SQL query lists triggers with `rdsadmin`@`localhost` as the definer.

SELECT
    TRIGGER_SCHEMA,
    TRIGGER_NAME,
    DEFINER
FROM
    information_schema.triggers
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists events with `rdsadmin`@`localhost` as the definer.

SELECT
    EVENT_SCHEMA,
    EVENT_NAME
FROM
    information_schema.events
```

```
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists views with `rdsadmin`@`localhost` as the definer.
SELECT
    TABLE_SCHEMA,
    TABLE_NAME
FROM
    information_schema.views
WHERE
    DEFINER = 'rdsadmin@localhost';
```

Migrazione logica da My SQL ad Amazon Aurora My SQL utilizzando mysqldump

Poiché Amazon Aurora My SQL è un database SQL compatibile con My, puoi utilizzare l'`mysqldump`utilità per copiare i dati dal tuo database My o SQL MariaDB su un cluster Aurora My DB esistente. SQL

Per una discussione su come eseguire questa operazione con SQL database My molto grandi, vedi Importazione di dati in un'istanza DB My SQL o MariaDB con tempi di inattività ridotti. Per SQL i miei database con quantità di dati inferiori, vedere Importazione di dati da un database My SQL o MariaDB a un'istanza My SQL o MariaDB DB.

Migrazione dei dati da un'istanza database RDS per MySQL a un cluster di database Amazon Aurora MySQL

Puoi migrare (copiare) i dati a un cluster di database Amazon Aurora MySQL direttamente da un'istanza database RDS per MySQL.

Argomenti

- Migrazione di una snapshot RDS for MySQL a Aurora
- Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora

> **Note**
>
> Poiché Amazon Aurora MySQL è compatibile con MySQL, puoi migrare i dati dal database MySQL impostando la replica tra il database MySQL e un cluster di database Amazon Aurora MySQL. Per ulteriori informazioni, consulta Replica con Amazon Aurora.

Migrazione di una snapshot RDS for MySQL a Aurora

Puoi effettuare la migrazione di uno snapshot DB di un'istanza database RDS for MySQL per creare un cluster database Aurora MySQL. Il nuovo cluster DB Aurora MySQL viene popolato con i dati dall'istanza database RDS for MySQL originale. Lo snapshot di database deve essere realizzato da un'istanza database Amazon RDS che esegue MySQL compatibile con Aurora MySQL.

Puoi effettuare la migrazione di una snapshot DB manuale o automatizzata. Dopo aver creato il cluster database, puoi le creare le repliche Aurora facoltative.

> **Note**
>
> Puoi anche migrare un'istanza database RDS per MySQL a un cluster di database Aurora MySQL creando una replica di lettura Aurora dell'istanza database RDS per MySQL di origine. Per ulteriori informazioni, consulta Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora.
> Non è possibile migrare ad Aurora MySQL 3.05 e versioni successive da alcune versioni precedenti di MySQL 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Si consiglia di eseguire l'aggiornamento a MySQL 8.0.28 prima della migrazione.

La procedura generale da seguire è riportata di seguito:

1. Determina la quantità di spazio da predisporre per il cluster database Aurora MySQL. Per ulteriori informazioni, consulta Di quanto spazio ho bisogno?
2. Usa la console per creare lo snapshot nella AWS regione in cui si trova l'istanza Amazon RDS MySQL. Per informazioni sulla creazione di uno snapshot DB, consulta Creazione di uno snapshot DB.
3. Se lo snapshot DB non si trova nella stessa AWS regione del cluster DB, usa la console Amazon RDS per copiare lo snapshot DB in quella regione. AWS Per informazioni sulla copia di una snapshot DB, consulta Copia di una snapshot DB.
4. Utilizza la console per migrare la snapshot DB e creare un cluster database Aurora MySQL con gli stessi database dell'istanza database MySQL originale.

> **Warning**
>
> Amazon RDS limita ogni AWS account a una copia istantanea in ciascuna AWS regione alla volta.

Di quanto spazio ho bisogno?

Quando effettui la migrazione di una snapshot di un'istanza database MySQL in un cluster di database Aurora MySQL, Aurora utilizza un volume Amazon Elastic Block Store (Amazon EBS) per formattare i dati dalla snapshot prima della migrazione. In alcuni casi, è necessario ulteriore spazio per formattare i dati per la migrazione.

Le tabelle che non sono tabelle MyISAM e non sono compresse possono avere 16 TB di dimensioni. Se hai tabelle MyISAM, Aurora deve utilizzare ulteriori spazio nel volume per convertire le tabelle in modo che siano compatibili con Aurora MySQL. Se hai tabelle compresse, Aurora deve utilizzare ulteriore spazio nel volume per espandere queste tabelle prima di archiviarle nel volume cluster Aurora. A causa dei requisiti di spazio aggiuntivo, devi accertarti che nessuna delle tabelle MyISAM e compresse migrate dall'istanza database MySQL abbia una dimensione superiore a 8 TB.

Riduzione della quantità di spazio necessaria per migrare i dati in Amazon Aurora MySQL

Potrebbe essere necessario modificare lo schema del database prima di migrarlo in Amazon Aurora. Tale modifica può essere utile nei casi seguenti:

- Desideri accelerare il processo di migrazione.
- Non sai con precisione di quanto spazio necessiti.
- Hai provato a effettuare la migrazione dei dati ma la migrazione non è andata a buon fine per insufficienza di spazio.

Puoi applicare le seguenti modifiche per migliorare il processo di migrazione di un database in Amazon Aurora.

> **Important**
>
> Accertati di eseguire questi aggiornamenti in una nuova istanza database recuperata da una snapshot di un database di produzione, piuttosto che su un'istanza di produzione. Puoi migrare i dati dalla snapshot della nuova istanza database nel cluster database Aurora per evitare interruzioni di servizio nel database di produzione.

Tipo tabella	Limite o linea guida
Tabelle MyISAM	Aurora MySQL supporta solo tabelle InnoDB. Se hai tabelle MyISAM nel database, devono essere convertite prima di essere migrate in Aurora MySQL. Il processo di conversione richiede ulteriore spazio per la conversione da MyISAM a InnoDB durante la procedura di migrazione. Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, converti tutte le tabelle MyISAM in tabelle InnoDB prima di migrarle. Le dimensioni della tabella InnoDB risultante è equivalente alle dimensioni necessarie da Aurora MySQL per quella tabella. Per convertire una tabella MyISAM in InnoDB, esegui il comando seguente: `alter table <schema>.<table_name> engine=innodb, algorithm=copy;`
Tabelle compresse	Aurora MySQL non supporta tabelle compresse (ovvero tabelle create con `ROW_FORMAT=COMPRESSED`).

Tipo tabella	Limite o linea guida
	Per ridurre la possibilità di rimanere senza spazio o per velocizzare il processo di migrazione, espandi le tabelle compresse impostando `ROW_FORMAT` su `DEFAULT`, `COMPACT`, `DYNAMIC`, o `REDUNDANT`. Per ulteriori informazioni, consulta Formati di riga in InnoDB nella documentazione di MySQL.

Puoi utilizzare il seguente script SQL nell'istanza database MySQL esistente per elencare le tabelle nel database che sono tabelle MyISAM o compresse.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Amazon Aurora.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
```

```
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Lo script produce output simile all'output nell'esempio seguente. L'esempio visualizza due tabelle che devono essere convertite da MyISAM a InnoDB. L'output include anche le dimensioni approssimative di ogni tabella in megabyte (MB).

```
+---------------------------------+------------------+
| ==> MyISAM or Compressed Tables | Approx size (MB) |
+---------------------------------+------------------+
| test.name_table                 |          2102.25 |
| test.my_table                   |            65.25 |
+---------------------------------+------------------+
2 rows in set (0.01 sec)
```

Migrazione di uno snapshot DB di RDS per MySQL in un cluster database Aurora MySQL

È possibile migrare uno snapshot DB di un'istanza DB RDS for MySQL per creare un cluster Aurora MySQL DB utilizzando o il. AWS Management Console AWS CLI Il nuovo cluster DB Aurora MySQL viene popolato con i dati dall'istanza database RDS for MySQL originale. Per informazioni sulla creazione di uno snapshot DB, consulta Creazione di uno snapshot DB.

Se lo snapshot DB non si trova nella AWS regione in cui desideri collocare i dati, copia lo snapshot DB in quella regione. AWS Per informazioni sulla copia di una snapshot DB, consulta Copia di una snapshot DB.

Console

Quando migri lo snapshot DB utilizzando AWS Management Console, la console esegue le azioni necessarie per creare solo il cluster DB.

Puoi decidere anche che il nuovo cluster DB Aurora MySQL venga crittografato a riposo utilizzando una AWS KMS key.

Per migrare uno snapshot MySQL DB utilizzando il AWS Management Console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Inizia la migrazione dall'istanza database MySQL o dalla snapshot:

   Per iniziare la migrazione dall'istanza database:

   1. Nel riquadro di navigazione, seleziona Databases (Database), quindi seleziona l'istanza database MySQL.
   2. In Actions (Operazioni) scegliere Migrate latest snapshot (Migrazione snapshot più recente).

   Per iniziare la migrazione dalla snapshot:

   1. SelezionareSnapshots (Snapshot).
   2. Nella pagina Snapshots (Snapshot) seleziona la snapshot che desideri migrare in un cluster di database Aurora MySQL.
   3. Scegliere Snapshot Actions (Operazioni snapshot), quindi scegliere Migrate Snapshot (Migrazione snapshot).

   Viene visualizzata la pagina Migrate Database (Migrazione database).
3. Impostare i valori seguenti nella pagina Migrate Database (Migra database):

   - Migrate to DB Engine (Migrazione al motore DB): seleziona `aurora`.
   - DB Engine Version (Versione motore database): seleziona la versione di motore database per il cluster di database Aurora MySQL.

- Classe di istanza database: scegliere una classe dell'istanza database che abbia lo spazio di archiviazione e la capacità richiesti per il database, ad esempio `db.r3.large`. I volumi dei cluster Aurora aumentano automaticamente quando aumenta la quantità di dati nel database. Un volume del cluster Aurora può raggiungere una dimensione massima di 128 tebibytes (TiB). È quindi sufficiente selezionare una classe di istanza database che soddisfi i requisiti di storage correnti. Per ulteriori informazioni, consulta Panoramica dell'archiviazione di Amazon Aurora.
- Identificatore dell'istanza DB: digita un nome per il cluster DB che sia univoco per il tuo account nella AWS regione selezionata. Questo identificatore viene utilizzato negli indirizzi degli endpoint per le istanze nel cluster di database. Potresti scegliere di aggiungere alcune informazioni al nome, ad esempio includendo la AWS regione e il motore DB selezionati. **`aurora-cluster1`**

  L'identificatore istanze database presenta i seguenti vincoli:

  - Deve contenere da 1 a 63 caratteri alfanumerici o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutte le istanze DB per AWS account e per AWS regione.
- Virtual Private Cloud (VPC): se hai già un VPC esistente, puoi utilizzarlo con il cluster di database Aurora MySQL selezionando l'identificatore VPC, ad esempio `vpc-a464d1c1`. Per ulteriori informazioni sulla creazione di un VPC, consulta Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 .

  Diversamente, puoi stabilire che Aurora crei un VPC per te selezionando Create a new VPC (Crea un nuovo VPC).
- DB subnet group (Gruppo di sottoreti DB): un gruppo di sottoreti esistente può essere utilizzato con il cluster database Aurora MySQL selezionando l'identificatore del gruppo di sottoreti, ad esempio `gs-subnet-group1`.

  Diversamente, puoi stabilire che Aurora crei un gruppo di sottoreti selezionando Create a new subnet group (Crea nuovo gruppo di sottoreti).
- Public accessibility (Accesso pubblico): seleziona No (No) per specificare che le istanze nel cluster database si possono accedere solamente da risorse all'interno del proprio VPC. Selezionare Yes (Si) per specificare che le istanze nel cluster database sono accessibili dalle risorse nella rete pubblica. Il valore predefinito è Yes (Sì).

> **Note**
>
> Il cluster DB di produzione potrebbe non trovarsi in una sottorete pubblica, poiché solo i server applicativi richiedono l'accesso al cluster DB. Se non è necessario che il cluster database si trovi in una sottorete pubblica, impostare Publicly Accessible (Accessibile pubblicamente) su No.

- Availability Zone (Zona di disponibilità): seleziona la zona di disponibilità per ospitare l'istanza primaria per il cluster di database Aurora MySQL. Per permettere a Aurora di selezionare automaticamente una zona di disponibilità, seleziona No Preference (Nessuna preferenza).
- Database Port (Porta database): digita la porta predefinita da usare quando si effettua la connessione a istanze nel cluster di database Aurora MySQL. Il valore di default è `3306`.

> **Note**
>
> Potrebbe essere attivo un firewall aziendale che non permette l'accesso alle porte predefinite come la porta predefinita di MySQL 3306. In questo caso, fornire un valore di porta permesso dal firewall aziendale. Ricorda il valore di porta per utilizzarlo successivamente, al momento della connessione al cluster di database Aurora MySQL.

- Encryption (Crittografia): seleziona Enable Encryption (Abilita crittografia) perché il nuovo cluster di database Aurora MySQL venga crittografato mentre è inattivo. Se scegli Enable Encryption (Abilita crittografia), devi scegliere una chiave KMS come valore AWS KMS key.

  Se la snapshot DB non è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo.

  Se la snapshot DB è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo utilizzando la chiave di crittografia specificata. Puoi specificare la chiave di crittografia utilizzata dalla snapshot DB o una chiave diversa. Non puoi creare un cluster database non crittografato da una snapshot DB crittografata.
- Aggiornamento automatico della versione secondaria: questa impostazione non si applica ai cluster di database Aurora MySQL.

Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consultare Aggiornamenti del motore di database per Amazon Aurora My SQL.

4. Selezionare Migrate (Migra) per effettuare la migrazione della snapshot DB.
5. Selezionare Instances (Istanze) e quindi l'icona con la freccia per visualizzare i dettagli del cluster DB e monitorare l'avanzamento della migrazione. Nella pagina dei dettagli, puoi trovare l'endpoint del cluster utilizzato per la connessione all'istanza principale del cluster DB. Per ulteriori informazioni sulla connessione a un cluster di database Aurora MySQL, consulta Connessione a un cluster database Amazon Aurora.

AWS CLI

Puoi creare un cluster database Aurora da uno snapshot DB di una istanza database RDS for MySQL utilizzando il comando `restore-db-cluster-from-snapshot` con i seguenti parametri:

- `--db-cluster-identifier`: nome del cluster di database da creare.
- `--engine aurora-mysql`: per un cluster di database compatibile con MySQL 5.7 o 8.0.
- `--kms-key-id`— Con cui AWS KMS key crittografare facoltativamente il cluster DB, a seconda che lo snapshot del DB sia crittografato.
  - Se la snapshot DB non è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo. Altrimenti, il cluster database non è crittografato.
  - Se la snapshot DB è crittografata, specifica una chiave di crittografia per fare in modo che il cluster database sia crittografato mentre è inattivo utilizzando la chiave di crittografia specificata. Altrimenti, il cluster database è crittografato mentre è inattivo utilizzando la chiave di crittografia per la snapshot DB.

  > **Note**
  >
  > Non puoi creare un cluster database non crittografato da una snapshot DB crittografata.

- `--snapshot-identifier`: l'Amazon Resource Name (ARN) dello snapshot DB da migrare. Per ulteriori informazioni su Amazon RDS ARNs, consulta Amazon Relational Database Service (Amazon RDS).

Quando migri la snapshot DB utilizzando il comando `RestoreDBClusterFromSnapshot`, il comando crea sia il cluster database sia l'istanza primaria.

In questo esempio, si crea un cluster DB compatibile con MySQL 5.7 *`mydbcluster`* denominato da un'istantanea DB con un ARN impostato su. *`mydbsnapshotARN`*

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

In Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

In questo esempio, si crea un cluster DB compatibile con MySQL 5.7 *`mydbcluster`* denominato da un'istantanea DB con un ARN impostato su. *`mydbsnapshotARN`*

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

In Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora

Aurora utilizza la funzionalità di replica dei log binari dei motori My SQL DB per creare un tipo speciale di cluster DB chiamato replica di lettura Aurora per un'origine per l'istanza My DB. RDS SQL Gli aggiornamenti apportati all'origine RDS per l'istanza My SQL DB vengono replicati in modo asincrono nella replica di lettura Aurora.

Si consiglia di utilizzare questa funzionalità per migrare da un'istanza RDS for My SQL DB a un cluster Aurora SQL My DB creando una replica di lettura Aurora del codice sorgente per l'RDSistanza My DB. SQL Quando il ritardo di replica tra l'istanza RDS for My SQL DB e la replica di lettura Aurora è 0, è possibile indirizzare le applicazioni client alla replica di lettura Aurora e quindi interrompere la replica per rendere la replica di lettura Aurora un cluster Aurora My DB autonomo. SQL La migrazione potrebbe richiedere del tempo, circa diverse ore per tebibyte (TiB) di dati.

Per l'elenco delle regioni in cui è disponibile Aurora, consulta Amazon Aurora in Riferimenti generali di AWS.

Quando crei una replica di lettura Aurora di un'istanza RDS for My SQL DB, Amazon RDS crea uno snapshot DB della tua fonte RDS per l'istanza My SQL DB (privata per Amazon e senza RDS costi). Amazon migra RDS quindi i dati dallo snapshot DB alla replica di lettura Aurora. Dopo che i dati dello snapshot DB sono stati migrati nel nuovo SQL cluster Aurora My DB, Amazon RDS avvia la replica tra l'SQListanza RDS for My DB e il cluster Aurora My DB. SQL Se l'istanza RDS for My SQL DB contiene tabelle che utilizzano motori di archiviazione diversi da InnoDB o che utilizzano il formato di riga compresso, puoi accelerare il processo di creazione di una replica di lettura Aurora modificando tali tabelle per utilizzare il motore di archiviazione InnoDB e il formato di riga dinamico prima di creare la replica di lettura Aurora. Per ulteriori informazioni sul processo di copia di uno snapshot My SQL DB in un cluster Aurora My SQL DB, vedere. Migrazione dei dati da un'istanza database RDS per MySQL a un cluster di database Amazon Aurora MySQL

È possibile avere una sola replica di lettura Aurora per un'istanza RDS for My SQL DB.

> **Note**
>
> I problemi di replica possono sorgere a causa delle differenze di funzionalità tra Aurora SQL My e la versione del motore di database SQL My dell'RDSistanza for SQL My DB che è la versione principale di replica. Se riscontri un errore, puoi trovare assistenza nel forum della RDS community di Amazon o contattando Supporto AWS.

Non è possibile creare una replica di lettura Aurora se l'istanza RDS for My SQL DB è già l'origine per una replica di lettura interregionale.
Non è possibile migrare ad Aurora SQL My versione 3.05 e successive da alcune versioni RDS precedenti di SQL My 8.0, tra cui 8.0.11, 8.0.13 e 8.0.15. Ti consigliamo di eseguire l'aggiornamento alla versione 8.0.28 di for My prima della migrazione. RDS SQL

Per ulteriori informazioni sulle repliche di SQL lettura personali, consulta Lavorare con le repliche di lettura delle istanze DB MariaDB, My SQL e Postgre. SQL

Creazione di una replica di lettura Aurora

È possibile creare una replica di lettura Aurora per un'istanza RDS for My SQL DB utilizzando la console, il, o il AWS CLI. RDS API

Console

Per creare una replica di lettura Aurora da un'origine RDS per l'istanza My DB SQL

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli l'istanza My SQL DB che desideri utilizzare come origine per la replica di lettura Aurora.
4. Per Actions (Operazioni), scegliere Create Aurora read replica (Crea replica di lettura Aurora).
5. Scegliere le specifiche del cluster di database da utilizzare per la replica di lettura Aurora, come descritto nella tabella seguente.

Opzione	Descrizione
DB instance class (Classe istanza database)	Scegliere una classe di istanza database che definisca i requisiti di elaborazione e di memoria per l'istanza primaria nel cluster DB. Per ulteriori informazioni sulle opzioni di classe di istanza database, consulta Classi di istanze DB Amazon Aurora.
Multi-AZ deployment (Implementazione Multi-AZ)	Seleziona Crea replica in una zona diversa per creare una replica di standby del nuovo cluster di database in un'altra zona di disponibilità nella regione AWS

Opzione	Descrizione
	di destinazione per il supporto per failover. Per altre informazioni sulle zone di disponibilità multiple, consulta Regioni e zone di disponibilità.
DB instance identifier (Identificatore istanze DB)	Digitare un nome per l'istanza primaria nel cluster di database della replica di lettura Aurora. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint per l'istanza primaria del nuovo cluster di database. L'identificatore istanze database presenta i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere unica per tutte le istanze DB di ogni AWS account, per ogni regione. AWS Poiché il cluster di database della replica di lettura Aurora viene creato da uno snapshot dell'istanza database di origine, il nome utente master e la password master per la replica di lettura Aurora corrispondono al nome utente master e alla password master per l'istanza database di origine.
Cloud privato virtuale () VPC	Seleziona VPC per ospitare il cluster DB. Seleziona Crea nuovo VPC per far sì che Aurora ne crei uno VPC per te. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Opzione	Descrizione
Gruppo di sottoreti DB	Scegliere il gruppo di sottoreti di database da utilizzare per il cluster di database. Selezionare Create new DB subnet group (Crea nuovo gruppo di sottoreti DB) per fare in modo che Aurora crei automaticamente un gruppo di sottoreti di database. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.
Public accessibility (Accesso pubblico)	Selezionare `Yes` (Sì) per assegnare al cluster di database un indirizzo IP pubblico. Altrimenti, selezionare `No`. Le istanze nel cluster di database possono essere una combinazione di istanze database pubbliche e private. Per ulteriori informazioni su come non consentire l'accesso pubblico per le istanze, consulta Nascondere cluster database in un VPC da Internet.
Availability zone (Zona di disponibilità)	Stabilire se si desidera specificare una zona di disponibilità particolare. Per ulteriori informazioni sulle zone di disponibilità, consultare Regioni e zone di disponibilità.
VPCgruppo di sicurezza (firewall)	Seleziona Crea nuovo gruppo VPC di sicurezza per fare in modo che Aurora crei un gruppo VPC di sicurezza per te. Seleziona Seleziona gruppi VPC di sicurezza esistenti per specificare uno o più gruppi VPC di sicurezza per proteggere l'accesso di rete al cluster DB. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Opzione	Descrizione
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. I cluster Aurora My SQL DB utilizzano per impostazione predefinita la SQL porta My, 3306. I firewall presso alcune aziende bloccano le connessioni a questa porta. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster di database.
DB parameter group (Gruppo di parametri database)	Seleziona un gruppo di parametri DB per il cluster Aurora My SQL DB. Puoi utilizzare il gruppo parametri del database predefinito di Aurora oppure puoi creare il tuo gruppo parametri del database personalizzato. Per ulteriori informazioni sui gruppi di parametri database, consulta .
DB cluster parameter group (Gruppo di parametri del cluster di database)	Seleziona un gruppo di parametri del cluster DB per il cluster Aurora My SQL DB. Puoi utilizzare il gruppo di parametri del cluster di database predefinito di Aurora oppure puoi creare il tuo gruppo di parametri. Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta .

Opzione	Descrizione
Encryption (Crittografia)	Selezionare Disable encryption (Disabilita crittografia) se non desideri che il nuovo cluster di database sia crittografato. Selezionare Enable Encryption (Abilita crittografia) affinché il nuovo cluster di database Aurora sia crittografato mentre è inattivo. Se scegli Abilita la crittografia, devi scegliere una KMS chiave come AWS KMS keyvalore. Se l'istanza My SQL DB non è crittografata, specifica una chiave di crittografia per crittografare il cluster DB a riposo. Se l'istanza My SQL DB è crittografata, specifica una chiave di crittografia per crittografare il cluster DB a riposo utilizzando la chiave di crittografia specificata. È possibile specificare la chiave di crittografia utilizzat a dall'istanza My SQL DB o una chiave diversa. Non è possibile creare un cluster DB non crittografato da un'istanza My SQL DB crittografata.
Priority (Priorità)	Scegliere una priorità di failover per il cluster di database. Se non si specifica alcun valore, l'imposta zione predefinita è tier-1 (livello 1). Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazioni, consulta Tolleranz a ai guasti di un cluster DB Aurora.
Backup retention period (Periodo di retention dei backup)	Selezionare l'intervallo di tempo, da 1 a 35 giorni, per il quale Aurora conserva le copie di backup del database. Le copie di Backup possono essere utilizzat e per point-in-time ripristinare (PITR) il database fino al secondo.

Opzione	Descrizione
Enhanced Monitoring	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Scegli il IAM ruolo che hai creato per consentire ad Aurora di comunicare con Amazon CloudWatch Logs per te, oppure scegli Predefinito per fare in modo che Aurora crei un ruolo per te denominato. `rds-monitoring-role` Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Questa impostazione non si applica ai cluster Aurora My SQL DB. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consulta. Aggiornamenti del motore di database per Amazon Aurora My SQL

Opzione	Descrizione
Maintenance window (Finestra di manutenzione)	Selezionare Select window (Seleziona finestra) e specifica l'intervallo temporale settimanale durante il quale può avvenire la manutenzione del sistema. Oppure, seleziona No preference (Nessuna preferenz a) per consentire ad Aurora di assegnare un periodo casuale.

6. Scegliere Create read replica (Crea replica di lettura).

AWS CLI

Per creare una replica di lettura Aurora da un'origine RDS per l'istanza My SQL DB, usa i `create-db-instance` AWS CLI comandi `create-db-cluster`and per creare un nuovo cluster Aurora My DB. SQL Quando chiami il `create-db-cluster` comando, includi il `--replication-source-identifier` parametro per identificare l'Amazon Resource Name (ARN) per l'istanza My SQL DB di origine. Per ulteriori informazioni su Amazon RDSARNs, consulta Amazon Relational Database Service (RDSAmazon).

Non specificare il nome utente principale, la password principale o il nome del database poiché la replica di lettura Aurora utilizza lo stesso nome utente principale, password principale e nome del database dell'istanza My SQL DB di origine.

PerLinux, omacOS: Unix

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 \
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Per Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 ^
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Se utilizzi la console per creare una replica di lettura Aurora, Aurora crea automaticamente l'istanza primaria per la replica di lettura Aurora del cluster di database. Se si utilizza il AWS CLI per creare una replica di lettura Aurora, è necessario creare in modo esplicito l'istanza principale per il cluster DB. L'istanza primaria è la prima istanza creata in un cluster di database.

È possibile creare un'istanza primaria per il cluster DB utilizzando il `create-db-instance` AWS CLI comando con i seguenti parametri.

- `--db-cluster-identifier`

  Nome del cluster DB.
- `--db-instance-class`

  Nome della classe dell'istanza database da utilizzare per l'istanza primaria.
- `--db-instance-identifier`

  Nome dell'istanza primaria.
- `--engine aurora`

In questo esempio, si crea un'istanza primaria denominata *`myreadreplicainstance`* per il cluster DB denominato *`myreadreplicacluster`*, utilizzando la classe di istanza DB specificata in *`myinstanceclass`*.

Example

Per LinuxmacOS, oUnix:

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass \
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier myreadreplicacluster ^
    --db-instance-class myinstanceclass ^
    --db-instance-identifier myreadreplicainstance ^
```

```
    --engine aurora
```

RDS API

Per creare una replica di lettura Aurora da un'origine RDS per l'istanza My SQL DB, usa RDS API i comandi e `CreateDBCluster` `CreateDBInstance`Amazon per creare un nuovo cluster Aurora DB e un'istanza primaria. Non specificare il nome utente principale, la password principale o il nome del database poiché la replica di lettura Aurora utilizza lo stesso nome utente principale, password principale e nome del database come origine RDS per l'istanza My SQL DB.

Puoi creare un nuovo cluster Aurora DB per una replica di lettura Aurora da un'origine RDS per l'istanza My SQL DB utilizzando il RDS API comando `CreateDBCluster`Amazon con i seguenti parametri:

- `DBClusterIdentifier`

  Nome del cluster di database da creare.
- `DBSubnetGroupName`

  Nome del gruppo di sottoreti database da associare al cluster DB.
- `Engine=aurora`
- `KmsKeyId`

  Con cui AWS KMS key crittografare facoltativamente il cluster DB, a seconda che l'istanza My SQL DB sia crittografata.
  - Se l'istanza My SQL DB non è crittografata, specifica una chiave di crittografia per crittografare il cluster DB a riposo. Altrimenti, il cluster di database è crittografato mentre è inattivo utilizzando la chiave di crittografia predefinita per l'account.
  - Se l'istanza My SQL DB è crittografata, specifica una chiave di crittografia per crittografare il cluster DB a riposo utilizzando la chiave di crittografia specificata. Altrimenti, il cluster DB viene crittografato a riposo utilizzando la chiave di crittografia per l'istanza My SQL DB.

  > **Note**
  >
  > Non è possibile creare un cluster DB non crittografato da un'istanza My SQL DB crittografata.
- `ReplicationSourceIdentifier`

L'Amazon Resource Name (ARN) per l'istanza My SQL DB di origine. Per ulteriori informazioni su Amazon RDSARNs, consulta Amazon Relational Database Service (RDSAmazon).

- `VpcSecurityGroupIds`

  L'elenco dei gruppi di EC2 VPC sicurezza da associare a questo cluster DB.

In questo esempio, si crea un cluster DB denominato *`myreadreplicacluster`* da un'istanza My SQL DB di origine con un ARN set su *`mysqlprimaryARN`*, associato a un sottogruppo di sottoreti DB denominato *`mysubnetgroup`* e un gruppo VPC di sicurezza denominato *`mysecuritygroup`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBCluster
    &DBClusterIdentifier=myreadreplicacluster
    &DBSubnetGroupName=mysubnetgroup
    &Engine=aurora
    &ReplicationSourceIdentifier=mysqlprimaryARN
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &VpcSecurityGroupIds=mysecuritygroup
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150927/us-east-1/rds/aws4_request
    &X-Amz-Date=20150927T164851Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=6a8f4bd6a98f649c75ea04a6b3929ecc75ac09739588391cd7250f5280e716db
```

Se utilizzi la console per creare una replica di lettura Aurora, Aurora crea automaticamente l'istanza primaria per la replica di lettura Aurora del cluster di database. Se si utilizza il AWS CLI per creare una replica di lettura Aurora, è necessario creare in modo esplicito l'istanza principale per il cluster DB. L'istanza primaria è la prima istanza creata in un cluster di database.

Puoi creare un'istanza primaria per il tuo cluster DB utilizzando il RDS API comando `CreateDBInstance`Amazon con i seguenti parametri:

- `DBClusterIdentifier`

  Nome del cluster DB.

- `DBInstanceClass`

Nome della classe dell'istanza database da utilizzare per l'istanza primaria.

- `DBInstanceIdentifier`

  Nome dell'istanza primaria.

- `Engine=aurora`

In questo esempio, crei un'istanza primaria denominata *`myreadreplicainstance`* per il cluster DB denominato *`myreadreplicacluster`*, utilizzando la classe di istanza DB specificata in *`myinstanceclass`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBInstance
    &DBClusterIdentifier=myreadreplicacluster
    &DBInstanceClass=myinstanceclass
    &DBInstanceIdentifier=myreadreplicainstance
    &Engine=aurora
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140424/us-east-1/rds/aws4_request
    &X-Amz-Date=20140424T194844Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=bee4aabc750bf7dad0cd9e22b952bd6089d91e2a16592c2293e532eeaab8bc77
```

Visualizzazione di una replica di lettura Aurora

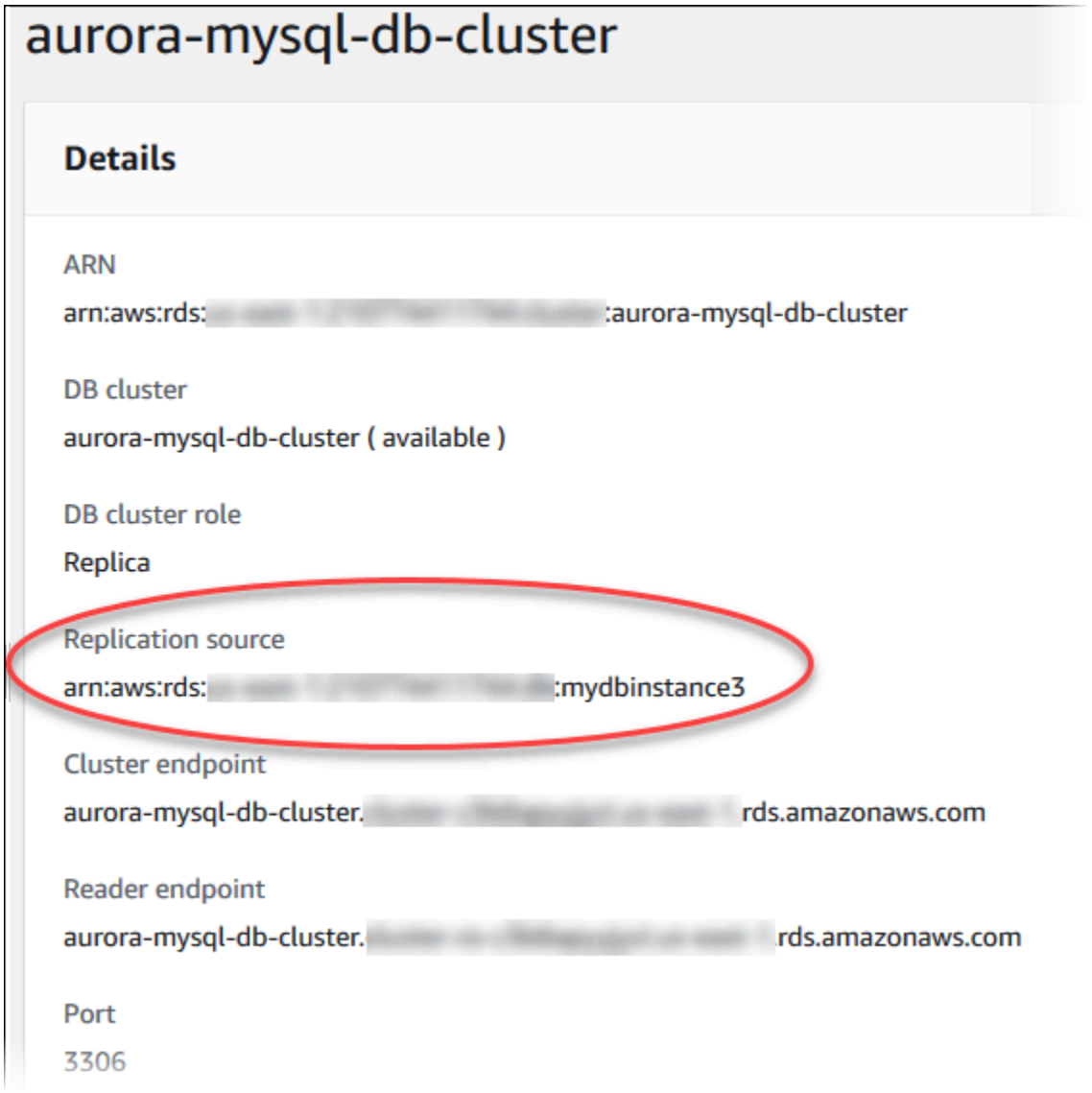
È possibile visualizzare le relazioni SQL di SQL replica tra My e Aurora My per i cluster Aurora My SQL DB utilizzando o il. AWS Management Console AWS CLI

Console

Per visualizzare l'istanza My SQL DB principale per una replica di lettura Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).

3. Selezionare il cluster di database affinché la replica di lettura Aurora mostri i propri dettagli. Le informazioni principali sull'istanza My SQL DB si trovano nel campo Replication source.



AWS CLI

Per visualizzare le relazioni SQL di SQL replica tra My e Aurora My per i cluster Aurora My SQL DB utilizzando i AWS CLI comandi and. `describe-db-clusters` `describe-db-instances`

Per determinare quale istanza My SQL DB è la principale, usa `describe-db-clusters` e specifica l'identificatore del cluster della replica di lettura Aurora per l'opzione. `--db-cluster-identifier` Fate riferimento all'`ReplicationSourceIdentifier` elemento nell'output dell'istanza DB che è la replica primaria. ARN

Per determinare quale cluster DB è la replica di lettura Aurora, usa `describe-db-instances`e specifica l'identificatore di istanza dell'istanza My SQL DB per l'opzione. `--db-instance-identifier` Fai riferimento all'elemento `ReadReplicaDBClusterIdentifiers` nell'output per l'identificatore cluster di database della replica di lettura Aurora.

Example

PerLinux, o: macOS Unix

```
aws rds describe-db-clusters \
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances \
    --db-instance-identifier mysqlprimary
```

Per Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances ^
    --db-instance-identifier mysqlprimary
```

Promozione di una replica di lettura Aurora

Al termine della migrazione, è possibile promuovere la replica di lettura Aurora a un cluster DB autonomo utilizzando o. AWS Management Console AWS CLI

Quindi, puoi indirizzare le applicazioni client all'endpoint per la replica di lettura Aurora. Per ulteriori informazioni sugli endpoint Aurora, consulta Connessioni endpoint Amazon Aurora. La promozione dovrebbe completarsi abbastanza velocemente; nel corso della promozione, è possibile eseguire operazioni di lettura e scrittura sulla replica di lettura Aurora. Tuttavia, non è possibile eliminare l'istanza My SQL DB primaria o scollegare l'istanza DB e la replica di lettura Aurora durante questo periodo.

Prima di promuovere la replica di lettura Aurora, interrompi la scrittura di qualsiasi transazione sull'istanza My SQL DB di origine, quindi attendi che il ritardo di replica sulla replica di lettura Aurora raggiunga lo 0. È possibile visualizzare il ritardo della replica di una replica di lettura di Aurora chiamando il comando (`SHOW SLAVE STATUS`Aurora My versione SQL 2) o (Aurora My versione

SQL 3) sulla replica di lettura di `SHOW REPLICA STATUS` Aurora. Controlla il valore secondi di latenza dal principale.

Puoi iniziare a scrivere la replica di lettura Aurora dopo che le transazioni di scrittura al master sono iniziate e il ritardo della replica equivale a 0. Se si scrive sulla replica di lettura di Aurora prima di questa operazione e si modificano tabelle che vengono modificate anche su My SQL primary, si rischia di interrompere la replica su Aurora. Se ciò avviene, devi eliminare la replica di lettura Aurora e crearla di nuovo.

Console

Come promuovere una replica di lettura Aurora a cluster di database Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona il cluster di database per la replica di lettura Aurora.
4. In Actions (Operazioni), selezionare Promote (Promuovi).
5. Selezionare Promote read replica (Promuovi replica di lettura).

Dopo la promozione, verificare che la promozione sia stata completata utilizzando la procedura seguente.

Per confermare che la replica di lettura Aurora è stata promossa

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione selezionare Events (Eventi).
3. Nella pagina Events (Eventi), verificare che esista un evento `Promoted Read Replica cluster to a stand-alone database cluster` per il cluster promosso.

Una volta completata la promozione, l'istanza My SQL DB principale e la replica di lettura Aurora vengono scollegate e, se lo desideri, puoi eliminare l'istanza DB in tutta sicurezza.

AWS CLI

Per promuovere una replica di lettura Aurora in un cluster DB autonomo, usa il comando. `promote-read-replica-db-cluster` AWS CLI

Example

PerLinux, o: macOS Unix

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Per Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

Gestione di Amazon Aurora MySQL

Nelle sezioni seguenti viene descritta la gestione di un cluster database Amazon Aurora MySQL.

Argomenti

- Gestione delle prestazioni e del dimensionamento per Amazon Aurora MySQL
- Backtrack di un cluster database Aurora
- Test di Amazon Aurora My SQL using Fault Injection
- Alterazione delle tabelle in Amazon Aurora mediante Fast DDL
- Visualizzazione dello stato del volume per un cluster DB Aurora MySQL

Gestione delle prestazioni e del dimensionamento per Amazon Aurora MySQL

Dimensionamento delle istanze database Aurora MySQL

È possibile dimensionare le istanze database Aurora MySQL in due diversi modi, ovvero tramite il dimensionamento delle istanze e il dimensionamento della lettura. Per ulteriori informazioni sul dimensionamento della lettura, consulta Dimensionamento della lettura.

Puoi dimensionare il cluster database Aurora MySQL in base alle necessità, modificando la classe di istanza database per ogni istanza database nel cluster database. Aurora MySQL supporta diverse classi di istanza database ottimizzate per Aurora. Non utilizzare classi di istanza db.t2 o db.t3 per cluster Aurora più grandi di dimensioni maggiori di 40 TB. Per le specifiche delle classi di istanza database supportate da Aurora MySQL, consulta Classi di istanze DB Amazon Aurora.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Numero massimo di connessioni a un'istanza database Aurora MySQL

Il numero massimo di connessioni consentite a un'istanza database di Aurora MySQL è determinato dal parametro `max_connections` nel gruppo di parametri a livello di istanza per l'istanza database.

La tabella seguente elenca il valore predefinito risultante di `max_connections` per ciascuna classe di istanza database disponibile per Aurora MySQL. È possibile aumentare il numero massimo di connessioni all'istanza database Aurora MySQL eseguendo il dimensionamento dell'istanza a una classe di istanza database con più memoria oppure impostando un valore maggiore per il parametro `max_connections` nel gruppo di parametri DB per l'istanza, fino a 16.000.

> **Tip**
>
> Se le applicazioni aprono e chiudono frequentemente connessioni o mantengono un numero elevato di connessioni di lunga durata aperte, ti consigliamo di utilizzare Amazon RDS Proxy. Proxy RDS è un proxy di database completamente gestito e ad alta disponibilità che utilizza il pooling di connessioni per condividere connessioni di database in modo sicuro ed efficiente. Per ulteriori informazioni sul Proxy RDS, consulta Proxy Amazon RDS per Aurora.

Per informazioni dettagliate su come le istanze Aurora Serverless v2 gestiscono questo parametro, vedi Numero massimo connessioni per Aurora Serverless v2.

Classe di istanza	Valore predefinito di max_connections		
db.t2.small	45		
db.t2.medium	90		
db.t3.small	45		
db.t3.medium	90		
db.t3.large	135		
db.t4g.medium	90		
db.t4g.large	135		
db.r3.large	1000		

Classe di istanza	Valore predefinito di max_connections		
db.r3.xlarge	2000		
db.r3.2xlarge	3000		
db.r3.4xlarge	4000		
db.r3.8xlarge	5000		
db.r4.large	1000		
db.r4.xlarge	2000		
db.r4.2xlarge	3000		
db.r4.4xlarge	4000		
db.r4.8xlarge	5000		
db.r4.16xlarge	6000		
db.r5.large	1000		
db.r5.xlarge	2000		
db.r5.2xlarge	3000		
db.r5.4xlarge	4000		
db.r5.8xlarge	5000		
db.r5.12xlarge	6000		
db.r5.16xlarge	6000		
db.r5.24xlarge	7000		

Classe di istanza	Valore predefinito di max_connections		
db.r6g.large	1000		
db.r6g.xlarge	2000		
db.r6g.2xlarge	3000		
db.r6g.4xlarge	4000		
db.r6g.8xlarge	5000		
db.r6g.12xlarge	6000		
db.r6g.16xlarge	6000		
db.r6i.large	1000		
db.r6i.xlarge	2000		
db.r6i.2xlarge	3000		
db.r6i.4xlarge	4000		
db.r6i.8xlarge	5000		
db.r6i.12xlarge	6000		
db.r6i.16xlarge	6000		
db.r6i.24xlarge	7000		
db.r6i.32xlarge	7000		
db.r7g.large	1000		
db.r7g.xlarge	2000		

Classe di istanza	Valore predefinito di max_connections		
db.r7g.2xlarge	3000		
db.r7g.4xlarge	4000		
db.r7g.8xlarge	5000		
db.r7g.12xlarge	6000		
db.r7g.16xlarge	6000		
db.r7i.large	1000		
db.r7i.xlarge	2000		
db.r7i.2xlarge	3000		
db.r7i.4xlarge	4000		
db.r7i.8xlarge	5000		
db.r7i.12xlarge	6000		
db.r7i.16xlarge	6000		
db.r7i.24xlarge	7000		
db.r7i.48xlarge	8000		
db.r8g.large	1000		
db.r8g.xlarge	2000		
db.r8g.2xlarge	3000		
db.r8g.4xlarge	4000		

Classe di istanza	Valore predefinito di max_connections
db.r8g.8xlarge	5000
db.r8g.12xlarge	6000
db.r8g.16xlarge	6000
db.r8g.24xlarge	7000
db.r8g.48xlarge	8000
db.x2g.large	2000
db.x2g.xlarge	3000
db.x2g.2xlarge	4000
db.x2g.4xlarge	5000
db.x2g.8xlarge	6000
db.x2g.12xlarge	7000
db.x2g.16xlarge	7000

Tip

Il calcolo `max_connections` dei parametri utilizza log base 2 (distinto dal logaritmo naturale) e il `DBInstanceClassMemory` valore in byte per la classe di istanza Aurora MySQL selezionata. Il parametro accetta solo valori interi, con le porzioni decimali troncate dai calcoli. La formula implementa i limiti di connessione come segue:

- 1000 incrementi di connessione per istanze R3, R4 e R5 più grandi
- Incremento di 45 connessioni per le varianti di memoria delle istanze T2 e T3

Esempio: per db.r6g.large, mentre la formula calcola 1069,2, il sistema implementa 1000 per mantenere modelli incrementali coerenti.

Se crei un nuovo gruppo di parametri per personalizzare il valore predefinito del limite di connessioni, verificherai che il limite di connessioni predefinito deriva da una formula basata sul valore `DBInstanceClassMemory`. Come illustrato nella precedente tabella, la formula produce limiti di connessione che aumentano di 1000 al raddoppio della memoria tra istanze R3, R4 e R5 sempre più grandi e di 45 per dimensioni di memoria diverse delle istanze T2 e T3.

Consultare Specifica dei parametri del database per ulteriori informazioni su come è calcolato `DBInstanceClassMemory`.

Le istanze database Aurora MySQL e RDS for MySQL hanno quantità di sovraccarico della memoria differenti. Pertanto, il valore `max_connections` può essere diverso per istanze database Aurora MySQL e RDS MySQL che utilizzano la stessa classe di istanza. I valori nella tabella si applicano solo alle istanze database Aurora MySQL.

Note

I limiti di connettività sensibilmente inferiori per le istanze T2 e T3 sono dovute al fatto che con Aurora queste classi di istanze sono destinate solo a scenari di test e sviluppo e non ai carichi di lavoro di produzione.

I limiti di connessione predefiniti si adattano ai sistemi che utilizzano i valori predefiniti per altre funzioni ad alto consumo di memori, come il pool di buffer e la cache delle query. Se modifichi queste altre impostazioni per il cluster, pensa a regolare il limite di connessione in base alla possibilità di aumento o diminuzione della memoria disponibile nelle istanze database.

Limiti di storage temporaneo per Aurora MySQL

Aurora MySQL memorizza tabelle e indici nel sottosistema di archiviazione Aurora. Aurora MySQL utilizza uno storage temporaneo o locale separato per file temporanei non persistenti e tabelle temporanee non InnoDB. Inoltre, sono inclusi i file utilizzati per scopi quali l'ordinamento di set di dati di grandi dimensioni durante l'elaborazione delle query o per le operazioni di creazione dell'indice. Non include le tabelle temporanee InnoDB.

Per ulteriori informazioni sulle tabelle temporanee in Aurora MySQL versione 3, consulta Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3. Per ulteriori informazioni sulle tabelle temporanee nella versione 2, consulta Comportamento temporaneo del tablespace in Aurora My versione 2 SQL.

I dati e i file temporanei su questi volumi vengono persi all'avvio e all'arresto dell'istanza DB e durante la sostituzione dell'host.

Questi volumi di storage locale sono supportati da Amazon Elastic Block Store (EBS) e possono essere estesi utilizzando una classe di istanza DB più grande. Per ulteriori informazioni sullo storage, consultare Archiviazione Amazon Aurora.

Lo storage locale viene utilizzato anche per importare dati da Amazon S3 `LOAD DATA FROM S3` utilizzando `LOAD XML FROM S3` o e per esportare dati in S3 utilizzando SELECT INTO OUTFILE S3. Per ulteriori informazioni sull'importazione e l'esportazione in S3, consulta quanto segue:

- Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3
- Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3

Aurora MySQL utilizza uno storage permanente separato per i log degli errori, i log generali, i log delle query lente e i log di controllo per la maggior parte delle classi di istanze Aurora MySQL DB (esclusi i tipi di classi di istanze burstable-performance come db.t2, db.t3 e db.t4g). I dati su questo volume vengono conservati all'avvio e all'arresto dell'istanza DB e durante la sostituzione dell'host.

Questo volume di storage permanente è supportato anche da Amazon EBS e ha una dimensione fissa in base alla classe di istanza DB. Non può essere esteso utilizzando una classe di istanza DB più grande.

La tabella seguente mostra la quantità massima di storage temporaneo e permanente disponibile per ogni classe di istanza Aurora MySQL DB. Per ulteriori informazioni sul supporto della classe di istanza database per Aurora, consultare Classi di istanze DB Amazon Aurora.

DB instance class (Classe istanza database)	temporary/local Storage massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di registro (GiB)
db.x2g.16xlarge	1280	500
db.x2g.12xlarge	960	500
db.x2g.8xlarge	640	500
db.x2g.4xlarge	320	500
db.x2g.2xlarge	160	60
db.x2g.xlarge	80	60
db.x2g.large	40	60
db.r8g.48xlarge	3840	500
db.r8g.24xlarge	1920	500
db.r8g.16xlarge	1280	500
db.r8g.12xlarge	960	500
db.r8g.8xlarge	640	500
db.r8g.4xlarge	320	500
db.r8g.2xlarge	160	60
db.r8g.xlarge	80	60
db.r8g.large	32	60
db.r7i.48xlarge	3840	500
db.r7i.24xlarge	1920	500
db.r7i.16xlarge	1280	500

DB instance class (Classe istanza database)	temporary/local Storage massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di registro (GiB)
db.r7i.12xlarge	960	500
db.r7i.8xlarge	640	500
db.r7i.4xlarge	320	500
db.r7i.2xlarge	160	60
db.r7i.xlarge	80	60
db.r7i.large	32	60
db.r7g.16xlarge	1280	500
db.r7g.12xlarge	960	500
db.r7g.8xlarge	640	500
db.r7g.4xlarge	320	500
db.r7g.2xlarge	160	60
db.r7g.xlarge	80	60
db.r7g.large	32	60
db.r6i.32xlarge	2560	500
db.r6i.24xlarge	1920	500
db.r6i.16xlarge	1280	500
db.r6i.12xlarge	960	500
db.r6i.8xlarge	640	500
db.r6i.4xlarge	320	500

DB instance class (Classe istanza database)	temporary/local Storage massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di registro (GiB)
db.r6i.2xlarge	160	60
db.r6i.xlarge	80	60
db.r6i.large	32	60
db.r6g.16xlarge	1280	500
db.r6g.12xlarge	960	500
db.r6g.8xlarge	640	500
db.r6g.4xlarge	320	500
db.r6g.2xlarge	160	60
db.r6g.xlarge	80	60
db.r6g.large	32	60
db.r5.24xlarge	1920	500
db.r5.16xlarge	1280	500
db.r5.12xlarge	960	500
db.r5.8xlarge	640	500
db.r5.4xlarge	320	500
db.r5.2xlarge	160	60
db.r5.xlarge	80	60
db.r5.large	32	60
db.r4.16xlarge	1280	500

DB instance class (Classe istanza database)	temporary/local Storage massimo disponibile (GiB)	Spazio di archiviazione massimo aggiuntivo disponibile per i file di registro (GiB)
db.r4.8xlarge	640	500
db.r4.4xlarge	320	500
db.r4.2xlarge	160	60
db.r4.xlarge	80	60
db.r4.large	32	60
db.t4g.large	32	–
db.t4g.medium	32	–
db.t3.large	32	–
db.t3.medium	32	–
db.t3.small	32	–
db.t2.medium	32	–
db.t2.small	32	–

Important

Questi valori rappresentano la quantità massima teorica di spazio di archiviazione libero su ogni istanza database. Lo spazio di archiviazione locale effettivo disponibile potrebbe essere inferiore. Aurora utilizza una parte dello spazio di archiviazione locale per i processi di gestione e l'istanza database utilizza una parte dello spazio di archiviazione locale anche prima che vengano caricati i dati. È possibile monitorare lo spazio di archiviazione temporaneo disponibile per una specifica istanza DB con la `FreeLocalStorage` CloudWatch metrica descritta in. CloudWatch Parametri Amazon per Amazon Aurora Al momento è possibile verificare la quantità di spazio di archiviazione gratuito. È inoltre possibile tracciare la quantità di spazio di archiviazione gratuito nel tempo. Il monitoraggio

dello spazio di archiviazione gratuito nel tempo consente di determinare se il valore è in aumento o in diminuzione o di trovare i valori minimi, massimi o medi.
(Non valido per Aurora Serverless v2)

Backtrack di un cluster database Aurora

Con Amazon Aurora My SQL -Compatible Edition, puoi eseguire il backtrack di un cluster DB fino a un'ora specifica, senza ripristinare i dati da un backup.

Indice

- Panoramica del backtrack
  - Finestra di backtrack
  - Orario di backtrack
  - Limiti del backtrack
- Disponibilità di regioni e versioni
- Considerazioni sull'aggiornamento per i cluster abilitati per backtrack
- Configurazione del backtracking di un cluster Aurora My DB SQL
- Esecuzione di un backtrack per un cluster Aurora My DB SQL
- Monitoraggio del backtracking per un cluster Aurora My DB SQL
- Abbonamento a un evento di backtrack con la console
- Recupero di backtrack esistenti
- Disabilitazione del backtracking per un cluster Aurora My DB SQL

Panoramica del backtrack

Il backtrack "riavvolge" il cluster database all'orario che specifichi. Il backtrack non è un'operazione sostitutiva del backup del cluster database che consente di eseguire un ripristino point-in-time dello stesso. Fornisce tuttavia i seguenti vantaggi rispetto alle operazioni di backup e ripristino tradizionali:

- Puoi annullare facilmente eventuali errori. Se esegui erroneamente un'azione distruttiva, ad esempio una DELETE WHERE clausola without a, puoi tornare indietro nel cluster DB a un periodo precedente all'azione distruttiva con un'interruzione minima del servizio.
- Puoi eseguire il backtrack di un cluster database rapidamente. Il ripristino point-in-time di un cluster database avvia un nuovo cluster database e lo ripristina a partire dai dati di backup o da

uno snapshot e questa operazione può durare varie ore. Il backtrack di un cluster database non richiede un nuovo cluster database e "riavvolge" il cluster all'orario specificato in pochi minuti.

- Puoi esplorare le modifiche ai dati precedenti. Puoi eseguire ripetutamente il backtrack di un cluster database avanti e indietro nel tempo per determinare quando si è verificata una determinata modifica. Ad esempio, puoi eseguire il backtrack di un cluster database tre ore indietro e quindi un'ora avanti. In questo caso, l'orario di backtrack è due ore prima l'orario di origine.

> **Note**
>
> Per informazioni sul ripristino point-in-time di un cluster database, consulta Panoramica di backup e ripristino di un cluster di database Aurora.

Finestra di backtrack

Con il backtrack, si ha una finestra di backtrack target e una finestra di backtrack effettiva:

- La finestra di backtrack target è il periodo di tempo durante il quale desideri eseguire il backtrack del cluster di database. Quando abiliti il backtrack, specifichi una finestra di backtrack target. Ad esempio, potresti specificare una finestra di backtrack target di 24 ore se desideri eseguire un backtrack di un giorno per il cluster di database.
- La finestra di backtrack effettiva è il periodo di tempo effettivo durante il quale puoi eseguire il backtrack del cluster di database, che può essere inferiore alla finestra di backtrack target. La finestra di backtrack effettiva è basata sul carico di lavoro e sullo storage disponibile per archiviare le informazioni relative alle modifiche del database denominate record di modifica.

Quando aggiorni il cluster di database Aurora con il backtrack abilitato, generi record di modifica. Aurora conserva i record di modifica per la finestra di backtrack di destinazione e per l'archiviazione di questi dati si applica una tariffa oraria. La finestra di backtrack target e il carico di lavoro sul cluster di database determinano il numero di record di modifica che vengono archiviati. Il carico di lavoro è il numero di modifiche che apporti al cluster di database in un determinato periodo di tempo. Il numero di record di modifica archiviati nella finestra di backtrack quando il carico di lavoro è pesante è maggiore rispetto a quello con un carico di lavoro leggero.

Puoi considerare la finestra di backtrack target come l'obiettivo per il periodo di tempo massimo durante il quale desideri eseguire il backtrack del cluster di database. Nella maggior parte dei casi, puoi eseguire un backtrack corrispondente al periodo di tempo massimo specificato. Tuttavia, in

alcuni casi, il cluster di database non può archiviare un numero sufficiente di record di modifica per eseguire un backtrack corrispondente al periodo di tempo massimo e la finestra di backtrack effettiva è più piccola della finestra di backtrack target. In genere, la finestra di backtrack effettiva è più piccola di quella target quando il carico di lavoro è estremamente pesante sul cluster di database. Quando la finestra di backtrack effettiva è più piccola di quella target, ti inviamo una notifica.

Quando il backtrack è abilitato per un cluster di database ed elimini una tabella archiviata in quel cluster, Aurora conserva quella tabella nei record di modifica del backtrack. In questo modo, puoi ritornare a un orario antecedente a quello in cui hai eliminato la tabella. Se tuttavia non hai spazio sufficiente nella finestra di backtrack per archiviare la tabella, è possibile che questa venga rimossa dai record di modifica del backtrack.

Orario di backtrack

Aurora esegue sempre il backtrack a un orario coerente per il cluster di database. In questo modo, viene eliminata la possibilità di transazioni di cui non è stato eseguito il commit al completamento del backtrack. Quando specifichi un orario per un backtrack, Aurora sceglie automaticamente l'orario coerente più vicino possibile. Questo approccio significa che il backtrack completato potrebbe non corrispondere esattamente all'ora specificata, ma è possibile determinare l'ora esatta per un backtrack utilizzando describe-db-cluster-backtracks AWS CLIcomando. Per ulteriori informazioni, consulta Recupero di backtrack esistenti.

Limiti del backtrack

I seguenti limiti si applicano al backtrack:

- Il backtrack è disponibile solo per i cluster di database creati con la funzionalità di backtrack abilitata. Non è possibile modificare un cluster DB per abilitare la funzionalità Backtrack. Puoi abilitare il backtrack quando crei un nuovo cluster di database o ripristini uno snapshot di un cluster di database.
- Il limite per una finestra di backtrack è 72 ore.
- Il backtrack influisce sull'intero cluster di database. Ad esempio, non puoi eseguire un backtrack selettivo di una singola tabella o di un singolo aggiornamento di dati.
- Non è possibile creare repliche di lettura interregionali da un cluster abilitato al backtrack, ma è comunque possibile abilitare la replica dei log binari (binlog) sul cluster. Se si tenta di eseguire il backtrack di un cluster DB per il quale è abilitata la registrazione binaria, in genere si verifica un errore a meno che non si scelga di forzare il backtrack. Qualsiasi tentativo di forzare un backtrack interromperà le repliche di lettura a valle e interferirà con altre operazioni, come le implementazioni blu/green.

- Non puoi eseguire il backtrack di un clone di database a un orario antecedente a quello di creazione del clone. Puoi tuttavia utilizzare il database originale per eseguire il backtrack a un orario antecedente a quello di creazione del clone. Per ulteriori informazioni sulla creazione di cloni di database, consulta Clonazione di un volume per un cluster di database Amazon Aurora.
- Il backtrack comporta una breve interruzione dell'istanza database. Devi arrestare o interrompere le applicazioni prima di avviare un'operazione di backtrack per assicurarti che non vi siano nuove richieste di lettura o scrittura. Durante l'operazione di backtrack, Aurora interrompe il database, chiude tutte le connessioni aperte ed elimina le letture e scritture di cui non è stato eseguito il commit. Attende quindi il completamento dell'operazione di backtrack.
- Non è possibile ripristinare un'istantanea interregionale di un cluster abilitato al backtrack in un AWS Regione che non supporta il backtracking.
- Se si esegue un aggiornamento sul posto per un cluster abilitato al backtrack da Aurora SQL My versione 2 alla versione 3, non è possibile tornare indietro a un momento precedente all'aggiornamento.

Disponibilità di regioni e versioni

Backtrack non è disponibile per SQL Aurora Postgre.

Di seguito sono riportati i motori supportati e la disponibilità regionale per Backtrack with SQL Aurora My.

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Stati Uniti orientali (Virginia settentrionale)	Tutte le versioni	Tutte le versioni
Stati Uniti orientali (Ohio)	Tutte le versioni	Tutte le versioni
Stati Uniti occidentali (California settentrionale)	Tutte le versioni	Tutte le versioni

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
US West (Oregon)	Tutte le versioni	Tutte le versioni
Africa (Città del Capo)	–	–
Asia Pacifico (Hong Kong)	–	–
Asia Pacifico (Giacarta)	–	–
Asia Pacifico (Malesia)	–	–
Asia Pacifico (Melbourne)	–	–
Asia Pacifico (Mumbai)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Osaka-Locale)	Tutte le versioni	Versione 2.07.3 e versioni successive
Asia Pacifico (Seoul)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Singapore)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Sydney)	Tutte le versioni	Tutte le versioni
Asia Pacifico (Tokyo)	Tutte le versioni	Tutte le versioni
Canada (Centrale)	Tutte le versioni	Tutte le versioni

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
Canada occidentale (Calgary)	–	–
Cina (Pechino)	–	–
China (Ningxia)	–	–
Europa (Francoforte)	Tutte le versioni	Tutte le versioni
Europa (Irlanda)	Tutte le versioni	Tutte le versioni
Europa (Londra)	Tutte le versioni	Tutte le versioni
Europa (Milano)	–	–
Europa (Parigi)	Tutte le versioni	Tutte le versioni
Europa (Spagna)	–	–
Europa (Stoccolma)	–	–
Europa (Zurigo)	–	–
Israele (Tel Aviv)	–	–
Medio Oriente (Bahrein)	–	–
Medio Oriente () UAE	–	–
Sud America (San Paolo)	–	–

Regione	Aurora La mia versione 3 SQL	Aurora La mia versione 2 SQL
AWS GovCloud (Stati Uniti orientali)	–	–
AWS GovCloud (Stati Uniti occidentali)	–	–

Considerazioni sull'aggiornamento per i cluster abilitati per backtrack

È possibile aggiornare un cluster DB abilitato al backtrack da Aurora My SQL versione 2 alla versione 3, poiché tutte le versioni minori di Aurora My versione 3 sono supportate per Backtrack. SQL

Configurazione del backtracking di un cluster Aurora My DB SQL

Per utilizzare la funzionalità di backtrack, devi abilitarla e specificare una finestra di backtrack target. In caso contrario, il backtrack è disabilitato.

Nella finestra di backtrack target, specifica il periodo di tempo durante il quale desideri "riavvolgere" il database utilizzando la funzionalità di backtrack. Aurora prova conservare un numero di record di modifica sufficiente per supportare quella finestra temporale.

Console

Puoi utilizzare la console per configurare il backtrack quando crei un nuovo cluster di database. È inoltre possibile modificare un cluster DB per modificare la finestra di backtrack per un cluster abilitato per backtrack. Se si disattiva completamente il backtracking per un cluster impostando la finestra backtrack su 0, non sarà poi possibile attivare nuovamente il backtrack per quel cluster.

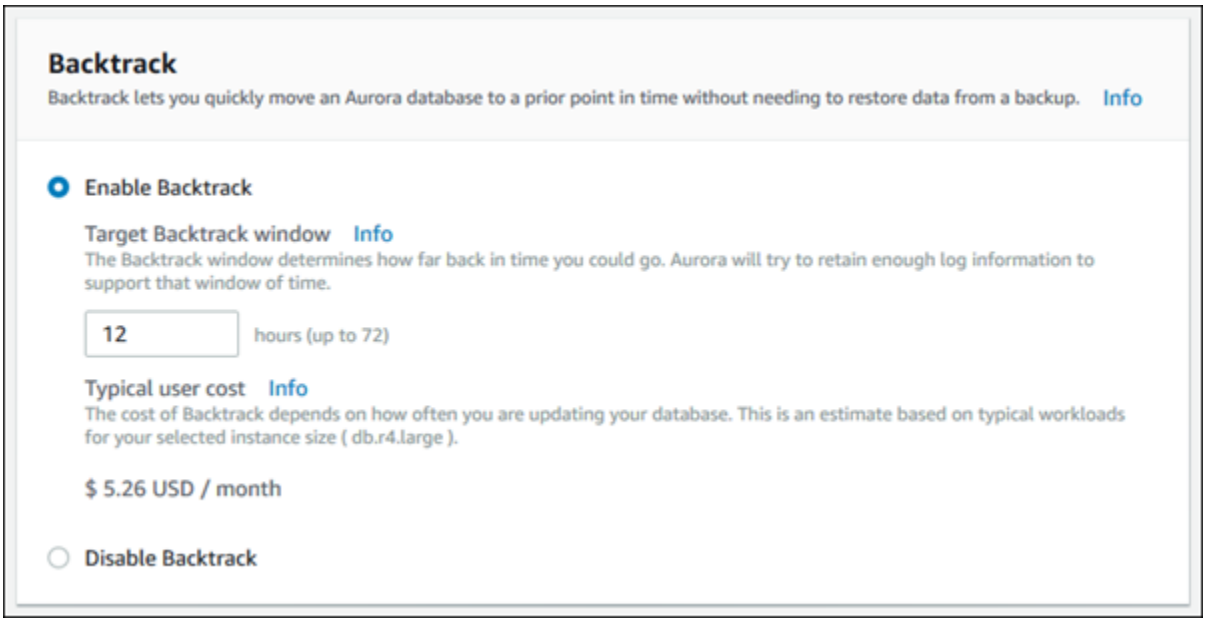
Argomenti

- Configurazione del backtrack mediante la console alla creazione di un cluster di database
- Configurazione del backtrack mediante la console alla modifica di un cluster database

Configurazione del backtrack mediante la console alla creazione di un cluster di database

Quando crei un nuovo cluster Aurora My SQL DB, il backtracking viene configurato quando scegli Abilita Backtrack e specifichi un valore della finestra Target Backtrack maggiore di zero nella sezione Backtrack.

Per creare un cluster database, segui le istruzioni in Creazione di un cluster database Amazon Aurora. L'immagine seguente mostra la sezione Backtrack.



Quando crei un nuovo cluster database, Aurora non dispone dei dati relativi al carico di lavoro del cluster database. Di conseguenza, non può stimare un costo per il nuovo cluster database. La console indica tuttavia un costo utente standard per la finestra di backtrack target specificata basato su un carico di lavoro tipico. Questo costo standard è un riferimento generale per il costo della funzionalità di backtrack.

> **Important**
>
> Il costo effettivo potrebbe non corrispondere a quello standard in quanto è basato sul carico di lavoro del cluster database.

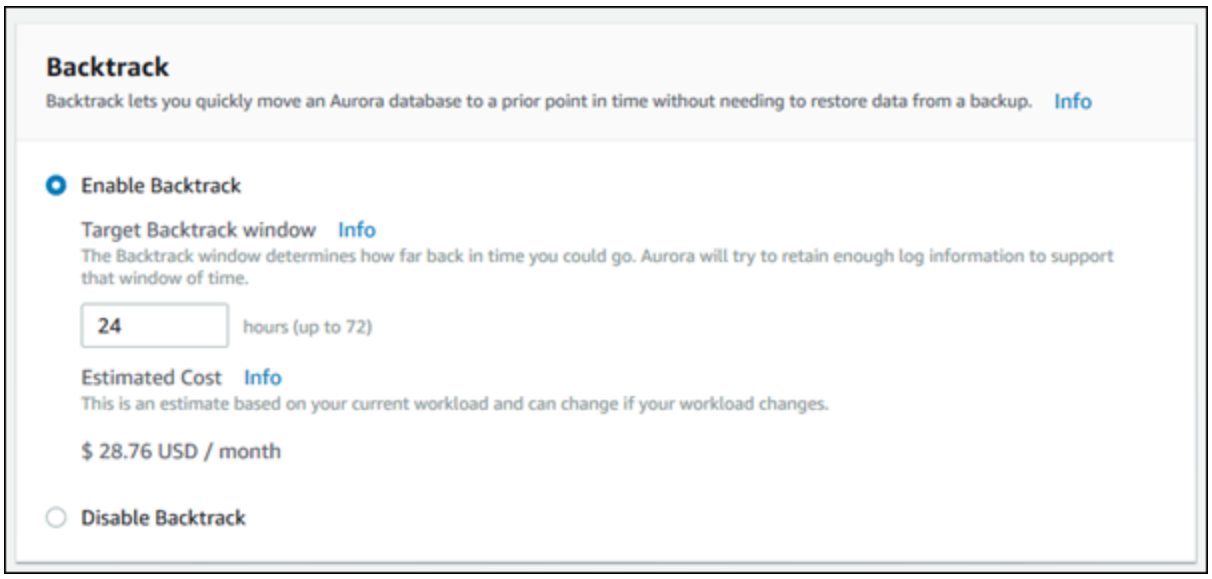
Configurazione del backtrack mediante la console alla modifica di un cluster database

Puoi modificare il backtrack per un cluster database mediante la console.

> **Note**
>
> Attualmente, è possibile modificare il backtracking solo per un cluster DB in cui è abilitata la funzionalità Backtrack. La sezione Backtrack non viene visualizzata per un cluster DB creato con la funzionalità Backtrack disattivata o se la funzionalità Backtrack è stata disattivata per il cluster DB.

Per modificare il backtrack per un cluster database mediante la console

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il cluster da modificare, quindi scegliere Modify (Modifica).
4. In Target Backtrack window (Finestra di backtrack target), modificare il periodo di tempo durante il quale si desidera eseguire il backtrack. Il limite è 72 ore.



   La console mostra il costo stimato per il periodo di tempo specificato in base al carico di lavoro precedente del cluster database:

   - Se il backtracking è stato disabilitato sul cluster DB, la stima dei costi si basa sulla `VolumeWriteIOPS` metrica per il cluster DB in Amazon. CloudWatch

- Se il backtracking era stato abilitato in precedenza sul cluster DB, la stima dei costi si basa sulla `BacktrackChangeRecordsCreationRate` metrica per il cluster DB in Amazon. CloudWatch

5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere una delle seguenti opzioni:

   - Apply during the next scheduled maintenance window (Applica durante la prossima finestra di manutenzione pianificata): –attendere la finestra di manutenzione successiva per applicare la modifica di Target Backtrack window (Finestra di backtrack target).
   - Apply immediately (Applica immediatamente): –applicare la modifica di Target Backtrack window (Finestra di backtrack target) appena possibile.
7. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Quando si crea un nuovo cluster Aurora My SQL DB utilizzando il create-db-cluster AWS CLIcomando, il backtracking viene configurato quando si specifica un `--backtrack-window` valore maggiore di zero. Il valore `--backtrack-window` specifica la finestra di backtrack target. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

È inoltre possibile specificare il `--backtrack-window` valore utilizzando i seguenti comandi: AWS CLI

- modify-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-istantanea
- restore-db-cluster-to-point-in-time

La procedura seguente descrive come modificare la finestra di backtrack target per un cluster database mediante l AWS CLI.

Per modificare la finestra di backtrack di destinazione per un cluster DB utilizzando il AWS CLI

- Chiamate il modify-db-cluster AWS CLIcomando e fornite i seguenti valori:

  - `--db-cluster-identifier`:– il nome del cluster database.

- `--backtrack-window`:– il numero massimo di secondi durante i quali si desidera eseguire il backtrack del cluster database.

L'esempio seguente imposta la finestra di backtrack target per `sample-cluster` su un giorno (86.400 secondi).

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 86400
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 86400
```

> **Note**
>
> Attualmente, è possibile abilitare il backtrack solo per un cluster database creato con la funzionalità di backtrack abilitata.

RDS API

Quando si crea un nuovo cluster Aurora My SQL DB utilizzando l'RDSAPIoperazione C reateDBClusterAmazon, il backtracking viene configurato quando si specifica un `BacktrackWindow` valore maggiore di zero. Il valore `BacktrackWindow` specifica la finestra di backtrack target per il cluster database specificato nel valore `DBClusterIdentifier`. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Puoi anche specificare il `BacktrackWindow` valore utilizzando le seguenti operazioni: API

- M odifyDBCluster
- R di estoreDBCluster S3

- R estoreDBCluster FromSnapshot
- R estoreDBCluster ToPointInTime

> **Note**
>
> Attualmente, è possibile abilitare il backtrack solo per un cluster database creato con la funzionalità di backtrack abilitata.

Esecuzione di un backtrack per un cluster Aurora My DB SQL

Puoi eseguire il backtrack di un cluster database a un timestamp di backtrack specificato. Se il timestamp di backtrack non è anteriore al primo orario di backtrack possibile e non è nel futuro, il backtrack del cluster database viene eseguito a quel timestamp.

In caso contrario, si ha in genere un errore. Inoltre, se cerchi di eseguire il backtrack di un cluster database per il quale la registrazione binaria è abilitata, in genere si verifica un errore a meno che non hai scelto di forzare il backtrack. Forzare un backtrack può interferire con altre operazioni che utilizzano la registrazione binaria.

> **Important**
>
> Il backtrack non genera voci binlog per le modifiche che esegue. Se la registrazione binaria è abilitata per il cluster database, il backtrack potrebbe non essere compatibile con l'implementazione binlog.

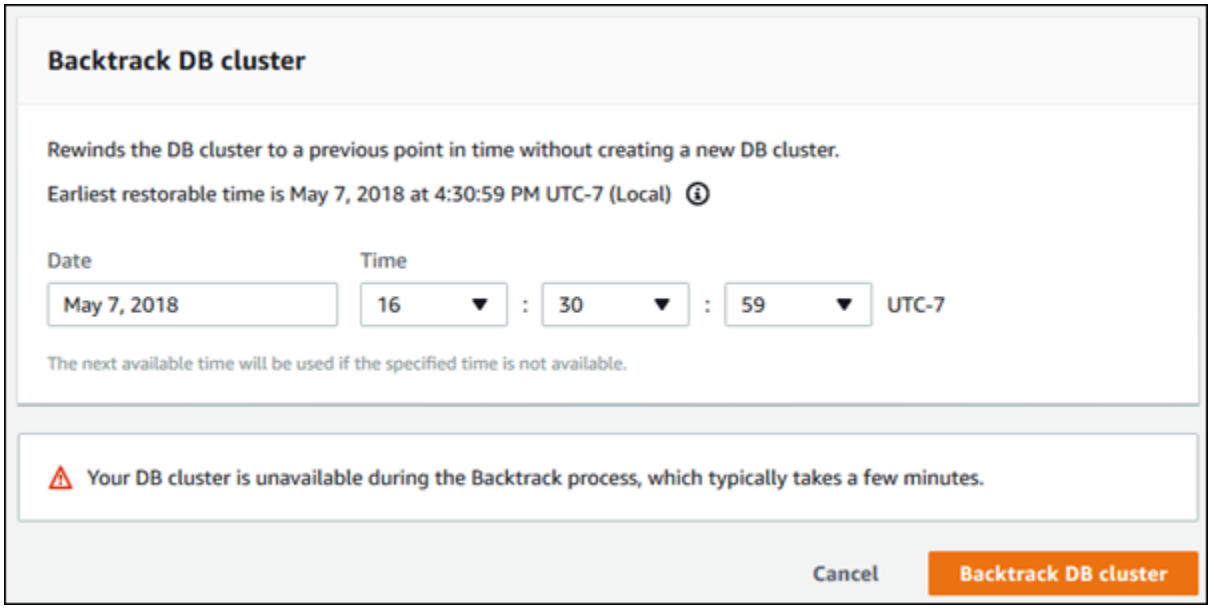
> **Note**
>
> Per i cloni di database, non puoi eseguire il backtrack del cluster database a un orario anteriore a quello di creazione del clone. Per ulteriori informazioni sulla creazione di cloni di database, consulta Clonazione di un volume per un cluster di database Amazon Aurora.

Console

La procedura seguente descrive come eseguire un'operazione di backtrack per un cluster database mediante la console.

Per eseguire un'operazione di backtrack mediante la console

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Instances (Istanze).
3. Scegliere l'istanza principale del cluster database per il quale si intende eseguire il backtrack.
4. In Actions (Operazioni), scegliere Backtrack DB cluster (Esegui backtrack del cluster database).
5. Nella pagina Backtrack DB cluster (Esegui backtrack del cluster database), inserire il timestamp di backtrack a cui eseguire il backtrack del cluster database.



6. Scegliere Backtrack DB cluster (Esegui backtrack del cluster database).

AWS CLI

La procedura seguente descrive il modo in cui eseguire il backtrack di un cluster database mediante l AWS CLI.

Per eseguire il backtrack di un cluster DB utilizzando il AWS CLI

- Chiamate il backtrack-db-cluster AWS CLIcomando e fornite i seguenti valori:
  - `--db-cluster-identifier`: il nome del cluster di database.
  - `--backtrack-to`— Il timestamp di backtrack verso cui eseguire il backtrack del cluster DB, specificato nel ISO formato 8601.

L'esempio seguente esegue il backtrack del cluster database `sample-cluster` con il timestamp 19 marzo 2018, ore 10.

PerLinux, o: macOS Unix

```
aws rds backtrack-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-to 2018-03-19T10:00:00+00:00
```

Per Windows:

```
aws rds backtrack-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-to 2018-03-19T10:00:00+00:00
```

RDS API

Per eseguire il backtrack di un cluster DB utilizzando Amazon RDSAPI, utilizza l'acktrackDBClusteroperazione B. Questa operazione esegue il backtrack del cluster database specificato nel valore `DBClusterIdentifier` all'orario specificato.

Monitoraggio del backtracking per un cluster Aurora My DB SQL

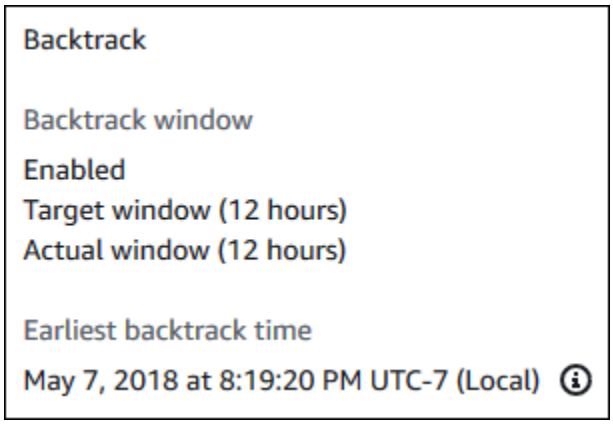
Puoi visualizzare le informazioni relative al backtrack e monitorare i parametri di backtrack per un cluster database.

Console

Per visualizzare le informazioni relative al backtrack e monitorare i parametri di backtrack mediante la console

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il nome del cluster database di cui si intende visualizzare le informazioni.

Le informazioni relative al backtrack sono visualizzate nella sezione Backtrack.



Quando il backtrack è abilitato, sono disponibili le seguenti informazioni:

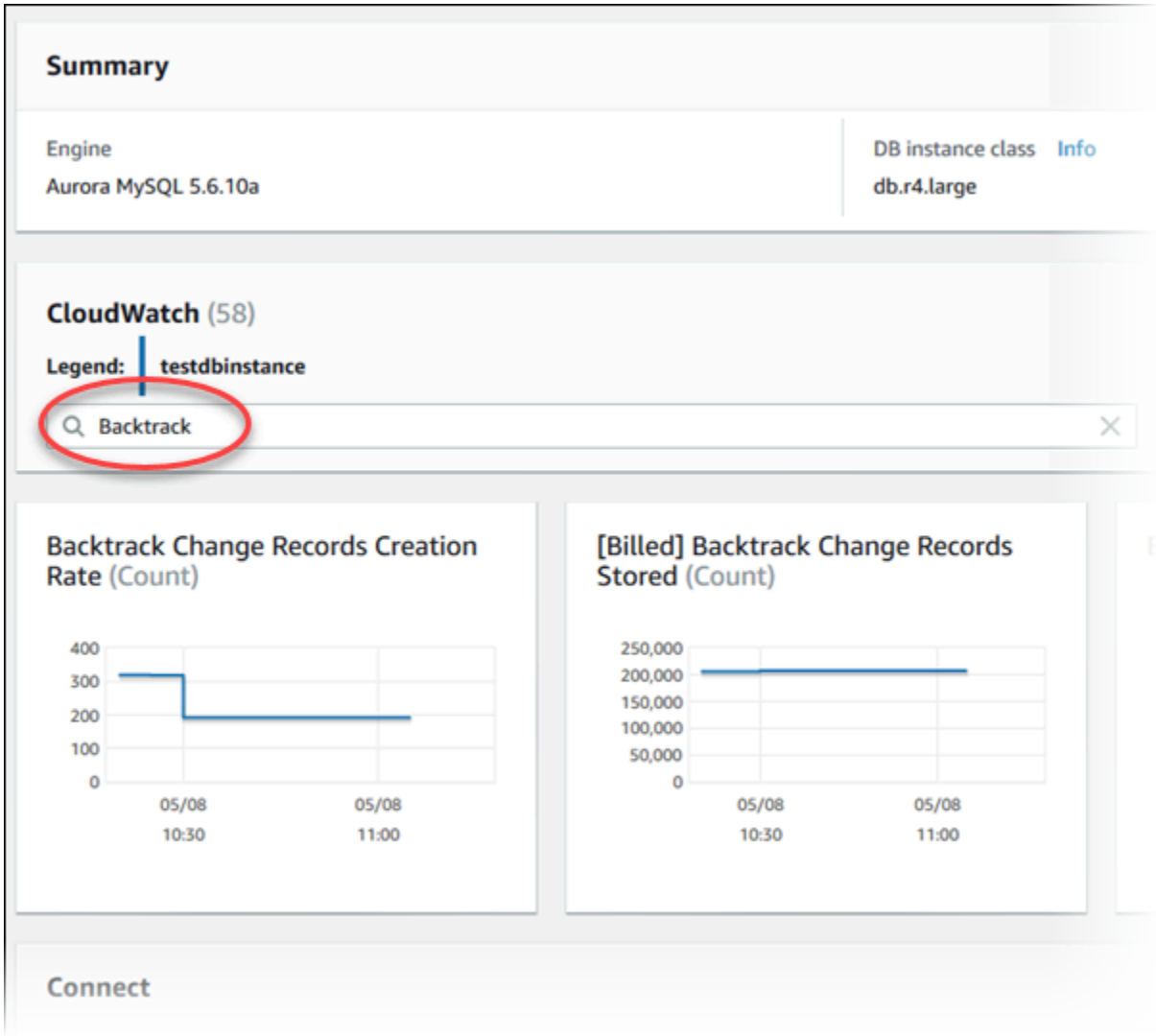
- Target window (Finestra target):– il periodo di tempo corrente specificato per la finestra di backtrack target. Il target è il periodo di tempo massimo durante il quale è possibile effettuare il backtrack se lo storage è sufficiente.
- Actual window (Finestra effettiva):– il periodo di tempo effettivo durante il quale è possibile eseguire il backtrack, che può essere inferiore alla finestra di backtrack target. La finestra di backtrack effettiva è basata sul carico di lavoro e sullo storage disponibile per conservare i record di modifica del backtrack.
- Earliest backtrack time (Primo orario di backtrack): –specifica il primo orario di backtrack possibile per il cluster database. Non è possibile eseguire il backtrack del cluster database a un orario antecedente a quello visualizzato.

4. Per visualizzare i parametri di backtrack per il cluster database, procedere come segue:

   a. Nel riquadro di navigazione, scegliere Instances (Istanze).

   b. Scegliere il nome dell'istanza principale per il cluster database di cui si intende visualizzarne i dettagli.

   c. Nella CloudWatchsezione, digita **`Backtrack`** nella CloudWatchcasella per mostrare solo le metriche di Backtrack.



Sono visualizzati i seguenti parametri:

- Backtrack Change Records Creation Rate (Count) (Frequenza di creazione record modifiche backtrack (numero)):– questo parametro visualizza il numero di record di modifica di backtrack creati in un intervallo di cinque minuti per il cluster database. È possibile utilizzare questo parametro per stimare il costo di backtrack per la finestra di backtrack target.
- [Billed] Backtrack Change Records Stored (Count) ([Fatturato] Record modifiche backtrack archiviati (numero)):– questo parametro visualizza il numero effettivo di record di modifica di backtrack utilizzati dal cluster di database.
- Backtrack Window Actual (Minutes) (Finestra di backtrack effettiva) (Minuti): –questo parametro indica se c'è una differenza tra la finestra di backtrack target e la finestra di backtrack effettiva. Ad esempio, se la finestra di backtrack target è 2 ore (120 minuti) e

questo parametro indica che la finestra di backtrack effettiva è 100 minuti, quest'ultima è più piccola della finestra target.

- Backtrack Window Alert (Count) (Avviso finestra di backtrack (numero)): –questo parametro indica quante volte la finestra di backtrack effettiva è inferiore alla finestra di backtrack target per un determinato periodo di tempo.

> **Note**
>
> I parametri seguenti potrebbero essere indietro rispetto all'orario corrente:
>
> - Backtrack Change Records Creation Rate (Count) (Frequenza di creazione record modifiche backtrack (numero))
> - [Billed] Backtrack Change Records Stored (Count) ([Fatturato] Record modifiche backtrack archiviati (numero))

AWS CLI

La procedura seguente descrive come visualizzare informazioni relative al backtrack per un cluster di database mediante l AWS CLI.

Per visualizzare le informazioni di backtrack per un cluster DB utilizzando il AWS CLI

- Chiamate il describe-db-clusters AWS CLIcomando e fornite i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.

  L'esempio seguente elenca le informazioni relative al backtrack per `sample-cluster`.

  Per LinuxmacOS, oUnix:

```
aws rds describe-db-clusters \
    --db-cluster-identifier sample-cluster
```

  Per Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier sample-cluster
```

RDS API

Per visualizzare le informazioni di backtrack per un cluster DB utilizzando Amazon RDSAPI, utilizza l'escribeDBClustersoperazione D. Questa operazione restituisce le informazioni relative al backtrack per il cluster database specificato nel valore `DBClusterIdentifier`.

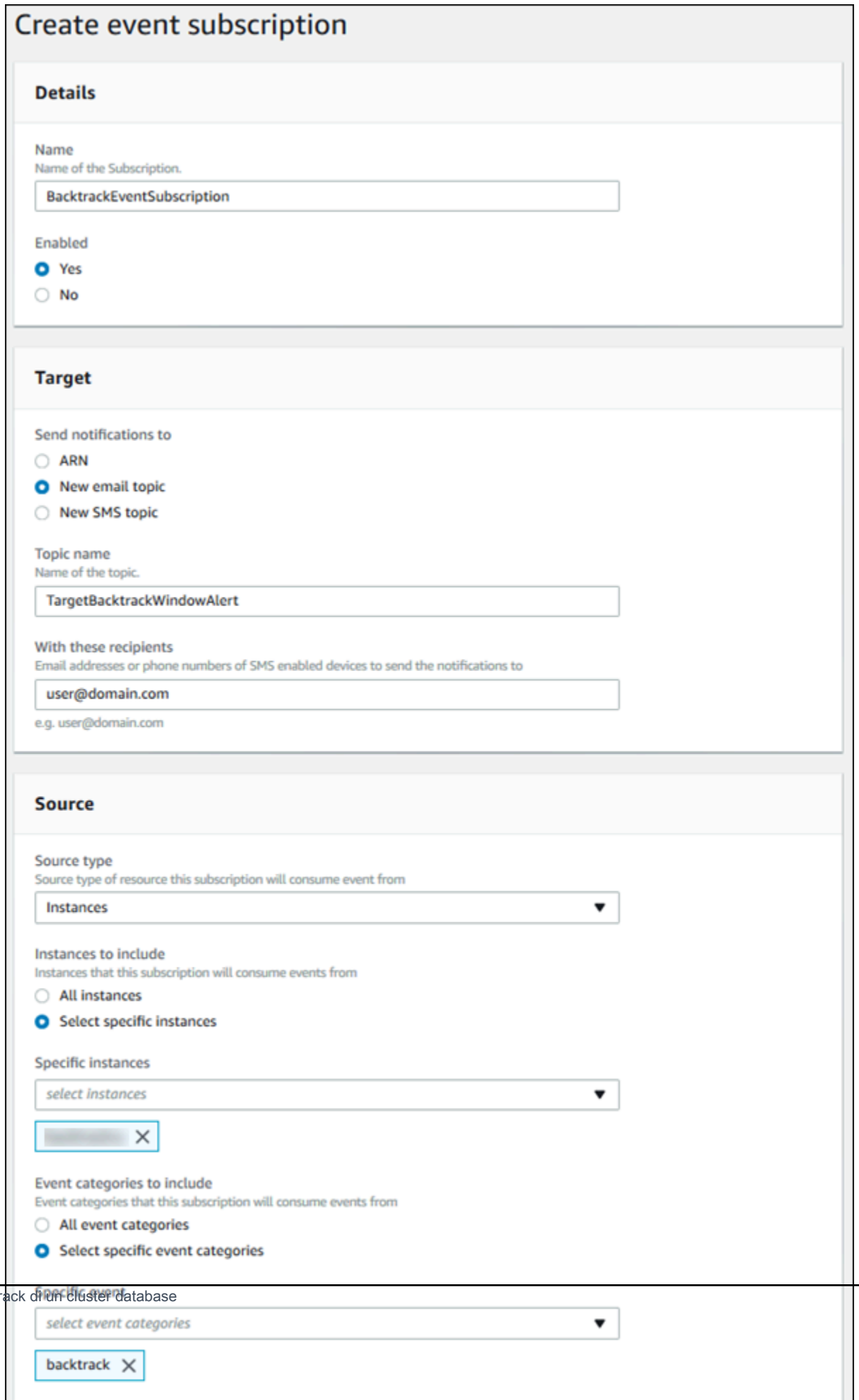
Abbonamento a un evento di backtrack con la console

La procedura seguente descrive come abbonarsi a un evento di backtrack mediante la console. L'evento ti invia un'email o una notifica di testo quando la finestra di backtrack effettiva è più piccola della finestra di backtrack target.

Per visualizzare le informazioni di backtrack mediante la console

1. Accedi al AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Event subscriptions (Abbonamenti a eventi).
3. Scegliere Create event subscription (Crea abbonamento a eventi).
4. Nella casella Name (Nome), digitare un nome per l'abbonamento all'evento e assicurarsi che Yes (Sì) è selezionato per Enabled (Abilitato).
5. Nella sezione Target, scegliere New email topic (Nuovo argomento e-mail).
6. In Topic name (Nome argomento), digitare un nome per l'argomento e in With these recipients (Con questi destinatari), digitare gli indirizzi e-mail o i numeri di telefono per ricevere le notifiche.
7. Nella sezione Source (origine), scegliere Instances (Istanze) per Source type (Tipo origine).
8. Per Instances to include (Istanze da includere), scegliere Select specific instances (Seleziona specifiche istanze) e scegliere l'istanza database.
9. In Event categories to include (Categorie di eventi da includere), scegliere Select specific event categories (Seleziona specifiche categorie di eventi), quindi scegliere Backtrack.

   La pagina dovrebbe essere simile alla seguente.


Create event subscription
Details
Name
Name of the Subscription.
BacktrackEventSubscription
Enabled
Yes
No
Target
Send notifications to
ARN
New email topic
New SMS topic
Topic name
Name of the topic.
TargetBacktrackWindowAlert
With these recipients
Email addresses or phone numbers of SMS enabled devices to send the notifications to
user@domain.com
e.g. user@domain.com
Source
Source type
Source type of resource this subscription will consume event from
Instances
Instances to include
Instances that this subscription will consume events from
All instances
Select specific instances
Specific instances
select instances
Event categories to include
Event categories that this subscription will consume events from
All event categories
Select specific event categories
select event categories
backtrack

10. Scegliere Create (Crea).

Recupero di backtrack esistenti

Puoi recuperare informazioni relative a backtrack esistenti per un cluster di database. Queste informazioni includono l'identificatore univoco del backtrack, la data e l'ora di inizio e di fine del backtrack, la data e l'ora in cui il backtrack è stato richiesto e lo stato corrente del backtrack.

> **Note**
>
> Non è attualmente possibile recuperare backtrack esistenti mediante la console.

AWS CLI

La procedura seguente descrive come recuperare i backtrack esistenti per un cluster DB utilizzando AWS CLI.

Per recuperare i backtrack esistenti utilizzando il AWS CLI

- Chiama il describe-db-cluster-backtracks AWS CLIcomanda e fornisci i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.

  L'esempio seguente recupera i backtrack esistenti per `sample-cluster`.

  In Linux, macOS, oppure Unix:

```
aws rds describe-db-cluster-backtracks \
    --db-cluster-identifier sample-cluster
```

  In Windows:

```
aws rds describe-db-cluster-backtracks ^
    --db-cluster-identifier sample-cluster
```

RDS API

Per recuperare informazioni sui backtrack per un cluster DB utilizzando Amazon RDSAPI, utilizza l'operazione D escribeDBCluster Backtracks. Questa operazione restituisce le informazioni relative ai backtrack per il cluster database specificato nel valore `DBClusterIdentifier`.

Disabilitazione del backtracking per un cluster Aurora My DB SQL

Puoi disabilitare la funzionalità di backtrack per un cluster di database.

Console

Puoi disabilitare il backtrack per un cluster di database mediante la console. Dopo aver disattivato completamente il backtracking per un cluster, non sarà possibile attivarlo di nuovo per tale cluster.

Per disabilitare la funzionalità di backtrack per un cluster di database mediante la console

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il cluster da modificare, quindi scegliere Modify (Modifica).
4. Nella sezione Backtrack, scegliere Disable Backtrack (Disabilita backtrack).
5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere una delle seguenti opzioni:
   - Apply during the next scheduled maintenance window (Applica durante la prossima finestra di manutenzione pianificata):– attendere la finestra di manutenzione successiva per applicare la modifica.
   - Apply immediately (Applica immediatamente):– applicare la modifica appena possibile.
7. Scegliere Modify Cluster (Modifica cluster).

AWS CLI

Puoi disabilitare la funzionalità Backtrack per un cluster DB utilizzando AWS CLI impostando la finestra di backtrack di destinazione su `0` (zero). Dopo aver disattivato completamente il backtracking per un cluster, non sarà possibile attivarlo di nuovo per tale cluster.

Per modificare la finestra di backtrack di destinazione per un cluster DB utilizzando il AWS CLI

- Chiamate il modify-db-cluster AWS CLIcomando e fornite i seguenti valori:
  - `--db-cluster-identifier`:– il nome del cluster database.
  - `--backtrack-window` – specifica `0` per disattivare il backtracking.

  L'esempio seguente disabilita la funzionalità di backtrack per `sample-cluster` impostando `--backtrack-window` su `0`.

  Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 0
```

  Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 0
```

RDS API

Per disabilitare la funzionalità Backtrack per un cluster DB utilizzando Amazon RDSAPI, utilizza l'odifyDBClusteroperazione M. Imposta il valore `BacktrackWindow` su `0` (zero) e specifica il cluster di database nel valore `DBClusterIdentifier`. Dopo aver disattivato completamente il backtracking per un cluster, non sarà possibile attivarlo di nuovo per tale cluster.

Test di Amazon Aurora My SQL using Fault Injection

È possibile testare la tolleranza agli errori del cluster Aurora My SQL DB utilizzando query di fault injection. Le query di Fault Injection vengono inviate come SQL comandi a un'istanza di Amazon Aurora. Queste funzionalità ti consentono di pianificare l'occorrenza simulata di uno dei seguenti eventi:

- Un arresto anomalo di un'istanza database di lettura o di scrittura

- Un errore di una replica Aurora
- Un errore del disco
- Una congestione del disco

Quando una query di fault injection specifica un arresto anomalo, impone un arresto anomalo dell'istanza Aurora SQL My DB. Le altre query fault injection generano simulazioni di eventi di errore, ma non causano l'evento. Quando invii una query fault injection, specifichi anche la durata della simulazione dell'evento di errore.

Puoi inviare una query fault injection a una delle istanze di replica Aurora eseguendo la connessione all'endpoint per la replica Aurora. Per ulteriori informazioni, consulta Connessioni endpoint Amazon Aurora.

L'esecuzione di query fault injection richiede tutti i privilegi dell'utente master. Per ulteriori informazioni, consulta Privilegi dell'account utente master.

Test dell'arresto anomalo di un'istanza

Puoi forzare un arresto anomalo di un'istanza Amazon Aurora utilizzando la query fault injection `ALTER SYSTEM CRASH`.

Per questa query fault injection, non si verificherà un failover. Se desideri testare un failover, puoi scegliere l'azione dell'istanza di failover per il tuo cluster DB nella RDS console oppure utilizzare il failover-db-clusterAWS CLI comando o l'operazione F. ailoverDBCluster RDS API

Sintassi

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Opzioni

Questa query fault injection accetta uno dei seguenti tipi di arresto anomalo:

- **`INSTANCE`**— Viene simulato un arresto anomalo del database SQL compatibile con My per l'istanza Amazon Aurora.
- **`DISPATCHER`** — Viene simulato un arresto anomalo del dispatcher sull'istanza di scrittura per il cluster database di Aurora. Il dispatcher scrive gli aggiornamenti sul volume del cluster per un cluster di database Amazon Aurora.

- **`NODE`**— Viene simulato un arresto anomalo sia del database SQL compatibile con My che del dispatcher per l'istanza Amazon Aurora. Per questa simulazione fault injection, viene inoltre eliminata la cache.

Il tipo di arresto anomalo predefinito è `INSTANCE`.

Test di un errore di replica Aurora

Puoi simulare l'errore di una replica Aurora mediante la query fault injection `ALTER SYSTEM SIMULATE READ REPLICA FAILURE`.

Un errore di replica Aurora blocca tutte le richieste da un'istanza di scrittura a una replica Aurora o tutte le repliche Aurora nel cluster di database per un periodo di tempo specificato. Al termine dell'intervallo di tempo, le repliche Aurora interessate verranno automaticamente sincronizzate con l'istanza del writer.

Sintassi

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE
    [ TO ALL | TO "replica name" ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

- **`percentage_of_failure`** — La percentuale di richieste da bloccare durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, non viene bloccata alcuna richiesta. Se specifichi 100, vengono bloccate tutte le richieste.
- Failure type (Tipo di errore) — Il tipo di errore da simulare. Specifica `TO ALL` per simulare errori per tutte le repliche Aurora nel cluster di database. Specifica `TO` e il nome della replica Aurora per simulare un errore di un'unica replica Aurora. Il tipo di errore predefinito è `TO ALL`.
- **`quantity`** — L'intervallo di tempo durante il quale simulare l'errore di replica di Aurora. L'intervallo è un numero seguito da un'unità di tempo. Il valore nell'unità specificata indica la durata della simulazione. Ad esempio, con `20 MINUTE` la simulazione durerà 20 minuti.

> **Note**
>
> Fai attenzione quando specifichi l'intervallo di tempo per l'evento di errore della replica Aurora. Se specifichi un intervallo di tempo troppo lungo e l'istanza master scrive una grande quantità di dati durante l'evento di errore, il cluster di database Aurora potrebbe ritenere che si è verificato un arresto anomalo della replica Aurora e quindi sostituirla.

Test di un errore del disco

Puoi simulare un errore del disco per un cluster database Aurora mediante la query fault injection `ALTER SYSTEM SIMULATE DISK FAILURE`.

Durante la simulazione di un errore del disco, il cluster di database Aurora contrassegna in modo aleatorio i segmenti del disco come difettosi. Le richieste a tali segmenti saranno bloccate per la durata della simulazione.

Sintassi

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

- **`percentage_of_failure`** — la percentuale del disco da contrassegnare come difettosa durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come difettosa. Se specifichi 100, tutto il disco è contrassegnato come difettoso.
- **`DISK index`** — uno specifico blocco di dati logico per il quale simulare l'evento di errore. Se viene superato l'intervallo di blocchi di dati logici disponibili, viene generato un errore indicante il valore di indice massimo che puoi specificare. Per ulteriori informazioni, consulta Visualizzazione dello stato del volume per un cluster DB Aurora MySQL.
- **`NODE index`** — uno specifico nodo di storage per il quale simulare l'evento di errore. Se viene superato l'intervallo di nodi di storage disponibili, viene generato un errore indicante il valore di

indice massimo che puoi specificare. Per ulteriori informazioni, consulta Visualizzazione dello stato del volume per un cluster DB Aurora MySQL.

- **`quantity`** — l'intervallo di tempo durante il quale simulare l'errore del disco. L'intervallo è un numero seguito da un'unità di tempo. Il valore nell'unità specificata indica la durata della simulazione. Ad esempio, con `20 MINUTE` la simulazione durerà 20 minuti.

Test di una congestione del disco

Puoi simulare un errore del disco per un cluster database Aurora mediante la query fault injection `ALTER SYSTEM SIMULATE DISK CONGESTION`.

Durante la simulazione della congestione del disco, il cluster di database Aurora contrassegna in modo aleatorio i segmenti del disco come congestionati. Le richieste a tali segmenti saranno ritardate di un periodo di tempo compreso tra il ritardo minimo e quello massimo specificati per la durata della simulazione.

Sintassi

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION
    BETWEEN minimum AND maximum MILLISECONDS
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

- **`percentage_of_failure`** — La percentuale del disco da contrassegnare come congestionata durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come congestionata. Se specifichi 100, tutto il disco è contrassegnato come congestionato.
- **`DISK index`** o **`NODE index`** — Uno specifico disco o nodo per il quale simulare l'evento di errore. Se viene superato l'intervallo di indici per il disco o il nodo, viene generato un errore indicante il valore di indice massimo che puoi specificare.
- **`minimum`** e **`maximum`** — Il valore minimo e massimo del ritardo di congestione in millisecondi. I segmenti del disco contrassegnati come congestionati saranno ritardati per un periodo di tempo

aleatorio compreso tra il valore minimo e quello massimo in millisecondi per la durata della simulazione.

- **`quantity`** — L'intervallo di tempo durante il quale simulare la congestione del disco. L'intervallo è un numero seguito da un'unità di tempo. Il valore nell'unità specificata indica la durata della simulazione. Ad esempio, con `20 MINUTE` la simulazione durerà 20 minuti.

Alterazione delle tabelle in Amazon Aurora mediante Fast DDL

Amazon Aurora include ottimizzazioni per eseguire un'operazione `ALTER TABLE` in atto, quasi istantaneamente. L'operazione viene eseguita senza la copia della tabella e senza impatto materiale sulle altre istruzioni DML. Poiché l'operazione non consuma storage temporaneo per la copia della tabella, rende le istruzioni DDL pratiche anche nel caso di tabelle di grandi dimensioni su tipi di classi small.

Aurora MySQL versione 3 è compatibile con la funzionalità MySQL 8.0 chiamata Instant DDL. Aurora MySQL versione 2 utilizza un'implementazione diversa chiamata Fast DDL.

Argomenti

- DDL istantaneo (Aurora MySQL versione 3)
- Fast DDL (Aurora MySQL versione 2)

DDL istantaneo (Aurora MySQL versione 3)

L'ottimizzazione eseguita da Aurora MySQL versione 3 per migliorare l'efficienza di alcune operazioni DDL è chiamata DDL istantaneo.

Aurora MySQL versione 3 è compatibile con il DDL istantaneo della community MySQL 8.0. Si esegue un'operazione DDL istantanea utilizzando la clausola `ALGORITHM=INSTANT` con l'istruzione `ALTER TABLE`. Per informazioni sulla sintassi e sull'utilizzo del DDL istantaneo, vedere ALTER TABLE e Online DDL Operations (Operazioni DDL online) nella documentazione MySQL.

I seguenti esempi dimostrano la funzionalità DDL immediata. Le istruzioni `ALTER TABLE` aggiungono colonne e modificano i valori di default delle colonne. Gli esempi includono colonne regolari e virtuali e tabelle regolari e partizionate. In ogni fase, è possibile visualizzare i risultati emettendo `SHOW CREATE TABLE` e istruzioni `DESCRIBE`.

```
mysql> CREATE TABLE t1 (a INT, b INT, KEY(b)) PARTITION BY KEY(b) PARTITIONS 6;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> ALTER TABLE t1 RENAME TO t2, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b SET DEFAULT 100, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b DROP DEFAULT, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN c ENUM('a', 'b', 'c'), ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 MODIFY COLUMN c ENUM('a', 'b', 'c', 'd', 'e'), ALGORITHM =
 INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN (d INT GENERATED ALWAYS AS (a + 1) VIRTUAL), ALGORITHM
 = INSTANT;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t2 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3 (a INT, b INT) PARTITION BY LIST(a)(
    ->   PARTITION mypart1 VALUES IN (1,3,5),
    ->   PARTITION MyPart2 VALUES IN (2,4,6)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE t3 ALTER COLUMN a SET DEFAULT 20, ALTER COLUMN b SET DEFAULT 200,
 ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t4 (a INT, b INT) PARTITION BY RANGE(a)
    ->   (PARTITION p0 VALUES LESS THAN(100), PARTITION p1 VALUES LESS THAN(1000),
    ->   PARTITION p2 VALUES LESS THAN MAXVALUE);
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE t4 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

```
/* Sub-partitioning example */
mysql> CREATE TABLE ts (id INT, purchased DATE, a INT, b INT)
    ->   PARTITION BY RANGE( YEAR(purchased) )
    ->     SUBPARTITION BY HASH( TO_DAYS(purchased) )
    ->     SUBPARTITIONS 2 (
    ->       PARTITION p0 VALUES LESS THAN (1990),
    ->       PARTITION p1 VALUES LESS THAN (2000),
    ->       PARTITION p2 VALUES LESS THAN MAXVALUE
    ->    );
Query OK, 0 rows affected (0.10 sec)

mysql> ALTER TABLE ts ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

Fast DDL (Aurora MySQL versione 2)

Fast DDL di Aurora MySQL è un'ottimizzazione progettata per migliorare le prestazioni di determinate modifiche allo schema, come l'aggiunta o l'eliminazione di colonne, riducendo i tempi di inattività e l'utilizzo delle risorse. Consente di completare queste operazioni in modo più efficiente rispetto ai metodi DDL tradizionali.

> **Important**
>
> Attualmente, è necessario abilitare la modalità laboratorio Aurora per utilizzare Fast DDL. Per informazioni sull'attivazione della modalità lab, consulta. Modalità di laboratorio per Amazon Aurora MySQL
>
> L'ottimizzazione Fast DDL è stata inizialmente introdotta in modalità lab su Aurora MySQL versione 2 per migliorare l'efficienza di alcune operazioni DDL. Nella versione 3 di Aurora MySQL, la modalità lab è stata interrotta e Fast DDL è stata sostituita dalla funzionalità MySQL 8.0 Instant DDL.

In MySQL, molte operazioni DDL (Data Definition Language) hanno un impatto considerevole sulle prestazioni.

Ad esempio, supponiamo che utilizzi un'operazione `ALTER TABLE` per aggiungere una colonna a una tabella. A seconda dell'algoritmo specificato per l'operazione, questa può comportare quanto segue:

- Creazione di una copia completa della tabella
- Creazione di una tabella temporanea per elaborare operazioni DML (Data Manipulation Language) simultanee
- Ricostruzione di tutti gli indici per la tabella
- Applicazione dei blocchi di tabelle durante l'applicazione di modifiche DML simultanee
- Rallentamento del throughput DML simultaneo

Questo impatto sulle prestazioni può essere particolarmente impegnativo in ambienti con tabelle di grandi dimensioni o volumi di transazioni elevati. Fast DDL aiuta a mitigare queste sfide ottimizzando le modifiche allo schema, il che consente operazioni più rapide e meno dispendiose in termini di risorse.

Limiti di Fast DDL

Fast DDL presenta attualmente le seguenti limitazioni:

- Supporta solo l'aggiunta di una colonna nullable, senza valori predefiniti, alla fine di una tabella esistente.
- La DDL veloce non funziona per le tabelle partizionate.
- DDL rapida non funziona per le tabella InnoDB che utilizzano il formato riga REDUNDANT.
- La DDL veloce non funziona per le tabelle con indici di ricerca full-text.
- Se la dimensione di record massima possibile per l'operazione DDL è troppo grande, Fast DDL non viene utilizzato. Una dimensione di record è troppo importante se è più grande della metà della dimensione della pagina. La dimensione massima di un record viene calcolata sommando la dimensione massima di ogni colonna. Per le colonne di dimensione variabile, conformemente agli standard InnoDB, i byte extern non sono inclusi nel calcolo.

Sintassi di Fast DDL

```
ALTER TABLE tbl_name ADD COLUMN col_name column_definition
```

Questa istruzione accetta le seguenti opzioni:

- **`tbl_name`** — il nome della tabella da modificare.
- **`col_name`** — il nome della colonna da aggiungere.

- **`col_definition`** — la definizione della colonna da aggiungere.

> **Note**
>
> Devi specificare una definizione di colonna nullable senza un valore predefinito. In caso contrario, Fast DDL non viene utilizzato.

Esempi di Fast DDL

Negli esempi seguenti viene illustrata la velocità delle operazioni Fast DDL. Il primo esempio SQL esegue istruzioni `ALTER TABLE` su una tabella di grandi dimensioni senza utilizzare Fast DDL. Questa operazione richiede molto tempo. Un esempio della CLI mostra come abilitare Fast DDL per il cluster. Quindi un altro esempio SQL esegue le stesse istruzioni `ALTER TABLE` su una tabella identica. Con Fast DDL abilitato, l'operazione è molto veloce.

In questo esempio viene utilizzata la tabella `ORDERS` del benchmark TPC-H, contenente 150 milioni di righe. Questo cluster utilizza intenzionalmente una classe di istanza relativamente piccola, per dimostrare quanto tempo possano richiedere le istruzioni `ALTER TABLE` quando non è possibile utilizzare Fast DDL. Nell'esempio viene creato un clone della tabella originale contenente dati identici. Il controllo dell'impostazione `aurora_lab_mode` conferma che il cluster non può utilizzare Fast DDL poiché la modalità lab non è abilitata. Quindi le istruzioni `ALTER TABLE ADD COLUMN` richiedono molto tempo per aggiungere nuove colonne alla fine della tabella.

```
mysql> create table orders_regular_ddl like orders;
Query OK, 0 rows affected (0.06 sec)

mysql> insert into orders_regular_ddl select * from orders;
Query OK, 150000000 rows affected (1 hour 1 min 25.46 sec)

mysql> select @@aurora_lab_mode;
+------------------+
| @@aurora_lab_mode |
+------------------+
|                0 |
+------------------+

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (40 min 31.41 sec)

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_coverletter varchar(512);
```

```
Query OK, 0 rows affected (40 min 44.45 sec)
```

Questo esempio esegue la stessa preparazione di una tabella di grandi dimensioni dell'esempio precedente. Tuttavia, non è possibile abilitare semplicemente la modalità lab all'interno di una sessione SQL interattiva. Tale impostazione deve essere abilitata in un gruppo di parametri personalizzato. A tale scopo è necessario chiudere la `mysql` sessione ed eseguire alcuni comandi AWS CLI o utilizzare il. AWS Management Console

```
mysql> create table orders_fast_ddl like orders;
Query OK, 0 rows affected (0.02 sec)

mysql> insert into orders_fast_ddl select * from orders;
Query OK, 150000000 rows affected (58 min 3.25 sec)

mysql> set aurora_lab_mode=1;
ERROR 1238 (HY000): Variable 'aurora_lab_mode' is a read only variable
```

Per abilitare la modalità lab per il cluster è necessario lavorare con un gruppo di parametri. Questo esempio di AWS CLI utilizza un gruppo di parametri del cluster, per garantire che tutte le istanze DB nel cluster utilizzino lo stesso valore per l'impostazione della modalità lab.

```
$ aws rds create-db-cluster-parameter-group \
  --db-parameter-group-family aurora5.7 \
    --db-cluster-parameter-group-name lab-mode-enabled-57 --description 'TBD'
$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].[ParameterName,ParameterValue]' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 0
$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --parameters ParameterName=aurora_lab_mode,ParameterValue=1,ApplyMethod=pending-
reboot
{
    "DBClusterParameterGroupName": "lab-mode-enabled-57"
}

# Assign the custom parameter group to the cluster that's going to use Fast DDL.
$ aws rds modify-db-cluster --db-cluster-identifier tpch100g \
  --db-cluster-parameter-group-name lab-mode-enabled-57
{
  "DBClusterIdentifier": "tpch100g",
```

```
  "DBClusterParameterGroup": "lab-mode-enabled-57",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.2",
  "Status": "available"
}

# Reboot the primary instance for the cluster tpch100g:
$ aws rds reboot-db-instance --db-instance-identifier instance-2020-12-22-5208
{
  "DBInstanceIdentifier": "instance-2020-12-22-5208",
  "DBInstanceStatus": "rebooting"
}

$ aws rds describe-db-clusters --db-cluster-identifier tpch100g \
  --query '*[].[DBClusterParameterGroup]' --output text
lab-mode-enabled-57

$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 1
```

Nell'esempio seguente vengono illustrati i passaggi rimanenti dopo che la modifica del gruppo di parametri ha effetto. Si verifica l'impostazione `aurora_lab_mode` per assicurarsi che il cluster possa utilizzare Fast DDL. Quindi viene eseguita l'istruzione `ALTER TABLE` per aggiungere colonne alla fine di un'altra tabella di grandi dimensioni. Questa volta, le istruzioni vengono completate molto rapidamente.

```
mysql> select @@aurora_lab_mode;
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 1 |
+-------------------+

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (1.51 sec)

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (0.40 sec)
```

Visualizzazione dello stato del volume per un cluster DB Aurora MySQL

In Amazon Aurora, un volume di cluster di database è costituito da una raccolta di blocchi logici. Ognuno di questi rappresenta 10 gigabyte di storage allocato. Questi blocchi sono denominati gruppi di protezione.

I dati in ogni gruppo di protezione sono replicati su sei dispositivi di storage fisico denominati nodi di storage. Questi nodi sono allocati su tre zone di disponibilità (AZ) nella regione AWS in cui si trova il cluster DB. Ogni nodo di storage contiene a sua volta uno o più blocchi di dati logici per il volume di cluster di database. Per ulteriori informazioni sui gruppi di protezione e i nodi di storage, consulta Introduzione del motore di storage Aurora nel blog AWS Database.

Puoi simulare l'errore di un intero nodo di storage o di un singolo blocco di dati logico in un nodo di storage. A questo proposito, devi utilizzare l'istruzione fault injection `ALTER SYSTEM SIMULATE DISK FAILURE`. Per l'istruzione, specifichi il valore di indice di uno specifico blocco di dati logico o nodo di storage. Tuttavia, se specifichi un valore di indice superiore al numero di blocchi di dati logici o nodi di storage utilizzati dal volume di cluster di database, l'istruzione restituisce un errore. Per ulteriori informazioni sulle query fault injection, consulta Test di Amazon Aurora My SQL using Fault Injection.

Puoi evitare tale errore utilizzando l'istruzione `SHOW VOLUME STATUS`. L'istruzione restituisce due variabili di stato del server, `Disks` e `Nodes`. Queste variabili rappresentano rispettivamente il numero totale di blocchi di dati logici e di nodi di storage per il volume di cluster di database.

Sintassi

```
SHOW VOLUME STATUS
```

Esempio

Il seguente esempio illustra un risultato SHOW VOLUME STATUS tipico.

```
mysql> SHOW VOLUME STATUS;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Disks         | 96    |
| Nodes         | 74    |
```

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

Ottimizzazione di Aurora MySQL

Gli eventi di attesa e gli stati di thread sono un importante strumento di ottimizzazione per Aurora MySQL. Se si riesce a scoprire perché le sessioni stanno aspettando risorse e cosa stanno facendo, è possibile ridurre meglio i colli di bottiglia. È possibile utilizzare le informazioni contenute in questa sezione per trovare possibili cause e azioni correttive.

Amazon DevOps Guru per RDS è in grado di determinare in modo proattivo se i database Aurora MySQL presentano condizioni problematiche che potrebbero causare problemi più seri in seguito. Amazon DevOps Guru per RDS pubblica una spiegazione e i suggerimenti per le azioni correttive negli approfondimenti proattivi. Questa sezione contiene gli approfondimenti per i problemi più comuni.

> **Important**
>
> Gli eventi di attesa e gli stati del thread in questa sezione sono specifici di Aurora MySQL. Utilizza le informazioni contenute in questa sezione per regolare solo Amazon Aurora, non Amazon RDS for MySQL.
> Alcuni eventi di attesa in questa sezione non hanno analoghi nelle versioni open source di questi motori di database. Altri eventi di attesa hanno gli stessi nomi degli eventi nei motori open source, ma si comportano in modo diverso. Ad esempio, l'archiviazione di Amazon Aurora funziona in modo diverso dall'archiviazione open source, pertanto gli eventi di attesa correlati all'archiviazione indicano condizioni di risorse differenti.

Argomenti

- Concetti essenziali per la regolazione di Aurora MySQL
- Regolazione di Aurora MySQL con eventi di attesa
- Regolazione di Aurora MySQL con stati del thread
- Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru

Concetti essenziali per la regolazione di Aurora MySQL

Prima di regolare il database di Aurora MySQL, assicurati di sapere quali sono gli eventi di attesa e gli stati del thread e perché si verificano. Esaminare anche l'architettura di base della memoria e del disco di Aurora MySQL quando si utilizza il motore di archiviazione InnoDB. Per un utile diagramma architettonico, consulta ilManuale di riferimento MySQL.

Argomenti

- Eventi di attesa di Aurora MySQL
- Stati del thread di Aurora MySQL
- Memoria di Aurora MySQL
- Procedure di Aurora MySQL

Eventi di attesa di Aurora MySQL

Un evento di attesa indica una risorsa per la quale è in attesa una sessione. Ad esempio, l'evento di attesa `io/socket/sql/client_connection` indica che un thread sta per gestire una nuova connessione. Le risorse tipiche che una sessione attende includono quanto segue:

- Accesso a thread singolo a un buffer, ad esempio, quando una sessione sta tentando di modificare un buffer
- Una riga attualmente bloccata da un'altra sessione
- Un file di dati letto
- Scrittura di un file log

Ad esempio, per soddisfare una query, la sessione potrebbe eseguire una scansione completa della tabella. Se i dati non sono già in memoria, la sessione attende il completamento dell'I/O del disco. Quando i buffer vengono letti in memoria, potrebbe essere necessario attendere perché altre sessioni accedono agli stessi buffer. Il database registra le attese utilizzando un evento di attesa predefinito. Questi eventi sono raggruppati in categorie.

Un evento di attesa non mostra di per sé un problema di prestazioni. Ad esempio, se i dati richiesti non sono in memoria, è necessaria la lettura dei dati dal disco. Se una sessione blocca una riga per un aggiornamento, un'altra sessione attende che la riga venga sbloccata in modo che possa aggiornarla. Un commit richiede di attendere il completamento della scrittura su un file di registro. Le attese sono parte integrante del normale funzionamento di un database.

Un gran numero di eventi di attesa in genere mostrano un problema di prestazioni. In questi casi, è possibile utilizzare i dati degli eventi di attesa per determinare dove stanno trascorrendo il tempo delle sessioni. Ad esempio, se un report che in genere viene eseguito in minuti ora viene eseguito per ore, è possibile identificare gli eventi di attesa che contribuiscono maggiormente al tempo di attesa totale. Se è possibile determinare le cause degli eventi di attesa principali, a volte è possibile apportare

modifiche che migliorano le prestazioni. Ad esempio, se la sessione è in attesa su una riga bloccata da un'altra sessione, è possibile terminare la sessione di blocco.

Stati del thread di Aurora MySQL

Uno stato generale del thread è un valore `State` associato all'elaborazione generale delle query. Ad esempio, lo stato del thread `sending data` indica che un thread sta leggendo e filtrando le righe per una query per determinare il set di risultati corretto.

Puoi usare gli stati del thread per regoalre Aurora MySQL in modo simile a come usi gli eventi di attesa. Ad esempio, frequenti occorrenze di `sending data` di solito indicano che una query non utilizza un indice. Per ulteriori informazioni sugli stati del thread, consulta Stati generali del thread nel Manuale di riferimento di MySQL.

Quando si utilizza Performance Insights, è vera una delle seguenti condizioni:

- Performance Schema è attivato: Aurora MySQL mostra gli eventi di attesa anziché lo stato del thread.
- Performance Schema non è attivato: Aurora MySQL mostra lo stato del thread.

Consigliamo di configurare il Performance Schema per la gestione automatica. Il Performance Schema fornisce ulteriori informazioni dettagliate e strumenti migliori per indagare sui potenziali problemi di prestazioni. Per ulteriori informazioni, consulta Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

Memoria di Aurora MySQL

In Aurora MySQL, le aree di memoria più importanti sono il pool di buffer e il buffer di registro.

Argomenti

- Pool di buffer

Pool di buffer

Il pool di buffer è l'area di memoria condivisa in cui Aurora MySQL memorizza nella cache i dati di tabella e indice. Le query possono accedere ai dati utilizzati di frequente direttamente dalla memoria senza leggere dal disco.

Il pool di buffer è strutturato come un elenco di pagine collegato. Una pagina può contenere più righe. Aurora MySQL utilizza un algoritmo utilizzato meno di recente (LRU) per invecchiare le pagine fuori dal pool.

Per ulteriori informazioni, consulta Buffer Pool (Pool di buffer) nel Manuale di riferimento di MySQL.

Procedure di Aurora MySQL

Aurora MySQL utilizza un modello di procedure molto diverso da Aurora PostgreSQL.

Argomenti

- Server MySQL (mysqld)
- Thread
- Pool di thread

Server MySQL (mysqld)

Il server MySQL è una procedura singola del sistema operativo denominato mysqld. Il server MySQL non genera procedure aggiuntive. Pertanto, un database di Aurora MySQL utilizza mysqld per eseguire la maggior parte del suo lavoro.

Quando si avvia il server MySQL, ascolta le connessioni di rete dai client MySQL. Quando un client si connette al database, mysqld apre un thread.

Thread

I thread del gestore delle connessioni associano ogni connessione client a un thread dedicato. Questo thread gestisce l'autenticazione, esegue istruzioni e restituisce i risultati al client. Il gestore delle connessioni crea nuovi thread quando necessario.

La cache dei thread è il set di thread disponibili. Quando una connessione termina, MySQL restituisce il thread alla cache del thread se la cache non è piena. La variabile del sistema `thread_cache_size` determina la dimensione della cache del thread.

Pool di thread

Il pool di thread è costituito da un certo numero di gruppi di thread. Ogni gruppo gestisce una serie di connessioni client. Quando un client si connette al database, il pool di thread assegna le connessioni

ai gruppi di thread in modo round-robin. Il pool di thread separa connessioni e thread. Non esiste una relazione fissa tra le connessioni e i thread che eseguono istruzioni ricevute da tali connessioni.

Regolazione di Aurora MySQL con eventi di attesa

La tabella seguente riassume gli eventi di attesa di Aurora MySQL che più comunemente indicano problemi di prestazioni. I seguenti eventi di attesa sono un sottoinsieme dell'elenco in Eventi Aurora My wait SQL.

Evento di attesa	Descrizione
cpu	Questo evento si verifica quando un thread è attivo nella CPU o è in attesa della CPU.
io/aurora_redo_log_flush	Questo evento si verifica quando una sessione sta scrivendo dati permanenti nell'arch iviazione di Aurora.
io/aurora_respond_to_client	Questo evento si verifica quando un thread è in attesa di restituire una serie di risultati a un client.
io/redo_log_flush	Questo evento si verifica quando una sessione sta scrivendo dati permanenti nell'arch iviazione di Aurora.
io/socket/sql/client_connessione	Questo evento si verifica quando un thread sta per gestire una nuova connessione.
io/table/sql/handler	Questo evento si verifica quando il lavoro è stato delegato a un motore di archiviazione.
synch/cond/innodb/row_blocca_aspetta	Questo evento si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga.
synch/cond/innodb/row_lock_wait_cond	Questo evento si verifica quando una sessione ha bloccato una riga per un aggiornamento e

Evento di attesa	Descrizione
	un'altra sessione tenta di aggiornare la stessa riga.
synch/cond/sql/MDL_contesto:: COND _wait_status	Questo evento si verifica quando ci sono thread in attesa del blocco dei metadati di una tabella.
synch/mutex/innodb/aurora_lock_thread_slot_futex	Questo evento si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga.
synch/mutex/innodb/buf_pool_mutex	Questo evento si verifica quando un thread ha acquisito un blocco nel pool di buffer InnoDB per accedere a una pagina in memoria.
synch/mutex/innodb/fil_sistema_mutex	Questo evento si verifica quando una sessione è in attesa di accedere alla cache di memoria del tablespace.
synch/mutex/innodb/trx_sys_mutex	Questo evento si verifica quando c'è un'elevata attività del database con un numero elevato di transazioni.
synch/sxlock/innodb/hash_serrature da tavolo	Questo evento si verifica quando le pagine non trovate nel pool del buffer devono essere lette da un file.

cpu

L'evento di `cpu` attesa si verifica quando un thread è attivo CPU o è in attesaCPU.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese

- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora SQL Le mie versioni 2 e 3

Context

Per ogni vCPU, una connessione può funzionare su questoCPU. In alcune situazioni, il numero di connessioni attive pronte per l'esecuzione è superiore al numero divCPUs. Questo squilibrio si traduce in connessioni in attesa di CPU risorse. Se il numero di connessioni attive rimane costantemente superiore al numero divCPUs, allora nell'istanza si verifica un CPU conflitto. La contesa provoca il verificarsi dell'evento di attesa `cpu`.

> **Note**
>
> La metrica Performance Insights per CPU è`DBLoadCPU`. Il valore di `DBLoadCPU` può differire dal valore della CloudWatch metrica`CPUUtilization`. Quest'ultima metrica viene raccolta da HyperVisor per un'istanza di database.

Le metriche del sistema operativo Performance Insights forniscono informazioni dettagliate sull'CPUutilizzo. Ad esempio, è possibile visualizzare i seguenti parametri:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Performance Insights riporta l'CPUutilizzo da parte del motore di database come`os.cpuUtilization.nice.avg`.

Probabili cause di aumento delle attese

Quando questo evento si verifica più del normale, eventualmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

- Query analitiche
- Transazioni altamente simultanee
- Transazioni di lunga durata
- Un improvviso aumento del numero di connessioni, noto come storm di login
- Un aumento del cambio di contesto

Azioni

Se l'evento di attesa `cpu` domina l'attività del database, non indica necessariamente un problema di prestazioni. Rispondi a questo evento solo quando le prestazioni diminuiscono.

A seconda della causa dell'aumento dell'CPUutilizzo, prendete in considerazione le seguenti strategie:

- Aumentare la CPU capacità dell'host. Questo approccio tipicamente offre un sollievo temporaneo.
- Identifica le principali query per una potenziale ottimizzazione.
- Reindirizza alcuni carichi di lavoro di sola lettura verso i nodi del lettore, se applicabile.

Argomenti

- Identifica le sessioni o le query che causano il problema
- Analizza e ottimizza l'elevato carico CPU di lavoro

Identifica le sessioni o le query che causano il problema

Per trovare le sessioni e le query, consulta la SQL tabella Top di Performance Insights per le SQL istruzioni con il CPU carico più elevato. Per ulteriori informazioni, consulta Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights.

In genere, una o due SQL istruzioni occupano la maggior parte dei CPU cicli. Concentra gli sforzi su queste istruzioni. Supponiamo che l'istanza DB ne abbia 2 vCPUs con un carico DB di 3,1 sessioni attive medie (AAS), tutte nello CPU stato. In questo caso, l'istanza è CPU vincolata. Considera le strategie seguenti:

- Esegui l'upgrade a una classe di istanze più grande con più istanzevCPUs.
- Ottimizza le tue query per ridurre il CPU carico.

In questo esempio, le SQL query principali hanno un carico in DB di 1,5AAS, tutte nello CPU stato. Un'altra SQL istruzione ha un carico di 0,1 nello CPU stato. In questo esempio, se si interrompe l'SQListruzione lowest-load, non si riduce in modo significativo il carico del database. Tuttavia, se si ottimizzano le due query ad alto carico per renderle due volte più efficienti, si elimina il collo di bottiglia. CPU Se si riduce il CPU carico dell'1,5 del 50 AAS percento, il valore AAS per ogni istruzione scende a 0,75. Il carico totale del DB impiegato CPU è ora 1,6AAS. Questo valore è inferiore alla CPU linea v massima di 2.0.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights. Consulta anche l'articolo di AWS supporto Come posso risolvere i problemi e risolvere l'CPUutilizzo elevato sulle mie istanze Amazon RDS for My? SQL .

Analizza e ottimizza l'elevato carico CPU di lavoro

Dopo aver identificato la o le query che aumentano l'CPUutilizzo, puoi ottimizzarle o terminare la connessione. L'esempio seguente mostra come terminare una connessione.

```
CALL mysql.rds_kill(processID);
```

Per ulteriori informazioni, consulta mysql.rds_kill.

Se si termina una sessione, l'azione potrebbe innescare un lungo ripristino dello stato precedente.

Seguire le linee guida per l'ottimizzazione delle query

Per ottimizzare le query, prendere in considerazione le seguenti linee guida:

- Eseguire l'istruzione `EXPLAIN`.

  Questo comando mostra i singoli passaggi coinvolti nell'esecuzione di una query. Per ulteriori informazioni, vedere Ottimizzazione delle interrogazioni con EXPLAIN nella documentazione personale. SQL
- Eseguire l'istruzione `SHOW PROFILE`.

  Utilizzare questa istruzione per esaminare i dettagli del profilo che possono indicare l'utilizzo delle risorse per le istruzioni eseguite durante la sessione corrente. Per ulteriori informazioni, vedi SHOWPROFILEDichiarazione nella mia SQL documentazione.
- Eseguire l'istruzione `ANALYZE TABLE`.

Utilizzate questa istruzione per aggiornare le statistiche dell'indice per le tabelle a cui accede la query ad alto CPU consumo. Analizzando l'istruzione, è possibile aiutare l'ottimizzatore a scegliere un piano di esecuzione appropriato. Per ulteriori informazioni, vedere ANALYZETABLEDichiarazione nella SQL documentazione personale.

Segui le linee guida per migliorare CPU l'utilizzo

Per migliorare CPU l'utilizzo in un'istanza di database, segui queste linee guida:

- Assicurarsi che tutte le query utilizzino indici appropriati.
- Scopri se è possibile utilizzare query parallele di Aurora. È possibile utilizzare questa tecnica per ridurre l'CPUutilizzo sul nodo principale riducendo l'elaborazione delle funzioni, il filtraggio delle righe e la proiezione delle colonne per la clausola. `WHERE`
- Scoprite se il numero di SQL esecuzioni al secondo soddisfa le soglie previste.
- Scopri se la manutenzione degli indici o la creazione di nuovi indici richiedono i CPU cicli necessari per il tuo carico di lavoro di produzione. Pianifica le attività di manutenzione al di fuori degli orari del picco di attività.
- Scopri se è possibile utilizzare il partizionamento per ridurre la serie dei dati delle query. Per ulteriori informazioni, consulta il post del blog Come pianificare e ottimizzare Amazon Aurora with My SQL compatibility for consolidated workload.

Verifica la presenza di congestioni dovute alla connessione

Se il `DBLoadCPU` parametro non è molto elevato, ma è elevato, la `CPUUtilization` causa dell'elevato CPU utilizzo non è riconducibile al motore di database. Un esempio classico è una congestione dovuta alla connessione.

Verificare se le condizioni seguenti sono vere:

- C'è un aumento sia della metrica Performance Insights che della `CPUUtilization` CloudWatch `DatabaseConnections` metrica Amazon.
- Il numero di thread in CPU è maggiore del numero di. vCPUs

Se le condizioni precedenti sono vere, considerare di diminuire il numero di connessioni al database. Ad esempio, è possibile utilizzare un pool di connessioni come RDS Proxy. Per conoscere le best

practice per una gestione e una scalabilità efficaci delle connessioni, consulta il white paper Amazon SQL DBA Aurora My Handbook for Connection Management.

io/aurora_redo_log_flush

L'evento `io/aurora_redo_log_flush` si verifica quando una sessione sta scrivendo dati persistenti nell'archiviazione di Amazon Aurora.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 2 SQL

Context

L'`io/aurora_redo_log_flush`evento riguarda un'operazione di scrittura input/output (I/O () in Aurora My. SQL

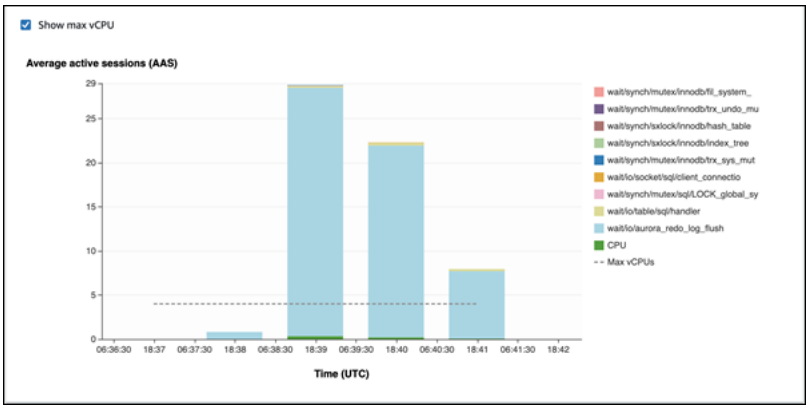
> **Note**
>
> In Aurora My SQL versione 3, questo evento di attesa è denominato io/redo_log_flush.

Probabili cause di aumento delle attese

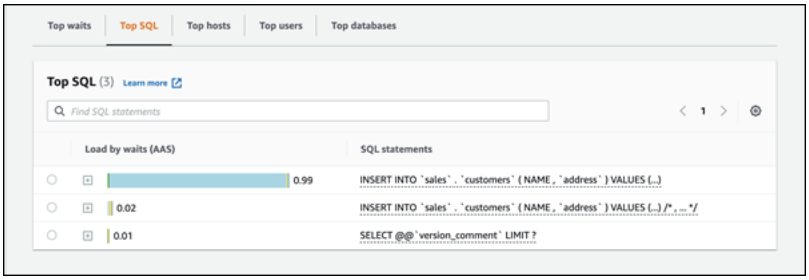
Per la persistenza dei dati, l'esecuzione dei commit richiede una scrittura duratura per un'archiviazione stabile. Se il database sta eseguendo troppi commit, c'è un evento di attesa sull'operazione di I/O di scrittura, l'evento di attesa `io/aurora_redo_log_flush`.

Negli esempi seguenti, 50.000 record vengono inseriti in un cluster Aurora SQL My DB utilizzando la classe di istanza DB db.r5.xlarge:

- Nel primo esempio, ogni sessione inserisce 10.000 dati riga per riga. Per impostazione predefinita, se un comando Data Manipulation Language (DML) non è incluso in una transazione, Aurora SQL My utilizza commit impliciti. Autocommit è attivo. Ciò significa che per ogni inserimento di riga è presente l'esecuzione di un commit. Performance Insights mostra che le connessioni dedicano la maggior parte del tempo in attesa dell'evento di attesa `io/aurora_redo_log_flush`.

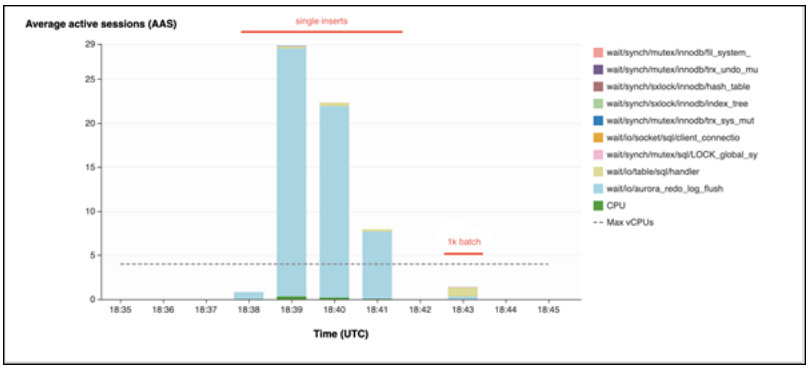


Ciò è causato dalle semplici istruzioni di inserimento utilizzate.



I 50.000 dati richiedono 3,5 minuti per essere inseriti.

- Nel secondo esempio, gli inserti sono realizzati in 1.000 batch, ovvero ogni connessione esegue 10 commit anziché 10.000. Performance Insights mostra che le connessioni non dedicano la maggior parte del loro tempo sull'evento di attesa `io/aurora_redo_log_flush`.



I 50.000 dati richiedono 4 secondi per essere inseriti.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query problematiche
- Raggruppa le operazioni di scrittura
- Disattivazione dell'autocommit
- Transazioni di utilizzo
- Utilizza i batch

Identificare le sessioni e le query problematiche

Se l'istanza database sta riscontrando un collo di bottiglia, il primo compito è quello di trovare le sessioni e le query che lo causano. Per un utile post sul blog AWS Database, consulta Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Per identificare sessioni e query che causano un collo di bottiglia

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli l'istanza database.
4. In Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Le query nella parte superiore dell'elenco causano il carico più alto sul database.

Raggruppa le operazioni di scrittura

Gli esempi seguenti attivano l'evento di attesa `io/aurora_redo_log_flush`. (Autocommit è attivo.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
```

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Per ridurre il tempo impiegato in attesa dell'evento di attesa `io/aurora_redo_log_flush`, raggruppa logicamente le operazioni di scrittura in una singola esecuzione di commit per ridurre le chiamate persistenti all'archiviazione.

Disattivazione dell'autocommit

Disattivare l'autocommit prima di apportare modifiche di grandi dimensioni che non si trovano all'interno di una transazione, come illustrato nell'esempio seguente.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Transazioni di utilizzo

È possibile utilizzare le transazioni, come illustrato nell'esempio seguente.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
```

```
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizza i batch

Si può anche apportare modifiche in batch, come nell'esempio seguente: Tuttavia, l'utilizzo di batch troppo grandi può causare problemi di prestazioni, specialmente nelle repliche di lettura o durante il point-in-time ripristino ()PITR.

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/aurora_respond_to_client

L'evento `io/aurora_respond_to_client` si verifica quando un thread è in attesa di restituire una serie di risultati a un client.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 2 SQL

Context

L'evento `io/aurora_respond_to_client` indica che un thread è in attesa di restituire una serie di risultati a un client.

L'elaborazione della query è completa e i risultati vengono restituiti al client dell'applicazione. Tuttavia, poiché non c'è abbastanza larghezza di banda di rete nel cluster del database, un thread è in attesa di restituire la serie di risultati.

Probabili cause di aumento delle attese

Quando l'evento `io/aurora_respond_to_client` appare più che normale, possibilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

Classe di istanza database insufficiente per il carico di lavoro

: La classe di istanza database utilizzata dal cluster del database non dispone della larghezza di banda di rete necessaria per elaborare il carico di lavoro in modo efficiente.

Serie di risultati di grandi dimensioni

: Si è verificato un aumento delle dimensioni della serie di risultati restituiti, poiché la query restituisce un numero maggiore di righe. La serie di risultati più ampia consuma una maggiore larghezza di banda di rete.

Aumento del carico sul client

: Potrebbe esserci CPU pressione, pressione della memoria o saturazione della rete sul client. Un aumento del carico sul client ritarda la ricezione dei dati dal cluster Aurora SQL My DB.

Maggiore latenza di rete

: Potrebbe esserci un aumento della latenza di rete tra il cluster Aurora SQL My DB e il client. Una maggiore latenza di rete aumenta il tempo necessario per la ricezione dei dati dal client.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Dimensionamento della classe dell'istanza database
- Verifica del carico di lavoro per risultati imprevisti
- Distribuisci il carico di lavoro con le istanze del lettore
- Usa il SQL modificatore _ BUFFER _ RESULT

Identificare le sessioni e le query che causano gli eventi

È possibile utilizzare Performance Insights per mostrare le query bloccate dall'evento di attesa `io/aurora_respond_to_client`. In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare le SQL interrogazioni responsabili di un carico elevato

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog AWS Database Analyze Amazon Aurora My SQL Workloads with Performance Insights.

Dimensionamento della classe dell'istanza database

Verifica l'aumento del valore delle CloudWatch metriche di Amazon relative al throughput di rete, ad esempio `NetworkReceiveThroughput` e. `NetworkTransmitThroughput` Se viene raggiunta la larghezza di banda di rete della classe di istanza database, è possibile dimensionare la classe di

istanza database utilizzata dal cluster del database modificando il cluster del database. Una classe di istanze database con larghezza di banda di rete maggiore restituisce i dati ai client in modo più efficiente.

Per informazioni sul monitoraggio dei CloudWatch parametri di Amazon, consultaVisualizzazione delle metriche nella console Amazon RDS. Per informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster di database, consulta Modifica di un cluster database Amazon Aurora.

Verifica del carico di lavoro per risultati imprevisti

Controlla il carico di lavoro sul cluster del database e assicurati che non produca risultati imprevisti. Ad esempio, potrebbero esserci query che restituiscono un numero di righe più alto del previsto. In questo caso, puoi utilizzare i parametri del contatore di Performance Insights come `Innodb_rows_read`. Per ulteriori informazioni, consulta Parametri contatore di Performance Insights.

Distribuisci il carico di lavoro con le istanze del lettore

È possibile distribuire il carico di lavoro di sola lettura con le repliche di Aurora. È possibile dimensionare orizzontalmente aggiungendo più repliche di Aurora. Ciò può comportare un aumento dei limiti per la larghezza di banda della rete. Per ulteriori informazioni, consulta Cluster database Amazon Aurora.

Usa il SQL modificatore _ BUFFER _ RESULT

Puoi aggiungere il modificatore `SQL_BUFFER_RESULT` alle istruzioni `SELECT` per forzare il risultato in una tabella temporanea prima che vengano restituite al client. Questo modificatore può aiutare a risolvere i problemi di prestazioni quando i blocchi InnoDB non vengono liberati perché le query sono presenti nello stato di attesa `io/aurora_respond_to_client`. Per ulteriori informazioni, vedi SELECTDichiarazione nella mia SQL documentazione.

io/redo_log_flush

L'evento `io/redo_log_flush` si verifica quando una sessione sta scrivendo dati persistenti nell'archiviazione di Amazon Aurora.

Argomenti

- Versioni del motore supportate

- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 3 SQL

Context

L'`io/redo_log_flush`evento riguarda un'operazione di scrittura input/output (I/O () in Aurora My. SQL

> **Note**
>
> In Aurora My SQL versione 2, questo evento di attesa è denominato io/aurora_redo_log_flush.

Probabili cause di aumento delle attese

Per la persistenza dei dati, l'esecuzione dei commit richiede una scrittura duratura per un'archiviazione stabile. Se il database sta eseguendo troppi commit, c'è un evento di attesa sull'operazione di I/O di scrittura, l'evento di attesa `io/redo_log_flush`.

Per esempi del comportamento di questo evento di attesa, vedi. io/aurora_redo_log_flush

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query problematiche
- Raggruppa le operazioni di scrittura
- Disattivazione dell'autocommit

- Transazioni di utilizzo
- Utilizza i batch

Identificare le sessioni e le query problematiche

Se l'istanza database sta riscontrando un collo di bottiglia, il primo compito è quello di trovare le sessioni e le query che lo causano. Per un utile post sul blog AWS Database, consulta Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Per identificare sessioni e query che causano un collo di bottiglia

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli l'istanza database.
4. In Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Le query nella parte superiore dell'elenco causano il carico più alto sul database.

Raggruppa le operazioni di scrittura

Gli esempi seguenti attivano l'evento di attesa `io/redo_log_flush`. (Autocommit è attivo.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

```
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Per ridurre il tempo impiegato in attesa dell'evento di attesa `io/redo_log_flush`, raggruppa logicamente le operazioni di scrittura in una singola esecuzione di commit per ridurre le chiamate persistenti all'archiviazione.

Disattivazione dell'autocommit

Disattivare l'autocommit prima di apportare modifiche di grandi dimensioni che non si trovano all'interno di una transazione, come illustrato nell'esempio seguente.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Transazioni di utilizzo

È possibile utilizzare le transazioni, come illustrato nell'esempio seguente.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
END
```

Utilizza i batch

Si può anche apportare modifiche in batch, come nell'esempio seguente: Tuttavia, l'utilizzo di batch troppo grandi può causare problemi di prestazioni, specialmente nelle repliche di lettura o durante il point-in-time ripristino ()PITR.

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/socket/sql/client_connessione

L'evento `io/socket/sql/client_connection` si verifica quando un thread sta per gestire una nuova connessione.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora SQL Le mie versioni 2 e 3

Context

L'evento `io/socket/sql/client_connection` indica che mysqld è impegnato a creare thread per gestire le nuove connessioni client in arrivo. In questo scenario, l'elaborazione della manutenzione delle nuove richieste di connessione client rallenta mentre le connessioni attendono l'assegnazione del thread. Per ulteriori informazioni, consulta Server MySQL (mysqld).

Probabili cause di aumento delle attese

Quando l'evento si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

- C'è un aumento improvviso delle nuove connessioni utente dall'applicazione alla tua RDS istanza Amazon.
- La tua istanza DB non può elaborare nuove connessioni perché la CPU rete o la memoria sono limitate.

Azioni

Se `io/socket/sql/client_connection` domina l'attività del database, non indica necessariamente un problema di prestazioni. In un database che non è inattivo, un evento di attesa è sempre in primo piano. Agisci solo quando le prestazioni diminuiscono. Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query problematiche
- Seguire le best practice per la gestione delle connessioni
- Dimensiona verso l'alto la tua istanza se le risorse vengono limitate
- Controlla i principali host e i migliori utenti
- Interrogare le tabelle performance_schema
- Controlla lo stato del thread delle query
- Verifica le tue richieste e le query
- Crea pool delle connessioni al database

Identificare le sessioni e le query problematiche

Se l'istanza database sta riscontrando un collo di bottiglia, il primo compito è quello di trovare le sessioni e le query che lo causano. Per un utile post sul blog, consulta Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Per identificare sessioni e query che causano un collo di bottiglia

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli l'istanza database.
4. In Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Le query nella parte superiore dell'elenco causano il carico più alto sul database.

Seguire le best practice per la gestione delle connessioni

Per gestire le tue connessioni, considera le seguenti strategie:

- Utilizza il pooling di connessioni.

  È possibile aumentare gradualmente il numero di connessioni secondo necessità. Per ulteriori informazioni, consulta il white paper Amazon Aurora My SQL Database Administrator's Handbook.
- Usa un nodo lettore per ridistribuire il traffico di sola lettura.

  Per ulteriori informazioni, consulta Repliche di Aurora e Connessioni endpoint Amazon Aurora.

Dimensiona verso l'alto la tua istanza se le risorse vengono limitate

Cerca esempi di limitazione nelle seguenti risorse:

- CPU

  Controlla i CloudWatch parametri di Amazon per un CPU utilizzo elevato.
- Rete

  Verifica la presenza di un aumento del valore delle CloudWatch metriche `network receive throughput` e. `network transmit throughput` Se la tua istanza ha raggiunto il limite di larghezza di banda di rete per la tua classe di istanza, valuta la possibilità di scalare l'RDSistanza a un tipo di classe di istanza superiore. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.
- Memoria liberabile

  Verifica se la metrica non è diminuita CloudWatch . `FreeableMemory` Inoltre, prendi in considerazione l'attivazione del monitoraggio avanzato. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Controlla i principali host e i migliori utenti

Usa Performance Insights per controllare i principali host e i migliori utenti. Per ulteriori informazioni, consulta Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights.

Interrogare le tabelle performance_schema

Per ottenere un conteggio accurato delle connessioni correnti e totali, eseguire una query sulle tabelle `performance_schema`. Con questa tecnica, si identifica l'utente o l'host di origine responsabile della creazione di un numero elevato di connessioni. Ad esempio, interrogare le tabelle `performance_schema` come indicato di seguito.

```
SELECT * FROM performance_schema.accounts;
SELECT * FROM performance_schema.users;
SELECT * FROM performance_schema.hosts;
```

Controlla lo stato del thread delle query

Se il problema relativo alle prestazioni è continuo, controlla lo stato del thread delle query. Nel client `mysql`, eseguire il comando seguente.

```
show processlist;
```

Verifica le tue richieste e le query

Per verificare la natura delle richieste e delle interrogazioni provenienti dagli account utente, usa AuroraAurora My SQL Advanced Auditing. Per informazioni su come attivare l'auditing, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.

Crea pool delle connessioni al database

Prendi in considerazione l'utilizzo di Amazon RDS Proxy per la gestione delle connessioni. Utilizzando RDS Proxy, puoi consentire alle tue applicazioni di raggruppare e condividere connessioni al database per migliorarne la scalabilità. RDS Il proxy rende le applicazioni più resistenti agli errori del database connettendosi automaticamente a un'istanza DB in standby e preservando al contempo le connessioni delle applicazioni. Per ulteriori informazioni, consulta Proxy Amazon RDS per Aurora.

io/table/sql/handler

L'evento `io/table/sql/handler` si verifica quando il lavoro è stato delegato a un motore di archiviazione.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora SQL Le mie versioni 2 e 3

Context

L'evento `io/table` indica un'attesa per l'accesso a una tabella. Questo evento si verifica indipendentemente dal fatto che i dati siano memorizzati nella cache nel pool di buffer o se si acceda su disco. L'evento `io/table/sql/handler` indica un aumento dell'attività del carico di lavoro.

Un handler è una routine specializzata in un determinato tipo di dati o incentrata su determinate attività speciali. Ad esempio, un handler di eventi riceve e digerisce eventi e segnali dal sistema operativo o da un'interfaccia utente. Un handler di memoria esegue processi relativi alla memoria. Un handler di input di file è una funzione che riceve l'input di file ed esegue attività speciali sui dati, in base al contesto.

Visualizzazioni come `performance_schema.events_waits_current` spesso mostrano `io/table/sql/handler` quando l'attesa effettiva è un evento di attesa annidato come un blocco. Quando l'attesa effettiva non è `io/table/sql/handler`, Approfondimenti sulle prestazioni segnala l'evento di attesa annidato. Quando Performance Insights riporta`io/table/sql/handler`, rappresenta l'elaborazione di InnoDB della richiesta di I/O e non un evento di attesa annidato nascosto. Per ulteriori informazioni, consulta Performance Schema Atom and Molecule Events nel My Reference Manual. SQL

L'`io/table/sql/handler`evento compare spesso negli eventi di massima attesa con attese di I/O come. `io/aurora_redo_log_flush`

Probabili cause di aumento delle attese

In Approfondimenti sulle prestazioni, picchi improvvisi nell'evento `io/table/sql/handler` indicano un aumento dell'attività del carico di lavoro. Aumento dell'attività significa un aumento dell'I/O.

Performance Insights filtra l'evento di nidificazione IDs e non segnala un'`io/table/sql/handler`attesa quando l'evento nidificato sottostante è un'attesa di blocco. Ad esempio, se l'evento causa principale è synch/mutex/innodb/aurora_lock_thread_slot_futex, Approfondimenti sulle prestazioni visualizza questa attesa nei primi eventi di attesa e non `io/table/sql/handler`.

In visualizzazioni come `performance_schema.events_waits_current`, le attese di `io/table/sql/handler` spesso appaiono quando l'attesa effettiva è un evento di attesa annidato come un blocco. Quando l'attesa effettiva è diversa da `io/table/sql/handler`, Approfondimenti sulle prestazioni cerca l'attesa annidata e segnala l'attesa effettiva anziché `io/table/sql/handler`. Quando Approfondimenti sulle prestazioni segnala `io/table/sql/handler`, la vera attesa è `io/table/sql/handler` e non un evento di attesa annidato nascosto. Per ulteriori informazioni, consulta Performance Schema Atom and Molecule Events nel manuale di riferimento My SQL 5.7.

Azioni

Se l'evento di attesa domina l'attività del database, non indica necessariamente un problema di prestazioni. Un evento di attesa è sempre in primo piano quando il database è attivo. È necessario agire solo quando le prestazioni diminuiscono.

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Verifica la presenza di una correlazione con i parametri dei contatori di Approfondimenti sulle prestazioni
- Verifica la presenza di altri eventi di attesa correlati

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare le SQL interrogazioni responsabili di un carico elevato

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Verifica la presenza di una correlazione con i parametri dei contatori di Approfondimenti sulle prestazioni

Verifica la presenza di parametri del contatore di Approfondimenti sulle prestazioni come `Innodb_rows_changed`. Se i parametri del contatore sono correlati con `io/table/sql/handler`, segui questi passaggi:

1. In Performance Insights, cerca i SQL rendiconti che rappresentano l'evento di `io/table/sql/handler` massima attesa. Se possibile, ottimizza questa istruzione in modo che restituisca un numero inferiore di righe.
2. Recupera le tabelle principali dalle visualizzazione `schema_table_statistics` e `x$schema_table_statistics`. Queste visualizzazioni mostrano la quantità di tempo impiegato per tabella. Per ulteriori informazioni, consulta Le viste schema_table_statistics e x $schema_table_statistics nel My Reference Manual. SQL

   Per impostazione predefinita, le righe vengono ordinate in base al tempo di attesa totale discendente. Le tabelle con il maggior numero di contese appaiono per prime. L'output indica se il tempo viene impiegato per le letture, le scritture, il recupero, gli inserimenti, gli aggiornamenti o le eliminazioni.

```
mysql> select * from sys.schema_table_statistics limit 1\G

*************************** 1. row ***************************
     table_schema: read_only_db
       table_name: sbtest41
    total_latency: 54.11 m
```

```
     rows_fetched: 6001557
    fetch_latency: 39.14 m
    rows_inserted: 14833
   insert_latency: 5.78 m
     rows_updated: 30470
   update_latency: 5.39 m
     rows_deleted: 14833
   delete_latency: 3.81 m
 io_read_requests: NULL
          io_read: NULL
  io_read_latency: NULL
io_write_requests: NULL
         io_write: NULL
 io_write_latency: NULL
 io_misc_requests: NULL
  io_misc_latency: NULL
1 row in set (0.11 sec)
```

Verifica la presenza di altri eventi di attesa correlati

Se `synch/sxlock/innodb/btr_search_latch` e `io/table/sql/handler` insieme contribuiscono maggiormente all'anomalia del carico del database, verificare se la variabile `innodb_adaptive_hash_index` è attivata. Se lo è, considera la possibilità di aumentare il valore del parametro `innodb_adaptive_hash_index_parts`.

Se l'indice adattivo Hash è disattivato, considera la possibilità di attivarlo. Per ulteriori informazioni sul My Adaptive Hash Index, consulta le seguenti risorse: SQL

- L'articolo L'indice adattivo Hash in InnoDB è adatto al mio carico di lavoro? sul sito web di Percona
- Adaptive Hash Index nel My Reference Manual SQL
- L'articolo Contention in My SQL InnoDB: informazioni utili dalla sezione Semafori sul sito Web di Percona

> **Note**
>
> L'indice adattivo Hash non è supportato nelle istanze database di lettura di Aurora.
> In alcuni casi, le prestazioni potrebbero risultare scadenti su un'istanza di lettura quando `synch/sxlock/innodb/btr_search_latch` e `io/table/sql/handler` sono

dominanti. In tal caso, prendere in considerazione il reindirizzamento temporaneo del carico di lavoro all'istanza database di scrittura e l'attivazione dell'indice adattivo Hash.

synch/cond/innodb/row_blocca_aspetta

L'evento `synch/cond/innodb/row_lock_wait` si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga. Per ulteriori informazioni, consulta InnoDB locking nella mia documentazione. SQL

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 3 SQL

Probabili cause di aumento delle attese

Più istruzioni Data Manipulation Language (DML) accedono alla stessa riga o alle stesse righe contemporaneamente.

Azioni

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Trova e rispondi alle SQL dichiarazioni responsabili di questo evento di attesa
- Trova e rispondi alla sessione di blocco

Trova e rispondi alle SQL dichiarazioni responsabili di questo evento di attesa

Usa Performance Insights per identificare le SQL dichiarazioni responsabili di questo evento di attesa. Considera le strategie seguenti:

- Se i blocchi di riga sono un problema persistente, considera la possibilità di riscrivere l'applicazione per utilizzare il blocco ottimistico.
- Utilizza istruzioni multiriga.
- Distribuisci il carico di lavoro su diversi oggetti di database. Per farlo, puoi utilizzare il partizionamento.

- Verifica il valore del parametro `innodb_lock_wait_timeout`. Controlla la durata di attesa delle transazioni prima di generare un errore di timeout.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Trova e rispondi alla sessione di blocco

Determina se la sessione di blocco è inattiva o attiva. Inoltre, scopri se la sessione proviene da un'applicazione o da un utente attivo.

Per identificare la sessione che tiene il blocco, è possibile eseguire `SHOW ENGINE INNODB STATUS`. Il seguente esempio mostra un output campione.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 1688153, ACTIVE 82 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 4244, OS thread handle 70369524330224, query id 4020834 172.31.14.179
 reinvent executing
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 24 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 4 n bits 72 index GEN_CLUST_INDEX of table test.t1 trx
 id 1688153 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

In alternativa, è possibile usare la seguente query per estrarre i dettagli sui blocchi correnti.

```
mysql> SELECT p1.id waiting_thread,
    p1.user waiting_user,
    p1.host waiting_host,
    it1.trx_query waiting_query,
    ilw.requesting_engine_transaction_id waiting_transaction,
    ilw.blocking_engine_lock_id blocking_lock,
    il.lock_mode blocking_mode,
    il.lock_type blocking_type,
    ilw.blocking_engine_transaction_id blocking_transaction,
    CASE it.trx_state
        WHEN 'LOCK WAIT'
        THEN it.trx_state
```

```
        ELSE p.state end blocker_state,
    concat(il.object_schema,'.', il.object_name) as locked_table,
    it.trx_mysql_thread_id blocker_thread,
    p.user blocker_user,
    p.host blocker_host
FROM performance_schema.data_lock_waits ilw
JOIN performance_schema.data_locks il
ON ilw.blocking_engine_lock_id = il.engine_lock_id
AND ilw.blocking_engine_transaction_id = il.engine_transaction_id
JOIN information_schema.innodb_trx it
ON ilw.blocking_engine_transaction_id = it.trx_id join information_schema.processlist p
ON it.trx_mysql_thread_id = p.id join information_schema.innodb_trx it1
ON ilw.requesting_engine_transaction_id = it1.trx_id join
 information_schema.processlist p1
ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
waiting_thread: 4244
waiting_user: reinvent
waiting_host: 123.456.789.012:18158
waiting_query: select id1 from test.t1 where id1=1 for update
waiting_transaction: 1688153
blocking_lock: 70369562074216:11:4:2:70369549808672
blocking_mode: X
blocking_type: RECORD
blocking_transaction: 1688142
blocker_state: User sleep
locked_table: test.t1
blocker_thread: 4243
blocker_user: reinvent
blocker_host: 123.456.789.012:18156
1 row in set (0.00 sec)
```

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.
- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni sull'identificazione delle transazioni bloccanti, vedere Utilizzo della transazione InnoDB e blocco delle informazioni nella mia documentazione. SQL

synch/cond/innodb/row_lock_wait_cond

L'evento `synch/cond/innodb/row_lock_wait_cond` si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga. Per ulteriori informazioni, consulta InnoDB locking nella mia documentazione. SQL

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 2 SQL

Probabili cause di aumento delle attese

Più istruzioni Data Manipulation Language (DML) accedono alla stessa riga o alle stesse righe contemporaneamente.

Azioni

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Trova e rispondi alle SQL dichiarazioni responsabili di questo evento di attesa
- Trova e rispondi alla sessione di blocco

Trova e rispondi alle SQL dichiarazioni responsabili di questo evento di attesa

Usa Performance Insights per identificare le SQL dichiarazioni responsabili di questo evento di attesa. Considera le strategie seguenti:

- Se i blocchi di riga sono un problema persistente, considera la possibilità di riscrivere l'applicazione per utilizzare il blocco ottimistico.
- Utilizza istruzioni multiriga.
- Distribuisci il carico di lavoro su diversi oggetti di database. Per farlo, puoi utilizzare il partizionamento.
- Verifica il valore del parametro `innodb_lock_wait_timeout`. Controlla la durata di attesa delle transazioni prima di generare un errore di timeout.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Trova e rispondi alla sessione di blocco

Determina se la sessione di blocco è inattiva o attiva. Inoltre, scopri se la sessione proviene da un'applicazione o da un utente attivo.

Per identificare la sessione che tiene il blocco, è possibile eseguire `SHOW ENGINE INNODB STATUS`. Il seguente esempio mostra un output campione.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 2771110, ACTIVE 112 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 24, OS thread handle 70369573642160, query id 13271336 172.31.14.179
 reinvent Sending data
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 43 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 3 n bits 0 index GEN_CLUST_INDEX of table test.t1 trx
 id 2771110 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

In alternativa, è possibile usare la seguente query per estrarre i dettagli sui blocchi correnti.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
```

```
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.
- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni sull'identificazione delle transazioni bloccanti, vedere Utilizzo della transazione InnoDB e blocco delle informazioni nella mia documentazione. SQL

synch/cond/sql/MDL_contesto:: COND _wait_status

L'evento `synch/cond/sql/MDL_context::COND_wait_status` si verifica quando ci sono thread in attesa del blocco dei metadati di una tabella.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora SQL Le mie versioni 2 e 3

Context

L'evento `synch/cond/sql/MDL_context::COND_wait_status` indica che ci sono thread in attesa di un blocco dei metadati di tabella. In alcuni casi, una sessione contiene un blocco di metadati su una tabella e un'altra sessione tenta di ottenere lo stesso blocco sulla stessa tabella. In tal caso, la seconda sessione attende l'evento di attesa `synch/cond/sql/MDL_context::COND_wait_status`.

My SQL utilizza il blocco dei metadati per gestire l'accesso simultaneo agli oggetti del database e garantire la coerenza dei dati. Il blocco dei metadati si applica a tabelle, schemi, eventi pianificati, tablespace e blocchi utente acquisiti con la funzione `get_lock` e i programmi memorizzati. I programmi memorizzati includono procedure, funzioni e attivazioni. Per ulteriori informazioni, consulta Blocco dei metadati nella documentazione personale. SQL

L'elenco I miei SQL processi mostra questa sessione nello stato. `waiting for metadata lock` In Approfondimenti sulle prestazioni, se `Performance_schema` è acceso, l'evento `synch/cond/sql/MDL_context::COND_wait_status` viene visualizzato.

Il timeout predefinito per una query in attesa di un blocco dei metadati si basa sul valore del parametro `lock_wait_timeout`, che viene impostato in modo predefinito a 31.536.000 secondi (365 giorni).

Per maggiori dettagli sui diversi blocchi InnoDB e sui tipi di blocchi che possono causare conflitti, consulta InnoDB Locking nella mia documentazione. SQL

Probabili cause di aumento delle attese

Quando l'evento `synch/cond/sql/MDL_context::COND_wait_status` appare più che normale, possibilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

Transazioni di lunga durata

Una o più transazioni stanno modificando una grande quantità di dati e mantengono i blocchi sulle tabelle per un tempo molto lungo.

Transazioni inattive

Una o più transazioni rimangono aperte per un lungo periodo, senza che venga eseguito il commit o riprimisto a una precedente versione.

DDListruzioni su tabelle di grandi dimensioni

Una o più istruzioni di definizione dei dati (DDL), come `ALTER TABLE` i comandi, sono state eseguite su tabelle molto grandi.

Blocchi espliciti della tabella

Ci sono blocchi espliciti sulle tabelle che non vengono rilasciate in modo tempestivo. Ad esempio, potrebbe essere eseguita un'applicazione su istruzioni `LOCK TABLE` impropriamente.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa e della versione del cluster Aurora My SQL DB.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Verifica la presenza di eventi passati
- Esegui interrogazioni su Aurora My versione 2 SQL
- Rispondere alla sessione di blocco

Identificare le sessioni e le query che causano gli eventi

È possibile utilizzare Approfondimenti sulle prestazioni per mostrare le query bloccate dall'evento di attesa `synch/cond/sql/MDL_context::COND_wait_status`. Tuttavia, per identificare la sessione di blocco, eseguire una query sulle tabelle dei metadati da `performance_schema` e `information_schema` sul cluster del database.

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare SQL le query responsabili di un carico elevato

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog AWS Database Analyze Amazon Aurora My SQL Workloads with Performance Insights.

Verifica la presenza di eventi passati

È possibile ottenere informazioni dettagliate su questo evento di attesa per verificare la presenza di occorrenze passate. Per fare ciò, completare questa procedura:

- Controlla il linguaggio di manipolazione dei dati (DML), la DDL velocità effettiva e la latenza per vedere se ci sono state modifiche nel carico di lavoro.

È possibile utilizzare Approfondimenti sulle prestazioni per trovare query in attesa di questo evento al momento del problema. Inoltre, è possibile visualizzare il digest delle query eseguite vicino al momento del problema.

- Se i registri di verifica o i registri generali sono attivati per il cluster el database, è possibile verificare la presenza di tutte le query eseguite sugli oggetti (schema.table) coinvolti nella transazione in attesa. È inoltre possibile verificare la presenza delle query completate in esecuzione prima della transazione.

Le informazioni disponibili per l'identificazione e la risoluzione dei problemi degli eventi passati sono limitate. L'esecuzione di queste verifiche non mostra quale oggetto è in attesa di informazioni. Tuttavia, è possibile identificare le tabelle con un carico pesante al momento dell'evento e la serie di righe gestite di frequente che causano conflitti al momento del problema. È quindi possibile utilizzare queste informazioni per riprodurre il problema in un ambiente di test e fornire informazioni dettagliate sulla sua causa.

Esegui interrogazioni su Aurora My versione 2 SQL

In Aurora My SQL versione 2, è possibile identificare direttamente la sessione bloccata interrogando `performance_schema` tabelle o `sys` viste dello schema. Un esempio può illustrare come interrogare le tabelle per identificare query e sessioni di blocco.

Nel seguente output dell'elenco dei processi, l'ID di connessione 89 è in attesa di un blocco dei metadati e sta eseguendo un comando `TRUNCATE TABLE`. In una query sulle tabelle `performance_schema` o visualizzazioni dello schema `sys`, l'output mostra che la sessione di blocco è 76.

```
MySQL [(none)]> select @@version, @@aurora_version;
+-----------+------------------+
| @@version | @@aurora_version |
+-----------+------------------+
| 5.7.12    | 2.11.5           |
+-----------+------------------+
1 row in set (0.01 sec)

MySQL [(none)]> show processlist;
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+-------------------------------+
| Id | User            | Host               | db        | Command | Time | State
                    | Info                          |
```

```
+----+-----------------+--------------------+-----------+---------+------
+--------------------------------+------------------------------+
|  2 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                           |
|  4 | rdsadmin        | localhost          | NULL      | Sleep   |    2 | NULL
                    | NULL                           |
|  5 | rdsadmin        | localhost          | NULL      | Sleep   |    1 | NULL
                    | NULL                           |
| 20 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                           |
| 21 | rdsadmin        | localhost          | NULL      | Sleep   |  261 | NULL
                    | NULL                           |
| 66 | auroramysql5712 | 172.31.21.51:52154 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 67 | auroramysql5712 | 172.31.21.51:52158 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 68 | auroramysql5712 | 172.31.21.51:52150 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 69 | auroramysql5712 | 172.31.21.51:52162 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 70 | auroramysql5712 | 172.31.21.51:52160 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 71 | auroramysql5712 | 172.31.21.51:52152 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 72 | auroramysql5712 | 172.31.21.51:52156 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 73 | auroramysql5712 | 172.31.21.51:52164 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 74 | auroramysql5712 | 172.31.21.51:52166 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 75 | auroramysql5712 | 172.31.21.51:52168 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
| 76 | auroramysql5712 | 172.31.21.51:52170 | NULL      | Query   |    0 | starting
                    | show processlist               |
| 88 | auroramysql5712 | 172.31.21.51:52194 | NULL      | Query   |   22 | User sleep
                    | select sleep(10000)            |
| 89 | auroramysql5712 | 172.31.21.51:52196 | NULL      | Query   |    5 | Waiting for
 table metadata lock | truncate table sbtest.sbtest1 |
+----+-----------------+--------------------+-----------+---------+------
+--------------------------------+------------------------------+
18 rows in set (0.00 sec)
```

Successivamente, una query sulle tabelle `performance_schema` o visualizzazioni dello schema `sys` mostra che la sessione di blocco è 76.

```
MySQL [(none)]> select * from sys.schema_table_lock_waits;

+---------------+-------------+-------------------+-------------
+-------------------------------+-------------------+-----------------------
+-------------------------------+--------------------+-----------------------------
+-------------------------------+--------------------+-------------
+-------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
| object_schema | object_name | waiting_thread_id | waiting_pid | waiting_account
           | waiting_lock_type | waiting_lock_duration | waiting_query
   | waiting_query_secs | waiting_query_rows_affected | waiting_query_rows_examined |
 blocking_thread_id | blocking_pid | blocking_account             | blocking_lock_type
 | blocking_lock_duration | sql_kill_blocking_query | sql_kill_blocking_connection |
+---------------+-------------+-------------------+-------------
+-------------------------------+-------------------+-----------------------
+-------------------------------+--------------------+-----------------------------
+-------------------------------+--------------------+-------------
+-------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
| sbtest        | sbtest1     |               121 |          89 |
 auroramysql5712@192.0.2.0    | EXCLUSIVE         | TRANSACTION           | truncate
 table sbtest.sbtest1 |                 10 |                            0 |
                0 |                108 |           76 | auroramysql5712@192.0.2.0    |
 SHARED_READ        | TRANSACTION            | KILL QUERY 76           | KILL 76
               |
+---------------+-------------+-------------------+-------------
+-------------------------------+-------------------+-----------------------
+-------------------------------+--------------------+-----------------------------
+-------------------------------+--------------------+-------------
+-------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
1 row in set (0.00 sec)
```

Rispondere alla sessione di blocco

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.

- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni sull'identificazione delle transazioni bloccanti, consulta Using InnoDB Transaction and Locking Information nella mia documentazione. SQL

synch/mutex/innodb/aurora_lock_thread_slot_futex

L'evento `synch/mutex/innodb/aurora_lock_thread_slot_futex` si verifica quando una sessione ha bloccato una riga per un aggiornamento e un'altra sessione tenta di aggiornare la stessa riga. Per ulteriori informazioni, consulta InnoDB locking in My Reference. SQL

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 2 SQL

Probabili cause di aumento delle attese

Più istruzioni Data Manipulation Language (DML) accedono alla stessa riga o alle stesse righe contemporaneamente.

Azioni

Consigliamo azioni diverse a seconda degli altri eventi di attesa visualizzati.

Argomenti

- Trova e rispondi alle SQL dichiarazioni responsabili di questo evento di attesa
- Trova e rispondi alla sessione di blocco

Trova e rispondi alle SQL dichiarazioni responsabili di questo evento di attesa

Usa Performance Insights per identificare le SQL dichiarazioni responsabili di questo evento di attesa. Considera le strategie seguenti:

- Se i blocchi di riga sono un problema persistente, considera la possibilità di riscrivere l'applicazione per utilizzare il blocco ottimistico.

- Utilizza istruzioni multiriga.
- Distribuisci il carico di lavoro su diversi oggetti di database. Per farlo, puoi utilizzare il partizionamento.
- Verifica il valore del parametro `innodb_lock_wait_timeout`. Controlla la durata di attesa delle transazioni prima di generare un errore di timeout.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Trova e rispondi alla sessione di blocco

Determina se la sessione di blocco è inattiva o attiva. Inoltre, scopri se la sessione proviene da un'applicazione o da un utente attivo.

Per identificare la sessione che tiene il blocco, è possibile eseguire `SHOW ENGINE INNODB STATUS`. Il seguente esempio mostra un output campione.

```
mysql> SHOW ENGINE INNODB STATUS;

----------------------TRANSACTION 302631452, ACTIVE 2 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 376, 1 row lock(s)
MySQL thread id 80109, OS thread handle 0x2ae915060700, query id 938819 10.0.4.12
 reinvent updating
UPDATE sbtest1 SET k=k+1 WHERE id=503
------- TRX HAS BEEN WAITING 2 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 148 page no 11 n bits 30 index `PRIMARY` of table
 `sysbench2`.`sbtest1` trx id 302631452 lock_mode X locks rec but not gap waiting
Record lock, heap no 30 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
```

In alternativa, è possibile usare la seguente query per estrarre i dettagli sui blocchi correnti.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
```

```
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
         FROM information_schema.innodb_lock_waits ilw
         JOIN information_schema.innodb_locks il
           ON ilw.blocking_lock_id = il.lock_id
          AND ilw.blocking_trx_id = il.lock_trx_id
         JOIN information_schema.innodb_trx it
           ON ilw.blocking_trx_id = it.trx_id
         JOIN information_schema.processlist p
           ON it.trx_mysql_thread_id = p.id
         JOIN information_schema.innodb_trx it1
           ON ilw.requesting_trx_id = it1.trx_id
         JOIN information_schema.processlist p1
           ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Quando si identifica la sessione, le opzioni includono quanto segue:

- Contattare il proprietario dell'applicazione o l'utente.

- Se la sessione di blocco è inattiva, considerare di terminare la sessione di blocco. Questa azione potrebbe innescare un lungo ripristino dello stato precedente. Per informazioni su come terminare una sessione, consulta Terminare una sessione o una query.

Per ulteriori informazioni sull'identificazione delle transazioni bloccanti, vedere Utilizzo delle informazioni di transazione e blocco di InnoDB nel My SQL Reference Manual.

synch/mutex/innodb/buf_pool_mutex

L'evento `synch/mutex/innodb/buf_pool_mutex` si verifica quando un thread ha acquisito un blocco nel pool di buffer InnoDB per accedere a una pagina in memoria.

Argomenti

- Versioni del motore rilevanti
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore rilevanti

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora La mia versione 2 SQL

Context

La mutex `buf_pool` è una mutex singola che protegge le strutture di dati di controllo del pool di buffer.

Per ulteriori informazioni, consulta Monitoring InnoDB Mutex Waits Using Performance Schema nella mia documentazione. SQL

Probabili cause di aumento delle attese

Si tratta di un evento di attesa specifico per il carico di lavoro. Cause comuni perché `synch/mutex/innodb/buf_pool_mutex` appaia tra i primi eventi di attesa includono quanto segue:

- Le dimensioni del pool di buffer non sono abbastanza grandi da contenere la serie di dati funzionante.

- Il carico di lavoro è più specifico per determinate pagine di una tabella specifica del database, causando contese nel pool di buffer.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Utilizzo di Performance Insights
- Crea repliche di Aurora
- Analisi delle dimensioni del pool di buffer
- Monitoraggio della cronologia di stato globale

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per visualizzare il SQL grafico principale nella console di gestione AWS

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Sotto il grafico di caricamento del database, scegli Top. SQL

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Utilizzo di Performance Insights

Questo evento è correlato al carico di lavoro. È possibile utilizzare Performance Insights per effettuare quanto segue:

- Identificare l'avvio degli eventi di attesa e se c'è qualche modifica nel carico di lavoro in quel periodo dai registri dell'applicazione o dalle origini correlate.
- Identifica le SQL dichiarazioni responsabili di questo evento di attesa. Esaminare il piano di esecuzione delle query per accertarsi che queste query siano ottimizzate e utilizzino indici appropriati.

  Se le query principali responsabili dell'evento di attesa sono correlate allo stesso oggetto o tabella di database, prendere in considerazione il partizionamento dell'oggetto o della tabella.

Crea repliche di Aurora

È possibile creare repliche di Aurora per gestire il traffico di sola lettura. È inoltre possibile utilizzare Auto Scaling di Aurora per gestire le sovratensioni nel traffico di lettura. Assicurarsi di eseguire attività di sola lettura pianificate e backup logici sulle repliche di Aurora.

Per ulteriori informazioni, consulta Auto Scaling di Amazon Aurora con repliche Aurora.

Analisi delle dimensioni del pool di buffer

Verificare se la dimensione del pool di buffer è sufficiente per il carico di lavoro esaminando il parametro `innodb_buffer_pool_wait_free`. Se il valore di questo parametro è alto e aumenta continuamente, ciò indica che la dimensione del pool di buffer non è sufficiente per gestire il carico di lavoro. Se `innodb_buffer_pool_size` è stato impostato correttamente, il valore di `innodb_buffer_pool_wait_free`dovrebbe essere piccolo. Per ulteriori informazioni, vedere Innodb_buffer_pool_wait_free nella mia documentazione. SQL

Aumentare le dimensioni del pool di buffer se l'istanza database dispone di memoria sufficiente per i buffer di sessione e le attività del sistema operativo. In caso contrario, modificare l'istanza database in una classe di istanza database più grande per ottenere memoria aggiuntiva che può essere allocata al pool di buffer.

> **Note**
>
> Aurora My regola SQL automaticamente il valore di in `innodb_buffer_pool_instances` base alla configurazione. `innodb_buffer_pool_size`

Monitoraggio della cronologia di stato globale

Monitorando i tassi di modifica delle variabili di stato, è possibile rilevare problemi di blocco o di memoria sull'istanza database. Attivare Global Status History (GoSH) se non è già attivo. Per ulteriori informazioni su GoSH, consulta Gestione della cronologia di stato globale.

Puoi anche creare CloudWatch parametri Amazon personalizzati per monitorare le variabili di stato. Per ulteriori informazioni, consulta Pubblicazione di parametri personalizzati.

synch/mutex/innodb/fil_sistema_mutex

L'evento `synch/mutex/innodb/fil_system_mutex` si verifica quando una sessione è in attesa di accedere alla cache di memoria tablespace.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora SQL Le mie versioni 2 e 3

Context

InnoDB utilizza tablespace per gestire l'area di archiviazione per tabelle e file di registro. La cache di memoria di tablespace è una struttura di memoria globale che mantiene le informazioni sulle tablespace. My SQL usa `synch/mutex/innodb/fil_system_mutex` waits per controllare l'accesso simultaneo alla cache di memoria del tablespace.

L'evento `synch/mutex/innodb/fil_system_mutex` indica che attualmente esistono più operazioni che devono recuperare e manipolare le informazioni nella cache di memoria di tablespace per la stessa tablespace.

Probabili cause di aumento delle attese

Quando l'evento `synch/mutex/innodb/fil_system_mutex` appare più che normale, possibilmente indicando un problema di prestazioni, generalmente accade quando sono presenti tutte le seguenti condizioni:

- Un aumento delle operazioni simultanee del linguaggio di manipolazione dei dati (DML) che aggiornano o eliminano i dati nella stessa tabella.
- La tablespace per questa tabella è molto ampia e ha molte pagine di dati.
- Il fattore di riempimento per queste pagine di dati è basso.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Riorganizza tabelle di grandi dimensioni durante le ore non di picco

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Considerare gli eventi di attesa che contribuiscono al carico più elevato per scoprire se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per trovare le SQL interrogazioni responsabili di un carico elevato

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Performance Insights per l'istanza database.

4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella parte inferiore della pagina, scegli Top SQL.

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Un altro modo per scoprire quali query causano un numero elevato di attese `synch/mutex/innodb/fil_system_mutex` è di controllare `performance_schema`, come nel seguente esempio.

```
mysql> select * from performance_schema.events_waits_current where EVENT_NAME='wait/
synch/mutex/innodb/fil_system_mutex'\G
*************************** 1. row ***************************
            THREAD_ID: 19
             EVENT_ID: 195057
         END_EVENT_ID: 195057
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:6700
          TIMER_START: 1010146190118400
            TIMER_END: 1010146196524000
           TIMER_WAIT: 6405600
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 2. row ***************************
            THREAD_ID: 23
             EVENT_ID: 5480
         END_EVENT_ID: 5480
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:5906
          TIMER_START: 995269979908800
```

```
            TIMER_END: 995269980159200
          TIMER_WAIT: 250400
               SPINS: NULL
       OBJECT_SCHEMA: NULL
         OBJECT_NAME: NULL
          INDEX_NAME: NULL
         OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 3. row ***************************
            THREAD_ID: 55
             EVENT_ID: 23233794
         END_EVENT_ID: NULL
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:449
          TIMER_START: 1010492125341600
            TIMER_END: 1010494304900000
          TIMER_WAIT: 2179558400
               SPINS: NULL
       OBJECT_SCHEMA: NULL
         OBJECT_NAME: NULL
          INDEX_NAME: NULL
         OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: 23233786
   NESTING_EVENT_TYPE: WAIT
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

Riorganizza tabelle di grandi dimensioni durante le ore non di picco

Riorganizza tabelle di grandi dimensioni che identifichi come fonte di un numero elevato di eventi di attesa `synch/mutex/innodb/fil_system_mutex` durante una finestra temporale di manutenzione al di fuori dell'orario di produzione. In questo modo si assicura che la pulizia della mappa delle tablespace interne non si verifichi quando l'accesso rapido alla tabella è fondamentale. Per informazioni sulla riorganizzazione delle tabelle, consulta l'OPTIMIZETABLEinformativa nel My Reference. SQL

synch/mutex/innodb/trx_sys_mutex

L'evento `synch/mutex/innodb/trx_sys_mutex` si verifica quando c'è un'elevata attività del database con un numero elevato di transazioni.

Argomenti

- Versioni del motore rilevanti
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore rilevanti

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni del motore:

- Aurora SQL Le mie versioni 2 e 3

Context

Internamente, il motore di database InnoDB utilizza il livello di isolamento della lettura ripetibile con snapshot per fornire coerenza di lettura. In questo modo è possibile point-in-time visualizzare il database al momento della creazione dello snapshot.

In InnoDB, tutte le modifiche vengono applicate al database non appena arrivano, indipendentemente dal fatto che sia stato eseguito il commit. Questo approccio significa che senza il controllo della concorrenza multiversione (MVCC), tutti gli utenti connessi al database visualizzano tutte le modifiche e le righe più recenti. Pertanto, InnoDB richiede un modo per tenere traccia delle modifiche per capire cosa ripristinare allo stato precedente quando necessario.

Per fare ciò, InnoDB utilizza un sistema di transazioni (`trx_sys`) per tenere traccia degli snapshot. Il sistema di transazioni effettua quanto segue:

- Tiene traccia dell'ID della transazione per ogni riga nei registri di annullamento.
- Utilizza una struttura interna di InnoDB chiamata `ReadView` che aiuta a identificare quali transazioni IDs sono visibili per un'istantanea.

Probabili cause di aumento delle attese

Qualsiasi operazione di database che richieda la gestione coerente e controllata (creazione, lettura, aggiornamento ed eliminazione) delle transazioni IDs genera una chiamata dal mutex`trx_sys`.

Queste chiamate avvengono all'interno di tre funzioni:

- `trx_sys_mutex_enter` – Crea il mutex.
- `trx_sys_mutex_exit` – Rilascia il mutex.
- `trx_sys_mutex_own` – Verifica se il mutex è di proprietà.

La strumentazione InnoDB Performance Schema tiene traccia di tutte le chiamate mutex `trx_sys`. Il monitoraggio include, a titolo esemplificativo ma non esaustivo, la gestione di `trx_sys` all'avvio o allo spegnimento del database, operazioni di ripristino dello stato precedente, pulizia degli annullamenti, accesso in lettura di righe e carichi del pool di buffer. L'elevata attività del database con un numero elevato di transazioni comporta la comparsa di `synch/mutex/innodb/trx_sys_mutex` tra i principali eventi di attesa.

Per ulteriori informazioni, consulta Monitoring InnoDB Mutex Waits Using Performance Schema nella mia documentazione. SQL

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare le sessioni e le query che causano gli eventi
- Esamina altri eventi di attesa

Identificare le sessioni e le query che causano gli eventi

In genere, i database con carico da moderato a significativo hanno eventi di attesa. Gli eventi di attesa possono essere accettabili se le prestazioni sono ottimali. Se le prestazioni non sono ottimali, esaminare dove il database impiega più tempo. Guarda gli eventi di attesa che contribuiscono al carico più elevato. Scopri se è possibile ottimizzare il database e l'applicazione per ridurre tali eventi.

Per visualizzare il grafico Top SQL nel AWS Management Console

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.
5. Nella tabella di caricamento del database, scegli Top SQL.

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog Analizza i miei SQL carichi di lavoro di Amazon Aurora con Performance Insights.

Esamina altri eventi di attesa

Esamina gli altri eventi di attesa associati all'evento di attesa `synch/mutex/innodb/trx_sys_mutex`. In questo modo, si possono acquisire ulteriori informazioni sulla natura del carico di lavoro. Un numero elevato di transazioni potrebbe ridurre la velocità effettiva, ma anche il carico di lavoro potrebbe renderlo necessario.

Per ulteriori informazioni su come ottimizzare le transazioni, vedere Ottimizzazione della gestione delle transazioni di InnoDB nella SQL mia documentazione.

synch/sxlock/innodb/hash_serrature da tavolo

L'`synch/sxlock/innodb/hash_table_locks`evento si verifica quando le pagine non trovate nel buffer pool devono essere lette dallo storage.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sull'evento di attesa sono supportate per le seguenti versioni:

- Aurora SQL Le mie versioni 2 e 3

Context

L'evento `synch/sxlock/innodb/hash_table_locks` indica che un carico di lavoro accede frequentemente a dati che non sono memorizzati nel pool di buffer. Questo evento di attesa è associato ad aggiunte di nuove pagine e a espulsioni di dati vecchi dal pool di buffer. I dati memorizzati nei dati nuovi e vecchi del pool di buffer devono essere memorizzati nella cache, in modo che le pagine vecchie vengano espulse per consentire la memorizzazione nella cache delle nuove pagine. My SQL utilizza un algoritmo (LRU) usato meno di recente per rimuovere le pagine dal pool di buffer. Il carico di lavoro sta tentando di accedere ai dati che non sono stati caricati nel pool di buffer o ai dati che sono stati espulsi dal pool di buffer.

Questo evento di attesa si verifica quando il carico di lavoro deve accedere ai dati nei file su disco o quando i blocchi vengono liberati o aggiunti all'elenco del pool di buffer. LRU Queste operazioni attendono di ottenere un blocco condiviso escluso (SX-Lock). Questo SX-Lock viene utilizzato per la sincronizzazione su tabella hash, che è una tabella in memoria progettata per migliorare le prestazioni di accesso al pool di buffer.

Per ulteriori informazioni, consulta Buffer Pool nella documentazione personale. SQL

Probabili cause di aumento delle attese

Quando l'evento di attesa `synch/sxlock/innodb/hash_table_locks` appare più che normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

Un pool di buffer sottodimensionato

La dimensione del pool di buffer è troppo piccola per mantenere in memoria tutte le pagine a cui si accede di frequente.

Carico di lavoro pesante

Il carico di lavoro sta causando frequenti espulsioni e ricariche di pagine di dati nella cache del buffer.

Errori di lettura delle pagine

Ci sono errori di lettura delle pagine nel pool di buffer, che potrebbero indicare il danneggiamento dei dati.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Aumenta le dimensioni del pool del buffer
- Miglioramento dei modelli di accesso ai dati
- Riduci o evita le scansioni complete della tabella
- Controllare i registri degli errori per la presenza del danneggiamento della pagina

Aumenta le dimensioni del pool del buffer

Assicurarsi che il pool del buffer sia ridimensionato in modo appropriato per il carico di lavoro. Per farlo è possibile controllare la percentuale di riscontri nella cache del pool di buffer. In genere, se il valore scende al di sotto del 95%, è consigliabile aumentare la dimensione del pool di buffer. Un pool di buffer più ampio può mantenere più a lungo in memoria le pagine a cui si accede di frequente. Per aumentare le dimensioni del pool di buffer, modificare il valore del parametro `innodb_buffer_pool_size`. Il valore predefinito di questo parametro è basato sulle dimensioni della classe di istanza database. Per ulteriori informazioni, consulta Best practice per la configurazione del mio SQL database di Amazon Aurora.

Miglioramento dei modelli di accesso ai dati

Controlla le query interessate da questa attesa e i relativi piani di esecuzione. Considera la possibilità di migliorare i modelli di accesso ai dati. Ad esempio, se si sta utilizzandomysqli_result:: fetch_array, si può provare ad aumentare la dimensione del recupero dell'array.

È possibile utilizzare Performance Insights per visualizzare query e sessioni che potrebbero causare l'evento di attesa `synch/sxlock/innodb/hash_table_locks`.

Per trovare SQL le query responsabili di un carico elevato

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un'istanza database. Viene visualizzato il pannello di controllo di Approfondimenti sulle prestazioni per l'istanza database.
4. Nel grafico Carico del database, scegli Dividi per attesa.

5. Nella parte inferiore della pagina, scegli Top SQL.

   Il grafico elenca le SQL interrogazioni responsabili del carico. Quelli in cima all'elenco sono le più responsabili. Per risolvere un collo di bottiglia, occorre concentrarsi su queste istruzioni.

Per un'utile panoramica sulla risoluzione dei problemi con Performance Insights, consulta il post del blog AWS Database Analyze Amazon Aurora My SQL Workloads with Performance Insights.

Riduci o evita le scansioni complete della tabella

Monitora il carico di lavoro per verificare se sta eseguendo scansioni a tabella completa e, in caso affermativo, ridurle o evitarle. Ad esempio, è possibile monitorare le variabili di stato come ad esempio `Handler_read_rnd_next`. Per ulteriori informazioni, consulta Server Status Variables nella documentazione personaleSQL.

Controllare i registri degli errori per la presenza del danneggiamento della pagina

È possibile controllare il mysql-error.log per la presenza di messaggi correlati al danneggiamento che sono stati rilevati in prossimità del momento in cui si è verificato il problema. I messaggi con cui è possibile lavorare per risolvere il problema si trovano nel registro degli errori. Potrebbe essere necessario ricreare oggetti segnalati come danneggiati.

Regolazione di Aurora MySQL con stati del thread

La tabella seguente riassume gli stati del thread generali più comuni per Aurora MySQL.

Stato del thread generale	Descrizione
???	Questo stato del thread indica che un thread sta elaborando un'istruzione `SELECT` che richiede l'uso di una tabella temporanea interna per ordinare i dati.
???	Questo stato del thread indica che un thread sta leggendo e filtrando le righe per cercare una query per determinare la serie di risultati corretti.

creazione di indice di ordinamento

Lo stato del thread `creating sort index` indica che un thread sta elaborando un'istruzione `SELECT` che richiede l'uso di una tabella temporanea interna per ordinare i dati.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sullo stato del thread sono supportate per le seguenti versioni:

- Aurora My SQL versione 2 fino alla 2.09.2

Context

Lo stato `creating sort index` appare quando una query con una clausola `ORDER BY` o `GROUP BY` non può utilizzare un indice esistente per eseguire l'operazione. In questo caso, My SQL deve

eseguire un'operazione più costosa. `filesort` Questa operazione viene generalmente eseguita in memoria se la serie di risultati non è troppo grande. In caso contrario, comporta la creazione di un file su disco.

Probabili cause di aumento delle attese

La comparsa di `creating sort index` non indica di per sé un problema. Se le prestazioni sono scadenti e si vedono frequenti istanze di `creating sort index`, la causa più probabile è query lente con operatori `ORDER BY` o `GROUP BY`.

Azioni

La linea guida generale è trovare query con clausole `ORDER BY` o `GROUP BY` associate agli aumenti dello stato `creating sort index`. Quindi verificare se l'aggiunta di un indice o l'aumento della dimensione del buffer di ordinamento risolve il problema.

Argomenti

- Attiva il Performance Schema se non è attivato
- Identificare le query problematiche
- Esaminare i piani di spiegazione per l'utilizzo di filesort
- Aumenta la dimensione del buffer di ordinamento

Attiva il Performance Schema se non è attivato

Performance Insights segnala gli stati del thread solo se gli strumenti di Performance Schema non sono attivati. Quando gli strumenti di Performance Schema sono attivati, Performance Insights segnala invece gli eventi di attesa. Gli strumenti di Performance Schema forniscono informazioni dettagliate aggiuntive e strumenti migliori quando si esaminano potenziali problemi di prestazione. Pertanto, è consigliabile attivare il Performance Schema. Per ulteriori informazioni, consulta Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

Identificare le query problematiche

Per identificare le query correnti che causano un aumento dello stato `creating sort index`, eseguire `show processlist` e vedi se una qualsiasi delle query ha `ORDER BY` o `GROUP BY`. Facoltativamente, eseguire `explain for connection N`, dove `N` è l'ID dell'elenco dei processi della query con `filesort`.

Per identificare le query precedenti che causano questi aumenti, attiva il registro delle query lente e trova le query con `ORDER BY`. Esegui `EXPLAIN` sulle query lente e cerca “utilizzo di filesort”. Per ulteriori informazioni, consulta Esaminare i piani di spiegazione per l'utilizzo di filesort.

Esaminare i piani di spiegazione per l'utilizzo di filesort

Identifica le istruzioni con clausole `ORDER BY` o `GROUP BY` che si traducono nello stato `creating sort index`.

Negli esempi seguenti viene illustrato come eseguire `explain` su una query. La colonna `Extra` mostra che questa query utilizza `filesort`.

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2064548
     filtered: 100.00
        Extra: Using filesort
1 row in set, 1 warning (0.01 sec)
```

L'esempio seguente mostra il risultato dell'esecuzione di `EXPLAIN` sulla stessa query dopo la creazione di un indice sulla colonna `c1`.

```
mysql> alter table mytable add index (c1);
```

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: index
possible_keys: NULL
          key: c1
```

```
      key_len: 1023
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: Using index
1 row in set, 1 warning (0.01 sec)
```

Per informazioni sull'utilizzo degli indici per l'ottimizzazione degli ordinamenti, vedere ORDERBY Optimization nella documentazione personaleSQL.

Aumenta la dimensione del buffer di ordinamento

Per verificare se una query specifica richiedeva un processo `filesort` che ha creato un file su disco, controllare il valore della variabile `sort_merge_passes` dopo aver eseguito la query. Di seguito viene riportato un esempio.

```
mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)

--- run query
mysql> select * from mytable order by u limit 10;
--- run status again:

mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)
```

Se il valore di `sort_merge_passes` è alto, si consideri di aumentare la dimensione del buffer di ordinamento. Applica l'aumento a livello di sessione, poiché aumentandolo a livello globale può aumentare significativamente la quantità di I RAM miei SQL utilizzi. L'esempio seguente mostra come modificare le dimensioni del buffer di ordinamento prima di eseguire una query.

```
mysql> set session sort_buffer_size=10*1024*1024;
```

```
Query OK, 0 rows affected (0.00 sec)
-- run query
```

invio dei dati

Lo stato del thread `sending data` indica che un thread sta leggendo e filtrando le righe per una query per determinare la serie di risultati corretti. Il nome è fuorviante perché implica che lo stato sta trasferendo i dati, non raccogliendo e preparando i dati da inviare in seguito.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni sullo stato del thread sono supportate per le seguenti versioni:

- Aurora MySQL versione 2 fino alla 2.09.2

Context

Molti stati del thread sono di breve durata. Operazioni che si verificano durante `sending data` tendono a eseguire un numero elevato di letture su disco o cache. Pertanto, `sending data` è spesso lo stato di esecuzione più lungo per tutta la durata di una determinata query. Questo stato viene visualizzato quando Aurora MySQL sta effettuando le seguenti operazioni:

- Lettura ed elaborazione di righe per un'istruzione `SELECT`
- Esecuzione di un numero elevato di letture da disco o memoria
- Completamento di una lettura completa di tutti i dati da una query specifica
- Lettura di dati da una tabella, da un indice o dal lavoro di una procedura archiviata
- Ordinamento, raggruppamento o ordinamento dei dati

Dopo che lo stato `sending data` termina la preparazione dei dati, lo stato del thread `writing to net` indica il ritorno dei dati al client. Tipicamente `writing to net` viene acquisito solo quando la serie di risultati è molto grande o una latenza di rete grave rallenta il trasferimento.

Probabili cause di aumento delle attese

La comparsa di `sending data` non indica di per sé un problema. Se le prestazioni sono scadenti e si vedono frequenti istanze di `sending data`, le cause più probabili sono le seguenti.

Argomenti

- Query inefficiente
- Configurazione del server non ottimale

Query inefficiente

Nella maggior parte dei casi, ciò che è responsabile di questo stato è una query che non utilizza un indice appropriato per trovare la serie di risultati di una query specifica. Ad esempio, si consideri una query che legge una tabella di dati di 10 milioni per tutti gli ordini effettuati in California, in cui la colonna di stato non è indicizzata o è scarsamente indicizzata. In quest'ultimo caso, l'indice potrebbe esistere, ma l'ottimizzatore lo ignora a causa della bassa cardinalità.

Configurazione del server non ottimale

Se vengono visualizzate diverse query nello stato `sending data`, il server di database potrebbe essere configurato in modo errato. In particolare, il server potrebbe presentare i seguenti problemi:

- Il server di database non dispone di capacità di calcolo sufficiente: I/O del disco, tipo e velocità del disco, CPU o numero di CPU.
- Il server necessita delle risorse allocate, come il pool di buffer di InnoDB per le tabelle InnoDB o il buffer di chiavi per le tabelle MyIsam.
- Impostazioni di memoria per thread come `sort_buffer`, `read_buffer` `ejoin_buffer` consumano più RAM di quanto richiesto, portando il server fisico a necessitare risorse di memoria.

Operazioni

La linea guida generale consiste nel trovare query che restituiscono un numero elevato di righe controllando il Performance Schema. Se la registrazione di query che non utilizzano indici è attivata, è anche possibile esaminare i risultati dei registri lenti.

Argomenti

- Attiva il Performance Schema se non è attivato
- Analisi delle impostazioni della memoria
- Esaminare i piani di spiegazione per l'utilizzo dell'indice
- Controllare il volume di dati restituiti
- Verifica la presenza di problemi di concorrenza
- Verificare la struttura delle query

Attiva il Performance Schema se non è attivato

Performance Insights segnala gli stati del thread solo se gli strumenti di Performance Schema non sono attivati. Quando gli strumenti di Performance Schema sono attivati, Performance Insights segnala invece gli eventi di attesa. Gli strumenti di Performance Schema forniscono informazioni dettagliate aggiuntive e strumenti migliori quando si esaminano potenziali problemi di prestazione. Pertanto, è consigliabile attivare il Performance Schema. Per ulteriori informazioni, consulta Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

Analisi delle impostazioni della memoria

Esaminare le impostazioni di memoria per i pool di buffer primari. Assicurarsi che questi pool siano ridimensionati in modo appropriato per il carico di lavoro. Se il database utilizza più istanze del pool di buffer, assicurati che non siano divisi in molti piccoli pool di buffer. I thread possono utilizzare solo un pool di buffer alla volta.

Assicurarsi che le seguenti impostazioni di memoria utilizzate per ciascun thread siano dimensionate correttamente:

- read_buffer
- read_rnd_buffer
- sort_buffer
- join_buffer
- binlog_cache

A meno che non si abbiano ragioni specifiche per modificare le impostazioni, utilizzare i valori predefiniti.

Esaminare i piani di spiegazione per l'utilizzo dell'indice

Per queri nello stato del thread `sending data`, esaminare il piano per determinare se vengono utilizzati indici appropriati. Se una query non utilizza un indice utile, si prenda in considerazione l'aggiunta di suggerimenti come `USE INDEX` o `FORCE INDEX`. I suggerimenti possono aumentare o diminuire notevolmente il tempo necessario per eseguire una query, quindi occorre prestare attenzione prima di aggiungerli.

Controllare il volume di dati restituiti

Controllare le tabelle sottoposte a query e la quantità di dati che contengono. È possibile archiviare qualcuno di questi dati? In molti casi, la causa di tempi di esecuzione delle query scadenti non è il risultato del piano della query, ma il volume di dati da elaborare. Molti sviluppatori sono molto efficienti nell'aggiungere dati a un database, ma raramente considerano il ciclo di vita del set di dati nelle fasi di progettazione e sviluppo.

Si consiglia di cercare query che funzionino bene nei database con volumi ridotti ma che funzionino male nel sistema attuale. A volte gli sviluppatori che progettano query specifiche potrebbero non rendersi conto che queste query restituiscono 350.000 righe. Gli sviluppatori potrebbero aver sviluppato le query in un ambiente con volumi inferiori con set di dati più piccoli rispetto agli ambienti di produzione.

Verifica la presenza di problemi di concorrenza

Verifica se sono in esecuzione contemporaneamente più query dello stesso tipo. Alcune forme di query vengono eseguite in modo efficiente quando vengono eseguite da sole. Tuttavia, se forme di query simili vengono eseguite insieme o in volume elevato, possono causare problemi di concorrenza. Spesso questi problemi sono causati quando il database utilizza tabelle temporanee per restituire dei risultati. Un livello di isolamento restrittivo delle transazioni può anche causare problemi di concorrenza.

Se le tabelle vengono lette e scritte contemporaneamente, il database potrebbe utilizzare i blocchi. Per aiutare a identificare i periodi di prestazioni scadenti, esaminare l'uso dei database attraverso processi batch su larga scala. Per visualizzare i blocchi e i ripristini degli stati precedenti recenti, esaminare l'output del comando `SHOW ENGINE INNODB STATUS`.

Verificare la struttura delle query

Verifica se le query acquisite da questi stati utilizzano sottoquery. Questo tipo di query spesso porta a prestazioni scadenti perché il database compila i risultati internamente e li sostituisce nuovamente

nella query per la restituzione dei dati. Questo processo è un passaggio aggiuntivo per il database. In molti casi, questo passaggio può causare prestazioni scadenti in condizioni di caricamento altamente concomitante.

Controlla anche se le tue query utilizzano un numero elevato di clausole `ORDER BY` e `GROUP BY`. In tali operazioni, spesso il database deve prima formare l'intero set di dati in memoria. Quindi deve ordinarlo o raggrupparlo in modo specifico prima di restituirlo al client.

Ottimizzazione di Aurora MySQL con approfondimenti proattivi di Amazon DevOps Guru

Gli approfondimenti proattivi di DevOps Guru rilevano le condizioni problematiche note nel cluster di database Aurora MySQL prima che si verifichino. Con DevOps Guru è possibile:

- Evitare molti problemi comuni relativi al database controllando la configurazione del database rispetto alle impostazioni consigliate comuni.
- Ricevere gli avvisi per le criticità relative al parco istanze che, se non controllate, possono portare a problemi più gravi in seguito.
- Ricevere gli avvisi per i nuovi problemi individuati.

Ogni approfondimento proattivo contiene un'analisi della causa del problema e i suggerimenti per le azioni correttive.

Argomenti

- La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo
- Il database sta creando tabelle temporanee su disco

La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo

A partire da *`date`* allora, l'elenco della cronologia delle modifiche alle righe è aumentato in modo significativo, fino a un *`length`* certo punto*`db-instance`*. Questo aumento influisce sulle prestazioni di chiusura delle query e arresto del database.

Argomenti

- Versioni del motore supportate
- Context

- Probabili cause di questo problema
- Operazioni
- Parametri rilevanti

Versioni del motore supportate

Queste informazioni approfondite sono supportate per tutte le versioni di Aurora MySQL.

Context

Il sistema di transazioni InnoDB gestisce il controllo della concorrenza multiversione (MVCC). Quando una riga viene modificata, la versione precedente alla modifica dei dati da modificare viene archiviata come record di annullamento in un log di annullamenti. Ogni record di annullamento ha un riferimento al record di ripristino precedente, formando un elenco collegato.

L'elenco della cronologia di InnoDB è un elenco globale dei log di annullamento per le transazioni sottoposte a commit. MySQL utilizza l'elenco della cronologia per eliminare i record e le pagine di log quando le transazioni non richiedono più la cronologia. La lunghezza dell'elenco della cronologia è il numero totale di log di annullamento che contengono modifiche nell'elenco della cronologia. Ogni log contiene una o più modifiche. Se la lunghezza dell'elenco della cronologia di InnoDB aumenta in modo significativo, indicando un numero elevato di precedenti versioni di riga, la chiusura delle query e l'arresto del database diventano più lenti.

Probabili cause di questo problema

Le cause tipiche di un lungo elenco della cronologia sono:

- Transazioni di lunga durata, in lettura o scrittura
- Carico elevato in scrittura

Operazioni

Consigliamo azioni diverse a seconda delle cause degli approfondimenti.

Argomenti

- Non iniziare alcuna operazione che comporti l'arresto del database finché non diminuiscono le dimensioni dell'elenco della cronologia di InnoDB
- Identificare e terminare le transazioni di lunga durata

- Usa Approfondimenti sulle prestazioni per individuare i principali host e i migliori utenti.

Non iniziare alcuna operazione che comporti l'arresto del database finché non diminuiscono le dimensioni dell'elenco della cronologia di InnoDB

Poiché un lungo elenco della cronologia di InnoDB rallenta l'arresto del database, riduci le dimensioni dell'elenco prima di iniziare le relative operazioni. Queste operazioni includono gli aggiornamenti della versione principale del database.

Identificare e terminare le transazioni di lunga durata

Puoi individuare le transazioni di lunga durata eseguendo la query `information_schema.innodb_trx`.

> **Note**
>
> Assicurati inoltre di cercare le transazioni di lunga durata nelle repliche di lettura.

Per identificare e terminare le transazioni di lunga durata

1. Nel client SQL esegui la seguente query:

```
SELECT a.trx_id,
      a.trx_state,
      a.trx_started,
      TIMESTAMPDIFF(SECOND,a.trx_started, now()) as "Seconds Transaction Has Been
 Open",
      a.trx_rows_modified,
      b.USER,
      b.host,
      b.db,
      b.command,
      b.time,
      b.state
FROM  information_schema.innodb_trx a,
      information_schema.processlist b
WHERE a.trx_mysql_thread_id=b.id
  AND TIMESTAMPDIFF(SECOND,a.trx_started, now()) > 10
ORDER BY trx_started
```

2. Termina ogni transazione di lunga durata con la procedura memorizzata. mysql.rds_kill

Usa Approfondimenti sulle prestazioni per individuare i principali host e i migliori utenti.

Ottimizza le transazioni in modo che un numero elevato di righe modificate venga immediatamente sottoposto a commit.

Parametri rilevanti

A questo approfondimento è correlato il seguente parametro:

- `trx_rseg_history_len`— Questa metrica contatrice può essere visualizzata in Performance Insights, oltre che nella `INFORMATION_SCHEMA.INNODB_METRICS` tabella. Per ulteriori informazioni, vedere la tabella delle metriche InnoDB INFORMATION_SCHEMA nella documentazione MySQL.
- `RollbackSegmentHistoryListLength`— Questa CloudWatch metrica di Amazon misura i log di annullamento che registrano le transazioni impegnate con record contrassegnati da eliminare. Questi record sono pianificati per essere elaborati dall'operazione di rimozione InnoDB. La metrica `trx_rseg_history_len` ha lo stesso valore di. `RollbackSegmentHistoryListLength`
- `PurgeBoundary`— Il numero di transazione fino al quale è consentito l'eliminazione da InnoDB. Se questa CloudWatch metrica non avanza per lunghi periodi di tempo, è una buona indicazione che l'eliminazione di InnoDB è bloccata da transazioni di lunga durata. Per indagare, controlla le transazioni attive sul tuo cluster Aurora MySQL DB. Questa metrica è disponibile solo per Aurora MySQL versione 2.11 e successive e versione 3.08 e successive.
- `PurgeFinishedPoint`— Il numero di transazione fino al quale viene eseguita l'eliminazione da InnoDB. Questa CloudWatch metrica può aiutarti a esaminare la velocità con cui procede l'eliminazione da InnoDB. Questa metrica è disponibile solo per Aurora MySQL versione 2.11 e successive e versione 3.08 e successive.
- `TransactionAgeMaximum`— L'età della transazione attiva più vecchia in esecuzione. Questa CloudWatch metrica è disponibile solo per Aurora MySQL versione 3.08 e successive.
- `TruncateFinishedPoint`— Il numero di transazione fino al quale viene eseguito il troncamento dell'annullamento. Questa CloudWatch metrica è disponibile solo per Aurora MySQL versione 2.11 e successive e versione 3.08 e successive.

Per ulteriori informazioni sulle metriche, consulta. CloudWatch Parametri a livello di istanza per Amazon Aurora

Il database sta creando tabelle temporanee su disco

L'utilizzo recente della tabella temporanea su disco è aumentato in modo significativo, fino al valore di *percentage*. Il database crea le tabelle temporanee al secondo come specificato in *number*. Ciò potrebbe influire sulle prestazioni e aumentare le operazioni su disco in *db-instance*.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di questo problema
- Operazioni
- Parametri rilevanti

Versioni del motore supportate

Queste informazioni approfondite sono supportate per tutte le versioni di Aurora MySQL.

Context

A volte è necessario che il server MySQL crei una tabella temporanea interna durante l'elaborazione di una query. Aurora MySQL può contenere una tabella temporanea interna in memoria, dove può essere elaborata dal motore di archiviazione TempTable o MEMORY o archiviata su disco da InnoDB. Per ulteriori informazioni, consulta Uso della tabella temporanea interna in MySQL nel Manuale di riferimento di MySQL.

Probabili cause di questo problema

Un aumento delle tabelle temporanee su disco indica l'uso di query complesse. Se la memoria configurata non è sufficiente per archiviare le tabelle temporanee in memoria, Aurora MySQL crea le tabelle su disco. Ciò può influire sulle prestazioni e aumentare le operazioni del disco.

Operazioni

Consigliamo azioni diverse a seconda delle cause degli approfondimenti.

- Per Aurora MySQL versione 3, consigliamo di utilizzare il motore di archiviazione TempTable.
- Ottimizza le query in modo da restituire meno dati selezionando solo le colonne necessarie.

Se attivi lo schema delle prestazioni con tutti gli strumenti `statement` abilitati e temporizzati, puoi eseguire la query `SYS.statements_with_temp_tables` per recuperare l'elenco delle query che utilizzano le tabelle temporanee. Per ulteriori informazioni, consulta Prerequisiti per l'utilizzo dello schema sys nella documentazione di MySQL.

- Prendi in considerazione l'indicizzazione delle colonne coinvolte nelle operazioni di ordinamento e raggruppamento.
- Riscrivi le query per evitare le colonne `BLOB` e `TEXT`. Queste colonne utilizzano sempre il disco.
- Ottimizza i parametri del database `tmp_table_size` e `max_heap_table_size`.

  Il valore predefinito di questi parametri è 16 MiB. Quando si utilizza il motore di archiviazione MEMORY per tabelle temporanee in memoria, la dimensione massima è definita dal valore `tmp_table_size` o `max_heap_table_size`, a seconda di quale sia il più piccolo. Quando viene raggiunta questa dimensione massima, MySQL converte automaticamente la tabella temporanea interna in memoria in una tabella temporanea interna di InnoDB su disco. Per ulteriori informazioni, consulta Uso del motore di archiviazione TempTable in Amazon RDS per MySQL e Amazon Aurora MySQL.

  > **Note**
  >
  > Quando si creano in modo esplicito tabelle MEMORY con CREATE TABLE, solo la variabile `max_heap_table_size` determina la dimensione massima di una tabella. Inoltre, non è prevista la conversione in un formato su disco.

Parametri rilevanti

A questo approfondimento sono correlati i seguenti parametri di Approfondimenti sulle prestazioni:

- Created_tmp_disk_tables
- Created_tmp_tables

Per ulteriori informazioni, consulta Created_tmp_disk_tables nella documentazione di MySQL.

Interrogazione parallela per Amazon Aurora My SQL

Questo argomento descrive l'ottimizzazione delle prestazioni delle query parallele per Amazon Aurora My SQL -Compatible Edition. Questa funzionalità utilizza un percorso di elaborazione speciale per alcune query che implicano grandi quantità di dati grazie all'architettura di storage condivisa di Aurora. Le query parallele funzionano al meglio con i cluster Aurora My SQL DB che dispongono di tabelle con milioni di righe e query analitiche che richiedono minuti o ore per essere completate.

Argomenti

- Panoramica delle interrogazioni parallele per Aurora My SQL
- Creazione di un cluster DB di query parallele in Aurora My SQL
- Attivazione e disattivazione delle interrogazioni parallele in Aurora My SQL
- Ottimizzazione delle interrogazioni parallele in Aurora My SQL
- Verifica delle istruzioni che utilizzano la query parallela per Aurora My SQL
- Monitoraggio delle interrogazioni parallele per Aurora My SQL
- SQLcostrutti per interrogazioni parallele in Aurora My SQL

Panoramica delle interrogazioni parallele per Aurora My SQL

Aurora My parallel query è un'ottimizzazione che SQL parallelizza alcuni degli I/O e del calcolo coinvolti nell'elaborazione di query a uso intensivo di dati. Il lavoro che viene parallelizzato include il recupero di righe dalla memoria, l'estrazione dei valori della colonna e la determinazione di quali righe corrispondono alle condizioni nelle clausole join e nella clausola `WHERE`. Questa attività che implicano grandi quantità di dati sono delegate (in termini di ottimizzazione del database, trasferite) a più nodi nel livello di storage distribuito di Aurora. Senza query parallele, ogni query porta tutti i dati scansionati su un singolo nodo all'interno del cluster Aurora SQL My (il nodo principale) e lì esegue tutta l'elaborazione delle query.

> **Tip**
>
> Il motore di SQL database Postgre ha anche una funzionalità chiamata «query parallela». Tale caratteristica non è correlata alla query Aurora parallela.

Quando la funzionalità di interrogazione parallela è attivata, il SQL motore Aurora My determina automaticamente quando le query possono trarne vantaggio, senza richiedere SQL modifiche

come suggerimenti o attributi di tabella. Nelle sezioni seguenti, viene descritto quando una query in parallelo è applicata a una query. Scoprirai inoltre come assicurarti che una query in parallelo sia applicata dove necessario.

> **Note**
>
> L'ottimizzazione delle query parallele offre il massimo vantaggio per le query a esecuzione prolungata che richiedono minuti o ore per il completamento. Aurora My SQL in genere non esegue l'ottimizzazione delle query parallele per query poco costose. In genere, inoltre, non esegue l'ottimizzazione delle query parallele se un'altra tecnica di ottimizzazione ha più senso, come la memorizzazione nella cache delle query, la memorizzazione nella cache del pool di buffer o le ricerche degli indici. Consulta se ritieni che le query in parallelo non sono utilizzate dove dovrebber Verifica delle istruzioni che utilizzano la query parallela per Aurora My SQL.

Argomenti

- Vantaggi
- Architettura
- Prerequisiti
- Limitazioni
- Costi di I/O con la query parallela

Vantaggi

Con la query parallela, puoi eseguire query analitiche a uso intensivo di dati su Aurora My tables. SQL In molti casi, è possibile ottenere un miglioramento order-of-magnitude delle prestazioni rispetto alla tradizionale divisione del lavoro per l'elaborazione delle query.

Alcuni dei vantaggi derivanti dall'uso delle query in parallelo sono:

- Miglioramento delle prestazioni di I/O a seguito delle parallelizzazione delle richieste di lettura fisiche in molteplici nodi di storage.
- Traffico di rete ridotto. Aurora non trasmette intere pagine di dati dai nodi di storage al nodo head e quindi non filtra le righe e le colonne superflue in seguito. Aurora trasmette invece tuple compatte contenenti solo i valori di colonna necessari per il set di risultati.

- CPUUtilizzo ridotto sul nodo principale, grazie alla riduzione dell'elaborazione delle funzioni, del filtraggio delle righe e della proiezione delle colonne per la clausola. `WHERE`
- Riduzione della sollecitazione della memoria a livello del pool di buffer. Le pagine elaborate dalla query parallela non vengono aggiunte al pool di buffer. Questo approccio riduce la possibilità che una scansione intensiva di dati sgomberi i dati utilizzati di frequente dal pool di buffer.
- Potenzialmente ridotta la duplicazione dei dati nella pipeline extract, transform, load (ETL), rendendo pratica l'esecuzione di query analitiche di lunga durata sui dati esistenti.

Architettura

La funzionalità di interrogazione parallela utilizza i principali principi architettonici di Aurora MySQL: disaccoppiamento del motore di database dal sottosistema di archiviazione e riduzione del traffico di rete semplificando i protocolli di comunicazione. Aurora My SQL utilizza queste tecniche per velocizzare le operazioni ad alta intensità di scrittura come l'elaborazione dei redo log. La query in parallelo applica gli stessi principi alle operazioni di lettura.

> **Note**
>
> L'architettura di Aurora My SQL parallel query è diversa da quella di funzionalità con nomi simili presenti in altri sistemi di database. Aurora La mia query SQL parallela non prevede il multiprocessing simmetrico (SMP) e quindi non dipende dalla CPU capacità del server di database. L'elaborazione parallela avviene nel livello di archiviazione, indipendentemente dal SQL server Aurora My che funge da coordinatore delle query.

Per impostazione predefinita, senza le query in parallelo, l'elaborazione di una query Aurora comporta la trasmissione di dati non elaborati a un singolo nodo nel cluster Aurora (il nodo head). Aurora esegue quindi tutte le ulteriori elaborazioni per quella query in un singolo thread su quel singolo nodo. Con le query parallele, gran parte di questo lavoro ad alta intensità CPU di I/O viene delegato ai nodi del livello di storage. Solo le righe compatte del set di risultati sono ritrasmesse al nodo head, con righe già filtrate e valori di colonna già estratti e trasformati. Il vantaggio in termini di prestazioni deriva dalla riduzione del traffico di rete, dalla riduzione dell'CPUutilizzo sul nodo principale e dalla parallelizzazione dell'I/O tra i nodi di storage. Il volume di operazioni di filtraggio, proiezione e I/O in parallelo non dipende dal numero di istanze database nel cluster Aurora che esegue la query.

Prerequisiti

L'utilizzo di tutte le funzionalità di query parallela richiede un cluster Aurora My SQL DB con versione 2.09 o successiva. Se si dispone già di un cluster che si desidera utilizzare con query parallela, è possibile aggiornarlo a una versione compatibile e abilitare la query parallela in seguito. In questo caso, assicurarsi di seguire la procedura di aggiornamento in Considerazioni relative agli aggiornamenti per le query in parallelo perché i nomi delle impostazioni di configurazione e i valori predefiniti sono diversi in queste versioni più recenti.

Le istanze database nel cluster devono utilizzare le classi di istanza `db.r*`.

Assicurati che l'ottimizzazione del join hash sia attivata per il cluster. Per scoprire come, consulta Abilitazione dell'hash join per cluster di query parallele.

Per personalizzare parametri quali `aurora_parallel_query` e `aurora_disable_hash_join`, è necessario disporre di un gruppo di parametri personalizzato da utilizzare con il cluster. È possibile specificare questi parametri singolarmente per ogni istanza DB utilizzando un gruppo di parametri DB. Tuttavia, si consiglia di specificarli in un gruppo di parametri cluster DB. In questo modo, tutte le istanze DB nel cluster ereditano le stesse impostazioni per questi parametri.

Limitazioni

Le seguenti limitazioni si applicano alla funzionalità di query in parallelo:

- La query parallela non è supportata con Aurora I/O-Optimized Configurazione dell'archiviazione del cluster DB.
- Non è possibile utilizzare la query parallela con le classi di istanza db.t2 o db.t3. Questa limitazione si applica anche se si richiede la query parallela utilizzando la variabile di sessione `aurora_pq_force`.
- La query parallela non si applica alle tabelle che utilizzano i formati di riga `COMPRESSED` o `REDUNDANT`. Utilizzare i formati di riga `COMPACT` o `DYNAMIC` per le tabelle che si intende utilizzare con la query parallela.
- Aurora utilizza un algoritmo basato sui costi per determinare se utilizzare il meccanismo di interrogazione parallela per ogni istruzione. SQL L'utilizzo di determinati SQL costrutti in un'istruzione può impedire la query parallela o rendere improbabile la query parallela per quell'istruzione. Per informazioni sulla compatibilità dei SQL costrutti con la query parallela, vedereSQLcostrutti per interrogazioni parallele in Aurora My SQL.

- Ogni istanza database Aurora può eseguire solo un numero specifico di sessioni di query in parallelo simultanee. Se una query comporta più parti che utilizzano una query in parallelo, come sottoquery, join o operatori `UNION`, quelle fasi sono eseguite in sequenza. L'istruzione viene considerata solo come singola sessione di query in parallelo. Puoi monitorare il numero di sessioni attive mediante le variabili di stato di query in parallelo. Puoi verificare il limite delle sessioni simultanee per una determinata istanza database eseguendo una query sulla variabile di stato `Aurora_pq_max_concurrent_requests`.
- L'interrogazione parallela è disponibile in tutto ciò Regioni AWS che Aurora supporta. Per la maggior parte Regioni AWS, la SQL versione minima richiesta di Aurora My per utilizzare la query parallela è la 2.09.
- La query parallela è progettata per migliorare le prestazioni delle query a uso intensivo di dati. Non è progettata per eseguire query leggere.
- Si consiglia di utilizzare i nodi reader per le SELECT istruzioni, in particolare quelle che richiedono un uso intensivo di dati.

Costi di I/O con la query parallela

Se il tuo SQL cluster Aurora My utilizza una query parallela, potresti notare un aumento `VolumeReadIOPS` dei valori. Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare un aumento delle operazioni di lettura e degli addebiti associati.

I costi di I/O delle query parallele vengono misurati a livello di archiviazione e sono uguali o maggiori con la query parallela attivata. Il vantaggio sta nel miglioramento delle prestazioni delle query. I motivi per i quali i costi di I/O sono potenzialmente più elevati con la query parallela sono due:

- Anche se alcuni dei dati di una tabella si trovano nel pool di buffer, la query parallela richiede che tutti i dati vengano scansionati a livello di archiviazione, con conseguenti costi di I/O.
- L'esecuzione di una query parallela non prepara il pool di buffer. Di conseguenza, le successive esecuzioni della stessa query parallela comportano l'intero costo di I/O.

Creazione di un cluster DB di query parallele in Aurora My SQL

Per creare un SQL cluster Aurora My con query parallela, aggiungervi nuove istanze o eseguire altre operazioni amministrative, si utilizzano le stesse AWS Management Console AWS CLI tecniche utilizzate con altri cluster Aurora My. SQL Puoi creare un cluster che sia compatibile con le query

in parallelo. È inoltre possibile creare un cluster DB per utilizzare le query parallele eseguendo il ripristino da un'istantanea di un cluster Aurora DB SQL compatibile con My. Se non conosci il processo di creazione di un nuovo SQL cluster Aurora My, puoi trovare informazioni di base e prerequisiti in. Creazione di un cluster database Amazon Aurora

Quando scegli una versione SQL del motore Aurora My, ti consigliamo di scegliere l'ultima versione disponibile. Attualmente, tutte le SQL versioni di Aurora My disponibili supportano la query parallela. Hai una maggiore flessibilità per attivare e disattivare la query parallela o utilizzare la query parallela con i cluster esistenti, se utilizzi le versioni più recenti.

Sia che si crei un nuovo cluster o si ripristini da un'istantanea, si utilizzano le stesse tecniche per aggiungere nuove istanze DB utilizzate con altri cluster Aurora My. SQL

Puoi creare un cluster di query parallelo utilizzando la RDS console Amazon o il AWS CLI.

Indice

- Creazione di un cluster abilitato per le query in parallelo mediante la console
- Creazione di un cluster di query parallelo utilizzando CLI

Creazione di un cluster abilitato per le query in parallelo mediante la console

Per creare un cluster abilitato per le query in parallelo mediante la console, procedi come descritto di seguito.

Per creare un cluster di query parallelo con AWS Management Console

1. Seguire la AWS Management Console procedura generale riportata inCreazione di un cluster database Amazon Aurora.
2. Per Tipo di motore, scegli Aurora My. SQL
3. Per Configurazione aggiuntiva, scegli un gruppo di parametri creato per il gruppo di parametri cluster di database. L'utilizzo di tale gruppo di parametri personalizzato è necessario per Aurora My SQL 2.09 e versioni successive. Nel gruppo di parametri del cluster DB, specificare le impostazioni dei parametri `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF`. In questo modo si attiva la query parallela per il cluster e si abilita l'ottimizzazione dell'hash join che funziona in combinazione con la query parallela.

Per verificare che un nuovo cluster è compatibile con le query in parallelo

1. Creare un cluster utilizzando la tecnica precedente.
2. (Per Aurora My SQL versione 2 o 3) Verifica che l'impostazione di `aurora_parallel_query` configurazione sia vera.

```
mysql> select @@aurora_parallel_query;
+-------------------------+
| @@aurora_parallel_query |
+-------------------------+
|                       1 |
+-------------------------+
```

3. (Per Aurora My SQL versione 2) Verifica che l'`aurora_disable_hash_join`impostazione sia falsa.

```
mysql> select @@aurora_disable_hash_join;
+----------------------------+
| @@aurora_disable_hash_join |
+----------------------------+
|                          0 |
+----------------------------+
```

4. Con alcune tabelle di grandi dimensioni e query a uso intensivo di dati, controllare i piani di query per confermare che alcune query utilizzano l'ottimizzazione delle query parallele. A tale scopo, segui la procedura in Verifica delle istruzioni che utilizzano la query parallela per Aurora My SQL.

Creazione di un cluster di query parallelo utilizzando CLI

È possibile creare un nuovo cluster di query parallele con la procedura CLI descritta di seguito.

Per creare un cluster di query parallelo con AWS CLI

1. (Facoltativo) Verifica quali SQL versioni di Aurora My sono compatibili con i cluster di query paralleli. Per fare ciò, utilizzare il comando `describe-db-engine-versions` e controllare il valore del campo `SupportsParallelQuery`. Per un esempio, consulta Verifica della compatibilità di Aurora My SQL version per le query parallele.
2. (Facoltativo) Creare un gruppo di parametri cluster DB personalizzato con le impostazioni `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF`. Utilizzare comandi come i seguenti.

```
aws rds create-db-cluster-parameter-group --db-parameter-group-family aurora-
mysql8.0 --db-cluster-parameter-group-name pq-enabled-80-compatible
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
enabled-80-compatible \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
enabled-80-compatible \
  --parameters
 ParameterName=aurora_disable_hash_join,ParameterValue=OFF,ApplyMethod=pending-
reboot
```

Se si esegue questo passaggio, specificare l'opzione `--db-cluster-parameter-group-name` *`my_cluster_parameter_group`* nell'istruzione `create-db-cluster` successiva. Sostituire il nome del proprio gruppo di parametri. Se si omette questo passaggio, si crea il gruppo di parametri e lo si associa al cluster in seguito, come descritto in Attivazione e disattivazione delle interrogazioni parallele in Aurora My SQL.

3. Segui la AWS CLI procedura generale riportata in. Creazione di un cluster database Amazon Aurora
4. Specificare il set di opzioni seguente:

   - Per l'opzione `--engine`, utilizzare `aurora-mysql`. Questi valori producono cluster di query paralleli compatibili con My SQL 5.7 o 8.0.
   - Per l' opzione `--db-cluster-parameter-group-name`, specificare il nome di un gruppo di parametri cluster DB creato e specificato per il valore del parametro `aurora_parallel_query=ON`. Se si omette questa opzione, è possibile creare il cluster con un gruppo di parametri predefinito e successivamente modificarlo per utilizzare tale gruppo di parametri personalizzato.
   - Per l'`--engine-version`opzione, usa una SQL versione di Aurora My compatibile con la query parallela. Utilizzare la procedura da Ottimizzazione delle interrogazioni parallele in Aurora My SQL per ottenere un elenco di versioni, se necessario.

     Il codice di esempio seguente mostra come fare. Sostituisci il tuo valore per ciascuna variabile di ambiente, ad esempio. *`$CLUSTER_ID`* In questo esempio è specificata anche l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con

Amazon Aurora e AWS Secrets Manager. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

```
aws rds create-db-cluster --db-cluster-identifier $CLUSTER_ID \
  --engine aurora-mysql --engine-version 8.0.mysql_aurora.3.04.1 \
  --master-username $MASTER_USER_ID --manage-master-user-password \
  --db-cluster-parameter-group-name $CUSTOM_CLUSTER_PARAM_GROUP

aws rds create-db-instance --db-instance-identifier ${INSTANCE_ID}-1 \
  --engine same_value_as_in_create_cluster_command \
  --db-cluster-identifier $CLUSTER_ID --db-instance-class $INSTANCE_CLASS
```

5. Verificare che la funzionalità di query in parallelo sia disponibile per il cluster creato o ripristinato.

   Verifica che l'impostazione di configurazione `aurora_parallel_query` sia presente. Se questa impostazione ha il valore 1, la query parallela è pronta per l'uso. Se questa impostazione ha il valore 0, impostarla su 1 prima di poter utilizzare la query parallela. In entrambi i casi, il cluster è in grado di eseguire query parallele.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Per ripristinare un'istantanea in un cluster di query parallelo con AWS CLI

1. Verifica quali SQL versioni di Aurora My sono compatibili con i cluster di query paralleli. Per fare ciò, utilizzare il comando `describe-db-engine-versions` e controllare il valore del campo `SupportsParallelQuery`. Per un esempio, consulta Verifica della compatibilità di Aurora My SQL version per le query parallele. Decidere quale versione utilizzare per il cluster ripristinato.
2. Individua un'istantanea del cluster SQL compatibile con Aurora My.
3. Seguire la procedura generale AWS CLI riportata in. Ripristino da uno snapshot cluster database

```
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier mynewdbcluster \
  --snapshot-identifier mydbclustersnapshot \
  --engine aurora-mysql
```

4. Verificare che la funzionalità di query in parallelo sia disponibile per il cluster creato o ripristinato. Utilizzare la stessa procedura di verifica descritta in Creazione di un cluster di query parallelo utilizzando CLI.

Attivazione e disattivazione delle interrogazioni parallele in Aurora My SQL

Quando la query parallela è attivata, Aurora My SQL determina se utilizzarla in fase di esecuzione per ogni query. Nel caso di join, unioni, sottoquery e così via, Aurora My determina SQL se utilizzare la query parallela in fase di esecuzione per ogni blocco di query. Per informazioni dettagliate, consulta Verifica delle istruzioni che utilizzano la query parallela per Aurora My SQL e SQLcostrutti per interrogazioni parallele in Aurora My SQL.

Puoi attivare o disattivare dinamicamente le query parallele di un'istanza database a livello globale e a livello di sessione mediante l'opzione aurora_parallel_query. È possibile modificare l'impostazione `aurora_parallel_query` nel gruppo di cluster DB per attivare o disattivare la query parallela per impostazione predefinita.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Per attivare o disattivare il parametro `aurora_parallel_query` a livello di sessione, utilizzare i metodi standard per modificare un'impostazione di configurazione client. Ad esempio, è possibile farlo tramite la riga di `mysql` comando o all'interno di un'applicazione or. JDBC ODBC Il comando sullo standard My SQL client è`set session aurora_parallel_query = {'ON'/'OFF'}`. Puoi anche aggiungere il parametro a livello di sessione alla JDBC configurazione o all'interno del codice dell'applicazione per attivare o disattivare le query parallele in modo dinamico.

È possibile modificare in modo permanente l'impostazione per il parametro `aurora_parallel_query`, per un'istanza DB specifica o per l'intero cluster. Se si specifica il valore del parametro in un gruppo di parametri DB, tale valore si applica solo a un'istanza DB specifica nel cluster. Se si specifica il valore del parametro in un gruppo di parametri cluster DB, tutte le istanze DB del cluster ereditano la stessa impostazione. Per attivare il parametro `aurora_parallel_query`, utilizza le tecniche applicabili ai gruppi di parametri, come descritto in . Effettua la procedura riportata di seguito.

1. Creare un gruppo di parametri cluster personalizzato (scelta consigliata) o un gruppo di parametri DB personalizzato.
2. In questo gruppo di parametri, aggiornare `parallel_query` al valore desiderato.
3. A seconda che sia stato creato un gruppo di parametri cluster DB o un gruppo di parametri DB, collegare il gruppo di parametri al cluster Aurora o alle istanze DB specifiche in cui si prevede di utilizzare la funzionalità di query parallela.

> **Tip**
>
> Poiché `aurora_parallel_query` è un parametro dinamico, non richiede il riavvio del cluster dopo aver modificato questa impostazione. Tuttavia, qualsiasi connessione che utilizzava una query parallela prima di attivare l'opzione continuerà a farlo fino a quando la connessione non viene chiusa o l'istanza viene riavviata.

È possibile modificare il parametro di interrogazione parallela utilizzando l'APIoperazione M odifyDBCluster ParameterGroup o M odifyDBParameter Group o il AWS Management Console.

Puoi attivare l'hash join per i cluster di query paralleli, attivare e disattivare le query parallele utilizzando la RDS console Amazon o il AWS CLI, e sovrascrivere l'ottimizzatore di query parallele.

Indice

- Abilitazione dell'hash join per cluster di query parallele
- Attivazione e disattivazione della query parallela utilizzando la console
- Attivazione e disattivazione delle interrogazioni parallele utilizzando CLI
- Sostituzione dell'ottimizzatore di query parallele

Abilitazione dell'hash join per cluster di query parallele

La query in parallelo viene in genere utilizzata per quei tipi di query che richiedono un uso intensivo di risorse che traggono vantaggio dall'ottimizzazione dell'hash join. Pertanto, è utile assicurarsi che i join hash siano abilitati per i cluster in cui si prevede di utilizzare la query parallela. Per informazioni su come utilizzare i join hash in modo efficace, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

Attivazione e disattivazione della query parallela utilizzando la console

Puoi abilitare o disabilitare le query in parallelo a livello dell'istanza database utilizzando gruppi di parametri.

Per attivare o disattivare la query parallela per un cluster DB con AWS Management Console

1. Creare un gruppo di parametri personalizzato, come descritto in .
2. Imposta aurora_parallel_query su 1 (attivato) o 0 (disattivato). Per impostazione predefinita, il parametro aurora_parallel_query è abilitato sui cluster dove la funzionalità di query in parallelo è disponibile.
3. Se si utilizza un gruppo di parametri cluster personalizzato, collegarlo al cluster Aurora DB in cui si prevede di utilizzare la funzionalità di query parallela. Se usi un gruppo di parametri di database personalizzato, collegalo a una o più istanze database nel cluster. Si consiglia di utilizzare un gruppo di parametri cluster. In questo modo si assicura che tutte le istanze DB nel cluster abbiano le stesse impostazioni per la query parallela e le funzionalità associate, ad esempio hash join.

Attivazione e disattivazione delle interrogazioni parallele utilizzando CLI

Puoi modificare il parametro di query in parallelo utilizzando il comando `modify-db-cluster-parameter-group` o `modify-db-parameter-group`. Scegliere il comando appropriato a seconda che si specifichi il valore di `aurora_parallel_query` tramite un gruppo di parametri cluster DB o un gruppo di parametri DB.

Per attivare o disattivare la query parallela per un cluster DB con CLI

- Modificare il parametro di query in parallelo utilizzando il comando `modify-db-cluster-parameter-group`. Utilizzare un comando come il seguente. Sostituire il nome appropriato per il proprio gruppo di parametri personalizzato. Sostituire `ON` o `OFF` per la parte `ParameterValue` dell'opzione `--parameters`.

```
$ aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "cluster_param_group_name"
```

```
}

aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters ParameterName=aurora_pq,ParameterValue=ON,ApplyMethod=pending-reboot
```

È inoltre possibile attivare o disattivare la query parallela a livello di sessione, ad esempio tramite la `mysql` riga di comando o all'interno di un'ODBCapplicazione JDBC or. A questo proposito, utilizza i metodi standard impiegati per modificare un'impostazione della configurazione client. Ad esempio, il comando sullo standard My SQL client è `set session aurora_parallel_query = {'ON'/'OFF'}` per Aurora My. SQL

Puoi anche aggiungere il parametro a livello di sessione alla JDBC configurazione o all'interno del codice dell'applicazione per attivare o disattivare le query parallele in modo dinamico.

Sostituzione dell'ottimizzatore di query parallele

È possibile utilizzare la variabile di sessione `aurora_pq_force` per sostituire l'ottimizzatore di query parallele e richiedere una query parallela per ogni query. Ti consigliamo di eseguire questa operazione solo a scopo di test. L'esempio seguente mostra come utilizzare `aurora_pq_force` in una sessione.

```
set SESSION aurora_parallel_query = ON;
set SESSION aurora_pq_force = ON;
```

Per disattivare la sostituzione, procedi come segue:

```
set SESSION aurora_pq_force = OFF;
```

Ottimizzazione delle interrogazioni parallele in Aurora My SQL

Per ottimizzare il tuo cluster DB per le query parallele, considera quali cluster DB trarrebbero vantaggio dalla query parallela e se eseguire l'aggiornamento per le query parallele. Quindi, ottimizza il carico di lavoro e crea oggetti dello schema per le query parallele.

Indice

- Pianificazione di un cluster di query parallele
  - Verifica della compatibilità di Aurora My SQL version per le query parallele

- Considerazioni relative agli aggiornamenti per le query in parallelo
  - Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL
  - Aggiornamento ad Aurora My 2.09 e versioni successive SQL
- Ottimizzazione delle prestazioni per le query in parallelo
- Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo

Pianificazione di un cluster di query parallele

La pianificazione di un cluster di database abilitato per la query parallela richiede di effettuare alcune scelte. Questi includono l'esecuzione dei passaggi di configurazione (creazione o ripristino di un cluster Aurora SQL My completo) e la decisione su come attivare le query parallele nel cluster DB.

Considerare quanto segue nell'ambito della pianificazione:

- Se utilizzi Aurora My compatibile con My SQL SQL 5.7, devi creare un cluster con provisioning. Quindi si attiva la query parallela utilizzando il parametro `aurora_parallel_query`.

  Se disponi di un SQL cluster Aurora My esistente, non è necessario creare un nuovo cluster per utilizzare la query parallela. È possibile associare il cluster o istanze DB specifiche nel cluster a un gruppo di parametri con il parametro `aurora_parallel_query` abilitato. In questo modo, è possibile ridurre il tempo e gli sforzi necessari per impostare i dati rilevanti da utilizzare con query parallele.
- Pianificare le tabelle di grandi dimensioni che è necessario riorganizzare in modo da poter utilizzare la query parallela quando si accede ad esse. Potrebbe essere necessario creare nuove versioni di alcune tabelle di grandi dimensioni in cui la query parallela è utile. Ad esempio, potrebbe essere necessario rimuovere gli indici di ricerca full-text. Per informazioni dettagliate, consultare Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo.

Verifica della compatibilità di Aurora My SQL version per le query parallele

Per verificare quali SQL versioni di Aurora My sono compatibili con i cluster di query paralleli, usa il `describe-db-engine-versions` AWS CLI comando e controlla il valore del campo. `SupportsParallelQuery` Il seguente esempio di codice mostra come verificare quali combinazioni sono disponibili per i cluster di query parallele in una AWS regione specificata. Assicurarsi di specificare la stringa di parametro `--query` completa su una singola riga.

```
aws rds describe-db-engine-versions --region us-east-1 --engine aurora-mysql \
```

```
--query '*[]|[?SupportsParallelQuery == `true`].[EngineVersion]' --output text
```

Il comando precedente genera un output simile al seguente: L'output può variare a seconda delle SQL versioni di Aurora My disponibili nella regione specificata AWS .

```
5.7.mysql_aurora.2.11.1
5.7.mysql_aurora.2.11.2
5.7.mysql_aurora.2.11.3
5.7.mysql_aurora.2.11.4
5.7.mysql_aurora.2.11.5
5.7.mysql_aurora.2.11.6
5.7.mysql_aurora.2.12.0
5.7.mysql_aurora.2.12.1
5.7.mysql_aurora.2.12.2
5.7.mysql_aurora.2.12.3
5.7.mysql_aurora.2.12.4
8.0.mysql_aurora.3.04.0
8.0.mysql_aurora.3.04.1
8.0.mysql_aurora.3.04.2
8.0.mysql_aurora.3.04.3
8.0.mysql_aurora.3.05.2
8.0.mysql_aurora.3.06.0
8.0.mysql_aurora.3.06.1
8.0.mysql_aurora.3.07.0
8.0.mysql_aurora.3.07.1
```

Dopo aver iniziato a utilizzare query parallele con un cluster, è possibile monitorare le prestazioni e rimuovere gli ostacoli all'utilizzo delle query parallele. Per quelle istruzioni, Ottimizzazione delle prestazioni per le query in parallelo.

Considerazioni relative agli aggiornamenti per le query in parallelo

A seconda delle versioni originali e di destinazione quando si aggiorna un cluster di query parallelo, è possibile trovare miglioramenti nei tipi di query che la query parallela può ottimizzare. È inoltre possibile che non sia necessario specificare un parametro speciale della modalità motore per la query parallela. Le sezioni seguenti spiegano le considerazioni quando si aggiorna un cluster in cui è attivata la query parallela.

Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL

Diverse SQL istruzioni, clausole e tipi di dati dispongono di un supporto per le query parallele nuovo o migliorato a partire da Aurora My versione 3. SQL Quando si esegue l'aggiornamento da una

versione precedente alla versione 3, verificare se ulteriori query possono trarre vantaggio dalle ottimizzazioni parallele delle query. Per informazioni su questi miglioramenti delle query parallele, vedere Tipi di dati di colonna,Tabelle partizionate, eFunzioni aggregate, clausole BY e clausole GROUP HAVING.

Se stai aggiornando un cluster di query parallele da Aurora My SQL 2.08 o versione precedente, scopri anche le modifiche relative all'attivazione delle query parallele. A questo scopo, leggere Aggiornamento ad Aurora My 2.09 e versioni successive SQL.

In Aurora My SQL versione 3, l'ottimizzazione degli hash join è attivata per impostazione predefinita. L'opzione di configurazione `aurora_disable_hash_join` delle versioni precedenti non viene utilizzata.

Aggiornamento ad Aurora My 2.09 e versioni successive SQL

In Aurora My SQL versione 2.09 e successive, la query parallela funziona per i cluster predisposti e non richiede il parametro engine mode. `parallelquery` Pertanto, non è necessario creare un nuovo cluster o ripristinare da uno snapshot esistente per utilizzare query parallele con queste versioni. È possibile utilizzare le procedure di aggiornamento descritte in Aggiornamento della versione secondaria o del livello di patch di un cluster Aurora My DB SQL per aggiornare il cluster a tale versione. È possibile aggiornare un cluster meno recente indipendentemente dal fatto che si trattasse di un cluster di query parallelo o di un cluster di cui è stato eseguito il provisioning. Per ridurre il numero di scelte nel menu Versione motore è possibile scegliere Mostra versioni che supportano la funzionalità di query parallela per filtrare le voci di tale menu. Quindi scegli Aurora My SQL 2.09 o versione successiva.

Dopo aver aggiornato un cluster di query parallele precedente ad Aurora My SQL 2.09 o versione successiva, si attiva la query parallela nel cluster aggiornato. La query parallela è disattivata per impostazione predefinita in queste versioni e la procedura per abilitarla è diversa. Anche l'ottimizzazione hash join è disattivata per impostazione predefinita e deve essere abilitata separatamente. Pertanto, assicurarsi di attivare nuovamente queste impostazioni dopo l'aggiornamento. Per istruzioni su questa operazione, vedere Attivazione e disattivazione delle interrogazioni parallele in Aurora My SQL e Abilitazione dell'hash join per cluster di query parallele.

In particolare, è possibile abilitare la query parallela utilizzando i parametri di configurazione `aurora_parallel_query=ON` e `aurora_disable_hash_join=OFF` invece di `aurora_pq_supported` e `aurora_pq`. I `aurora_pq` parametri `aurora_pq_supported` and sono obsoleti nelle versioni più recenti di Aurora My. SQL

Nel cluster aggiornato, l'attributo `EngineMode` ha il valore `provisioned` invece di `parallelquery`. Per verificare se la query parallela è disponibile per una versione del motore specificata, ora controllate il valore del `SupportsParallelQuery` campo nell'output del `describe-db-engine-versions` AWS CLI comando. Nelle SQL versioni precedenti di Aurora My, si verificava la presenza di `parallelquery` nell'`SupportedEngineModes`elenco.

Dopo l'aggiornamento ad Aurora My SQL versione 2.09 o successiva, puoi sfruttare le seguenti funzionalità. Queste funzionalità non sono disponibili per i cluster di query parallele che eseguono versioni precedenti di Aurora MySQL.

- Approfondimenti sulle prestazioni. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- Backtrack in corso Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.
- Arresto e avvio del cluster. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.

Ottimizzazione delle prestazioni per le query in parallelo

Per gestire le prestazioni di un carico di lavoro con la funzionalità di query in parallelo, assicurati che questa sia utilizzata per le query per le quali tale ottimizzazione risulta più utile.

A questo proposito, puoi eseguire le operazioni elencate di seguito:

- Assicurarsi che le tabelle più grandi siano compatibili con la query parallela. È possibile modificare le proprietà della tabella o ricreare alcune tabelle in modo che le query per tali tabelle possano trarre vantaggio dall'ottimizzazione delle query parallele. Per scoprire come, consulta Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo.
- Monitorare le query che utilizzano la funzionalità di query in parallelo. Per scoprire come, consulta Monitoraggio delle interrogazioni parallele per Aurora My SQL.
- Verificare che la query parallela venga utilizzata per le query a esecuzione prolungata e a uso intensivo di dati e con il giusto livello di concorrenza per il carico di lavoro. Per scoprire come, consulta Verifica delle istruzioni che utilizzano la query parallela per Aurora My SQL.
- Ottimizza il SQL codice per attivare la query parallela da applicare alle query che ti aspetti. Per scoprire come, consulta SQLcostrutti per interrogazioni parallele in Aurora My SQL.

Creazione di oggetti di schema per usufruire dei vantaggi delle query in parallelo

Prima di creare o modificare tabelle che si intende utilizzare per query parallele, assicurarsi di familiarizzare con i requisiti descritti in Prerequisiti e Limitazioni.

Poiché per le query in parallelo è necessario che le tabelle utilizzino l'impostazione `ROW_FORMAT=Compact` o `ROW_FORMAT=Dynamic`, verifica le impostazioni di configurazione Aurora per eventuali modifiche all'opzione di configurazione `INNODB_FILE_FORMAT`. Esegui l'istruzione `SHOW TABLE STATUS` per confermare il formato di riga per tutte le tabelle in un database.

Prima di modificare lo schema per consentire alla query parallela di lavorare con più tabelle, assicurarsi di eseguire il test. I test devono confermare se la query parallela determina un aumento netto delle prestazioni per tali tabelle. Assicurati inoltre che i requisiti di schema per le query in parallelo coincidano con i tuoi obiettivi.

Ad esempio, prima di passare da `ROW_FORMAT=Compressed` a `ROW_FORMAT=Compact` o `ROW_FORMAT=Dynamic`, compara le prestazioni dei carichi di lavoro rispetto alle tabelle di origine e a quelle nuove. Considera inoltre altri potenziali effetti come un aumento del volume di dati.

Verifica delle istruzioni che utilizzano la query parallela per Aurora My SQL

In regola generale, non è necessario eseguire specifiche operazioni per utilizzare le query in parallelo. Quando una query soddisfa i requisiti essenziali della funzionalità di query in parallelo, l'ottimizzatore di query decide automaticamente se utilizzare tale funzionalità per ogni query.

Se esegui dei test in un ambiente di sviluppo o di test, è possibile che la funzione di query in parallelo non sia utilizzata in quanto le tabelle non contengono una quantità sufficiente di righe o volumi di dati. I dati associati alle tabelle, soprattutto quelle create recentemente per effettuare dei test, possono anche trovarsi interamente nel pool di buffer.

Durante il monitoraggio o l'ottimizzazione delle prestazioni del cluster, devi determinare se le query in parallelo sono utilizzate nei contesti appropriati. È possibile modificare lo schema del database, le impostazioni, le SQL query o persino la topologia del cluster e le impostazioni di connessione all'applicazione per sfruttare questa funzionalità.

Per verificare se una query utilizza una query parallela, controlla il piano di query (noto anche come «plain plan») eseguendo l'`EXPLAIN`istruzione. Per esempi di come SQL le istruzioni, le clausole e le espressioni influiscono sull'`EXPLAIN`output di una query parallela, vedere. SQLcostrutti per interrogazioni parallele in Aurora My SQL

L'esempio seguente illustra la differenza tra un piano query tradizionale e un piano di query in parallelo. Questo piano di spiegazione è tratto dalla Query 3 del benchmark TPC -H. Molte delle query di esempio contenute in questa sezione utilizzano le tabelle del set di dati -H. TPC È possibile ottenere le definizioni delle tabelle, le interrogazioni e il dbgen programma che genera dati di esempio dal sito Web -h. TPC

```
EXPLAIN SELECT l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) AS revenue,
  o_orderdate,
  o_shippriority
FROM customer,
  orders,
  lineitem
WHERE c_mktsegment = 'AUTOMOBILE'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < date '1995-03-13'
AND l_shipdate > date '1995-03-13'
GROUP BY l_orderkey,
  o_orderdate,
  o_shippriority
ORDER BY revenue DESC,
  o_orderdate LIMIT 10;
```

Per impostazione predefinita, la query potrebbe avere un piano simile al seguente. Se il join hash non viene visualizzato nel piano di query, assicurarsi che l'ottimizzazione sia abilitata per prima.

```
+----+-------------+----------+------------+------+--------------+------+---------
+------+----------+----------+----------------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra                                              |
+----+-------------+----------+------------+------+--------------+------+---------
+------+----------+----------+----------------------------------------------------+
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |  1480234 |    10.00 | Using where; Using temporary; Using filesort       |
|  1 | SIMPLE      | orders   | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 14875240 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
|  1 | SIMPLE      | lineitem | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 59270573 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+----------+------------+------+--------------+------+---------
+------+----------+----------+----------------------------------------------------+
```

Per Aurora My SQL versione 3, si attiva l'hash join a livello di sessione emettendo la seguente dichiarazione.

```
SET optimizer_switch='block_nested_loop=on';
```

Per Aurora My SQL versione 2.09 e successive, impostate il parametro DB o il parametro `aurora_disable_hash_join` DB cluster su `0` (off). La disattivazione di `aurora_disable_hash_join` imposta il valore di `optimizer_switch` su `hash_join=on`.

Dopo aver attivato il join hash, prova a eseguire nuovamente l'istruzione `EXPLAIN`. Per informazioni su come utilizzare i join hash in modo efficace, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

Quando la funzionalità di query in parallelo di join hash è disabilitata, la query può avere un piano simile al seguente, che utilizza un hash join ma non le query in parallelo.

```
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
| id | select_type | table    |...| rows      | Extra
                          |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
|  1 | SIMPLE      | customer |...|   5798330 | Using where; Using index; Using
 temporary; Using filesort       |
|  1 | SIMPLE      | orders   |...| 154545408 | Using where; Using join buffer (Hash
 Join Outer table orders)   |
|  1 | SIMPLE      | lineitem |...| 606119300 | Using where; Using join buffer (Hash
 Join Outer table lineitem) |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
```

Quando la funzionalità di query in parallelo è abilitata, due fasi di questo piano possono utilizzare l'ottimizzazione mediante query in parallelo, come indicato dalla colonna `Extra` nell'output `EXPLAIN`. L'elaborazione a uso intensivo e intensivo CPU di I/O per questi passaggi viene trasferita al livello di archiviazione.

```
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                                    |
```

```
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using index; Using temporary; Using filesort
                                                      |
|  1 |...| Using where; Using join buffer (Hash Join Outer table orders); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra)   |
|  1 |...| Using where; Using join buffer (Hash Join Outer table lineitem); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Per informazioni su come interpretare l'EXPLAINoutput di una query parallela e sulle parti di SQL istruzioni a cui è possibile applicare la query parallela, vedereSQLcostrutti per interrogazioni parallele in Aurora My SQL.

L'esempio di output seguente mostra i risultati dell'esecuzione della query precedente su un'istanza db.r4.2xlarge con un pool di buffer a freddo. L'esecuzione della query risulta molto più rapida con la funzionalità di query in parallelo.

> **Note**
>
> Poiché la durata dipende da molti fattori ambientali, i risultati potrebbero essere differenti. Esegui sempre dei test di prestazioni personali per confermare i risultati con il tuo ambiente, carico di lavoro e così via.

```
-- Without parallel query
+------------+-------------+-------------+----------------+
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (24 min 49.99 sec)
```

```
-- With parallel query
+------------+-------------+-------------+----------------+
```

```
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (1 min 49.91 sec)
```

Molte delle query di esempio contenute in questa sezione utilizzano le tabelle di questo set di dati TPC -H, in particolare la PART tabella, che ha 20 milioni di righe e la seguente definizione.

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| p_partkey     | int(11)       | NO   | PRI | NULL    |       |
| p_name        | varchar(55)   | NO   |     | NULL    |       |
| p_mfgr        | char(25)      | NO   |     | NULL    |       |
| p_brand       | char(10)      | NO   |     | NULL    |       |
| p_type        | varchar(25)   | NO   |     | NULL    |       |
| p_size        | int(11)       | NO   |     | NULL    |       |
| p_container   | char(10)      | NO   |     | NULL    |       |
| p_retailprice | decimal(15,2) | NO   |     | NULL    |       |
| p_comment     | varchar(23)   | NO   |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Sperimenta con il tuo carico di lavoro per capire se SQL le singole istruzioni possono trarre vantaggio dalla query parallela. Utilizza quindi le tecniche di monitoraggio seguenti per determinare la frequenza d'uso delle query in parallelo con i carichi di lavoro reali nel tempo. Per tali carichi di lavoro, vengono applicati dei fattori supplementari come i limiti di simultaneità.

Monitoraggio delle interrogazioni parallele per Aurora My SQL

Se il tuo SQL cluster Aurora My utilizza una query parallela, potresti notare un aumento `VolumeReadIOPS` dei valori. Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare un aumento delle operazioni di lettura e degli addebiti associati.

Oltre ai CloudWatch parametri Amazon descritti inVisualizzazione delle metriche nella console Amazon RDS, Aurora fornisce altre variabili di stato globali. È possibile utilizzare queste variabili di stato globali per monitorare l'esecuzione di query parallele. Possono darti informazioni sul motivo per

cui l'ottimizzatore potrebbe utilizzare o meno la query parallela in una determinata situazione. Per accedere a tali variabili, puoi utilizzare il comando `SHOW GLOBAL STATUS`. Queste variabili sono descritte nella tabella seguente.

Una sessione di interrogazione parallela non è necessariamente una one-to-one mappatura con le query eseguite dal database. Supponiamo ad esempio che tuo il piano di query comporti due fasi che utilizzano la funzionalità di query in parallelo. In tal caso, la query comporta due sessioni in parallelo e i contatori dei tentativi di richieste e delle richieste riuscite vengono incrementati di due.

Quando esegui dei test con la funzionalità di query in parallelo mediante l'istruzione `EXPLAIN`, prevedi un aumento dei contatori designati come "non scelti" anche se le query non sono in esecuzione. Quando utilizzi la funzionalità di query in parallelo in un ambiente di produzione, puoi verificare se i contatori "non scelti" aumentano più rapidamente del previsto. A questo punto, è possibile regolare in modo che la query parallela venga eseguita per le query previste. A tale scopo, è possibile modificare le impostazioni del cluster, il mix di query, le istanze database in cui è abilitata la query parallela e così via.

Questi contatori sono monitorati a livello di istanza database. Quando ti connetti a un endpoint differente, è possibile che i parametri siano differenti in quanto ogni istanza database esegue il relativo set di query in parallelo. I parametri possono variare anche quando l'endpoint di lettura si connette a un'istanza database differente per ogni sessione.

Nome	Descrizione
`Aurora_pq_bytes_returned`	Il numero di byte per le strutture di dati tuple trasmesse al nodo head nel corso delle query in parallelo. Dividi questo valore per 16.384 per compararlo a `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	Il numero massimo di sessioni di query in parallelo eseguibili simultaneamente su questa istanza database Aurora. Si tratta di un numero fisso che dipende dalla classe dell'istanza AWS DB.
`Aurora_pq_pages_pushed_down`	Il numero di pagine di dati (ognuna con una dimensione fissa di 16 KiB) per le quali una

Nome	Descrizione
	query in parallelo ha evitato una trasmissione di rete al nodo head.
`Aurora_pq_request_attempted`	Il numero di sessioni di query in parallelo necessarie. Questo valore può rappresentare più di una sessione per query, a seconda di SQL costrutti come sottoquery e join.
`Aurora_pq_request_executed`	Il numero di sessioni di query in parallelo riuscite.
`Aurora_pq_request_failed`	Il numero di sessioni di query in parallelo che hanno restituito un errore al client. In alcuni casi, una richiesta di query in parallelo può non riuscire, ad esempio a causa di un problema nel livello di storage. In questi casi, viene eseguito un nuovo tentativo per la parte di query non riuscita utilizzando il meccanismo di query non in parallelo. Se anche il nuovo tentativo non riesce, viene restituito un errore al client e questo contatore viene incrementato.
`Aurora_pq_request_in_progress`	Il numero di sessioni di query in parallelo attualmente in corso. Questo numero si applica all'istanza database Aurora a cui sei connesso e non all'intero cluster di database Aurora. Per determinare se un'istanza database è prossima al relativo limite di simultaneità, compara questo valore a `Aurora_pq_max_concurrent_requests` .

Nome	Descrizione
`Aurora_pq_request_not_chosen`	Il numero di volte che una query in parallelo non è stata scelta per una query. Questo valore è la somma di vari altri contatori più granulari . Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_below_min_rows`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero di righe nella tabella. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_column_bit`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele a causa di un tipo di dati non supportato nell'elenco delle colonne proiettate.
`Aurora_pq_request_not_chosen_column_geometry`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con il tipo di `GEOMETRY` dati. Per informazioni sulle SQL versioni di Aurora My che rimuovono questa limitazione, consulta. Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL

Nome	Descrizione
`Aurora_pq_request_not_chosen_column_lob`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con un tipo di dati `LOB` o colonne `VARCHAR` memorizzate esternamente a causa della lunghezza dichiarata. Per informazioni sulle SQL versioni di Aurora My che rimuovono questa limitazione, consulta. Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL
`Aurora_pq_request_not_chosen_column_virtual`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene una colonna virtuale.
`Aurora_pq_request_not_chosen_custom_charset`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di colonne con un set di caratteri personalizzato.
`Aurora_pq_request_not_chosen_fast_ddl`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella viene attualmente modificata da un'istruzione fast `DDLALTER`.
`Aurora_pq_request_not_chosen_ few_pages_outside_buffer_pool`	Il numero di volte che una query in parallelo non è stata scelta anche se meno del 95% dei dati della tabella era già nel pool di buffer, in quanto non c'erano dati non memorizzati nel buffer sufficienti per giustificare l'utilizzo della funzione di query in parallelo.

Nome	Descrizione
`Aurora_pq_request_not_chosen_full_text_index`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di indici full-text.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	Il numero di volte che una query in parallelo non è stata scelta in quanto un'elevata percentuale di dati di tabella (attualmente maggiore del 95%) era già nel pool di buffer. In questi casi, l'ottimizzatore determina che la lettura dei dati del pool di buffer è più efficace. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_index_hint`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query include un hint di indice.
`Aurora_pq_request_not_chosen_innodb_table_format`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella utilizza un formato di riga InnoDB non supportato. La query parallela Aurora si applica solo ai formati di riga `COMPACT`, `REDUNDANT` e `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	Il numero di richieste di query in parallelo che hanno utilizzato il percorso di elaborazione di query non in parallelo in seguito all'avvio della query in una transazione di lunga durata. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.

Nome	Descrizione
`Aurora_pq_request_not_chosen_no_where_clause`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query non include alcuna clausola `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza una scansione di intervallo su un indice.
`Aurora_pq_request_not_chosen_row_length_too_long`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la lunghezza totale combinata di tutte le colonne è troppo lunga.
`Aurora_pq_request_not_chosen_small_table`	Il numero di volte che una query in parallelo non è stata scelta a causa della dimensione globale della tabella, come determinata dal numero di righe e dalla lunghezza media delle stesse. Un'istruzione `EXPLAIN` può increment are questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_temporary_table`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa riferimen to a tabelle temporanee che utilizzano i tipi di tabella `MyISAM` o `memory` non supportati.

Nome	Descrizione
`Aurora_pq_request_not_chosen_tx_isolation`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza un livello di isolamento delle transazioni non supportato. Sulle istanze del database di lettura, la query parallela si applica solo ai livelli di isolamento `REPEATABLE READ` e `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa parte di un'istruzione `UPDATE` o `DELETE`.
`Aurora_pq_request_not_chosen_unsupported_access`	Il numero di richieste di query in parallelo che utilizzano il percorso di elaborazione di query non in parallelo in quanto la clausola `WHERE` non soddisfa i criteri delle query in parallelo . Ciò può avvenire se la query non richiede l'analisi di un grande volume di dati oppure se la query è un'istruzione `DELETE` o `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallelo perché il cluster Aurora SQL My DB non utilizza una configurazione di archiviazione del cluster Aurora supportata. Questo parametro è disponibile in Aurora My SQL versione 3.04 e successive. Per ulteriori informazioni, consulta Limitazioni.

Nome	Descrizione
`Aurora_pq_request_throttled`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero massimo di query in parallelo simultanee già in esecuzione su una particolare istanza database Aurora.

SQLcostrutti per interrogazioni parallele in Aurora My SQL

Nella sezione seguente, puoi trovare maggiori dettagli sul motivo per cui determinate SQL istruzioni utilizzano o non utilizzano la query parallela. Questa sezione descrive anche come le SQL funzionalità di Aurora My interagiscono con le query parallele. Queste informazioni dettagliate possono consentirti di diagnosticare i problemi relativi alle prestazioni di un cluster che utilizza le query in parallelo o di comprendere il modo in cui queste vengono applicate a un particolare carico di lavoro.

La decisione di utilizzare la funzione di query in parallelo dipende da molti fattori che intervengono al momento dell'esecuzione dell'istruzione. Di conseguenza, per certe query, tale funzionalità può essere utilizzata sempre, mai o solo in determinate circostanze.

> **Tip**
>
> Quando visualizzi questi esempi inHTML, puoi usare il widget Copia nell'angolo superiore destro di ogni elenco di codici per copiare il SQL codice e provare tu stesso. L'utilizzo del widget Copia evita di copiare i caratteri aggiuntivi attorno alle righe del prompt `mysql>` e della continuazione `->`.

Argomenti

- EXPLAINdichiarazione
- WHEREClausola
- linguaggio di definizione dei dati (DDL)
- Tipi di dati di colonna
- Tabelle partizionate
- Funzioni aggregate, clausole BY e clausole GROUP HAVING

- Chiamate di funzione nella clausola WHERE
- LIMITClausola
- Operatori di confronto
- Join
- Sottoquery
- UNION
- Visualizzazioni
- Istruzioni del linguaggio di manipolazione dei dati () DML
- Transazioni e blocco
- Indici B-Tree
- Indici di ricerca a testo completo (FTS)
- Colonne virtuali
- Meccanismi di caching integrati
- Suggerimenti per l'ottimizzazione
- Le mie tabelle temporanee ISAM

EXPLAINdichiarazione

Come mostrato negli esempi in questa sezione, l'istruzione `EXPLAIN` indica se ogni fase di una query è idonea alla funzionalità di query in parallelo. Indica inoltre quali aspetti di una query possono essere trasferiti a un livello di storage. Di seguito sono riportati gli elementi più importanti del piano query:

- Un valore differente da `NULL` per la colonna `key` suggerisce che la query può essere eseguita in modo efficace mediante ricerche di indice e che l'utilizzo della funzionalità di query in parallelo è improbabile.
- Un valore basso della colonna `rows` (ovvero un valore non in milioni) suggerisce che la query non ha accesso a dati sufficienti per giustificare l'utilizzo della funzionalità di query in parallelo e che quindi è improbabile. Ciò significa che la query parallela è improbabile.
- La colonna `Extra` indica se l'uso della funzionalità di query in parallelo è previsto. Questo output è simile all'esempio seguente.

  ```
  Using parallel query (A columns, B filters, C exprs; D extra)
  ```

  Il numero `columns` rappresenta il numero di colonne a cui il blocco di query fa riferimento.

Il numero `filters` rappresenta il numero di predicati `WHERE` che rappresentano un semplice confronto tra un valore di colonna e una costante. Il confronto può ricercare un'uguaglianza, una disuguaglianza o un intervallo. Aurora può parallelizzare questi tipi di predicati in maniera più efficace.

Il numero `exprs` rappresenta il numero di espressioni come chiamate di funzione, operatori o altre espressioni che possono anche essere parallelizzate, sebbene meno efficacemente rispetto a una condizione di filtro.

Il numero `extra` rappresenta il numero di espressioni che non è possibile trasferire e che sono eseguite dal nodo head.

Consideriamo ad esempio il seguente output `EXPLAIN`.

```
mysql> explain select p_name, p_mfgr from part
    -> where p_brand is not null
    -> and upper(p_type) is not null
    -> and round(p_retailprice) is not null;
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table |...| rows     | Extra
                                 |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part  |...| 20427936 | Using where; Using parallel query (5
 columns, 1 filters, 2 exprs; 0 extra) |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
```

Le informazioni nella colonna `Extra` mostrano che cinque colonne sono estratte da ogni riga per valutare le condizioni della query e costruire il set di risultati. Un predicato `WHERE` comporta un filtro, ovvero una colonna che è testata direttamente nella clausola `WHERE`. Due clausole `WHERE` richiedono la valutazione di espressioni più complesse, che in questo caso implicano chiamate di funzione. Il campo `0 extra` conferma che tutte le operazioni nella clausola `WHERE` sono trasferite al livello di storage nell'ambito dell'elaborazione di query in parallelo.

Nei casi in cui una query in parallelo non viene scelta, puoi in genere dedurre il motivo dalle altre colonne dell'output `EXPLAIN`. Ad esempio, il valore `rows` può essere troppo basso oppure la colonna `possible_keys` può indicare che la query è in grado di utilizzare una ricerca di indice anziché

l'analisi di una grande quantità di dati. Nell'esempio seguente viene illustrata una query in cui l'ottimizzatore può stimare che la query eseguirà la scansione solo di un numero ridotto di righe. Lo fa in base alle caratteristiche della chiave primaria. In tal caso, nessuna query in parallelo è necessaria.

```
mysql> explain select count(*) from part where p_partkey between 1 and 100;
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows |
 Extra                    |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
|  1 | SIMPLE      | part  | range | PRIMARY       | PRIMARY | 4       | NULL |   99 |
 Using where; Using index |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
```

L'output che indica se una query in parallelo sarà utilizzata prende in considerazione tutti i fattori presenti al momento dell'esecuzione dell'istruzione `EXPLAIN`. L'ottimizzatore può effettuare una scelta differente quando la query è in esecuzione, se la situazione è cambiata nel frattempo. Ad esempio, `EXPLAIN` può indicare che un'istruzione utilizzerà una query in parallelo. Tuttavia, quando successivamente la query viene eseguita, può non usare la query in parallelo in base alle condizioni in quel momento. Tali condizioni possono includere diverse altre query parallele in esecuzione contemporaneamente. Possono anche includere righe che vengono eliminate dalla tabella, un nuovo indice in fase di creazione, troppo tempo che passa all'interno di una transazione aperta e così via.

WHEREClausola

Per utilizzare l'ottimizzazione mediante query in parallelo, una query deve includere una clausola `WHERE`.

Questa ottimizzazione accelera molti tipi di espressioni utilizzati nella clausola `WHERE`:

- Confronti semplici tra un valore di colonna e una costante, noti come filtri. Questi confronti sono ottimali quando trasferiti al livello di storage. Il numero di espressioni di filtro in una query è indicato nell'output `EXPLAIN`.
- Anche altri tipi di espressioni nella clausola `WHERE` sono trasferiti al livello di storage quando possibile. Il numero di tali espressioni in una query è indicato nell'output `EXPLAIN`. Queste espressioni possono essere chiamate di funzione, operatori `LIKE`, espressioni `CASE` e così via.

- Alcune funzioni e operatori non sono attualmente trasferiti dalla query in parallelo. Il numero di tali espressioni in una query è indicato come contatore `extra` nell'output `EXPLAIN`. Il resto della query può ancora utilizzare la funzionalità di query in parallelo.
- Sebbene le espressioni nell'elenco di selezione non siano trasferite, le query contenenti tali funzioni possono ancora beneficiare di una riduzione del traffico di rete per i risultati intermedi delle query in parallelo. Ad esempio, le query che chiamano funzioni di aggregazione nell'elenco di selezione possono beneficiare delle query in parallelo, anche se le funzioni di aggregazione non sono trasferite.

Ad esempio, la query seguente esegue un'analisi dell'intera tabella ed elabora tutti i valori per la colonna `P_BRAND`: Tuttavia, non utilizza la funzionalità di query in parallelo in quanto la query non include alcuna clausola `WHERE`.

```
mysql> explain select count(*), p_brand from part group by p_brand;
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows     |
 Extra                            |
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
|  1 | SIMPLE      | part  | ALL  | NULL          | NULL | NULL    | NULL | 20427936 |
 Using temporary; Using filesort |
+----+-------------+-------+------+---------------+------+---------+------+----------
+----------------------------------+
```

Al contrario, la query seguente include predicati `WHERE` che filtrano i risultati, di conseguenza la funzionalità di query in parallelo può essere utilizzata:

```
mysql> explain select count(*), p_brand from part where p_name is not null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
    -> group by p_brand;
+----+...+----------
+-------------------------------------------------------------------------------------------
+
| id |...| rows     | Extra
                                          |
+----+...+----------
+-------------------------------------------------------------------------------------------
+
```

```
|  1 |...| 20427936 | Using where; Using temporary; Using filesort; Using parallel
 query (5 columns, 1 filters, 2 exprs; 0 extra) |
+----+...+----------
+-----------------------------------------------------------------------------------------
+
```

Se l'ottimizzatore stima che il numero di righe restituite per un blocco di query è basso, la funzionalità di query in parallelo non viene utilizzata per quel blocco. L'esempio seguente mostra uno scenario in cui un operatore "maggiore di" nella colonna di chiave primaria viene applicato a milioni di righe, comportando l'utilizzo della funzionalità di query in parallelo. Per l'operatore inverso "minore di" viene stimata l'applicazione soltanto ad alcune righe e tale funzionalità non è quindi utilizzata.

```
mysql> explain select count(*) from part where p_partkey > 10;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+

mysql> explain select count(*) from part where p_partkey < 10;
+----+...+------+--------------------------+
| id |...| rows | Extra                    |
+----+...+------+--------------------------+
|  1 |...|    9 | Using where; Using index |
+----+...+------+--------------------------+
```

linguaggio di definizione dei dati (DDL)

In Aurora My SQL versione 2, la query parallela è disponibile solo per le tabelle per le quali non sono in sospeso operazioni fast data definition language (DDL). In Aurora My SQL versione 3, è possibile utilizzare la query parallela su una tabella contemporaneamente a un'operazione istantaneaDDL.

Instant DDL in Aurora My SQL versione 3 sostituisce la funzionalità veloce DDL di Aurora My versione 2. SQL Per informazioni su instantDDL, vedi. DDL istantaneo (Aurora MySQL versione 3)

Tipi di dati di colonna

In Aurora My SQL versione 3, la query parallela può funzionare con tabelle contenenti colonne con tipi di datiTEXT, BLOBJSON, e. GEOMETRY Può funzionare anche con colonne VARCHAR e CHAR con una lunghezza massima dichiarata superiore a 768 byte. Se la query fa riferimento a colonne contenenti tipi di oggetti così grandi, il lavoro aggiuntivo per recuperarli aggiunge un sovraccarico all'elaborazione delle query. In tal caso, verificare se la query può omettere i riferimenti a tali colonne. In caso contrario, eseguire i valori di riferimento per confermare se tali query sono più veloci con la query parallela attivata o disattivata.

In Aurora My SQL versione 2, la query parallela presenta le seguenti limitazioni per i tipi di oggetti di grandi dimensioni:

- I tipi di dati TEXT, BLOB, JSON e GEOMETRY non sono supportati con le query in parallelo. Una query che fa riferimento alle colonne di questi tipi non può utilizzare le query in parallelo.
- Le colonne a lunghezza variabile (tipi di dati VARCHAR e CHAR) sono compatibili con le query in parallelo fino a una lunghezza dichiarata massima di 768 byte. Una query che fa riferimento a qualsiasi colonna dei tipi dichiarati con una lunghezza massima superiore non può utilizzare le query in parallelo. Per le colonne che utilizzano set di caratteri multibyte, il limite di byte prende in considerazione il numero massimo di byte nel set di caratteri. Ad esempio, per il set di caratteri utf8mb4 (che ha un limite di lunghezza di 4 byte), una colonna VARCHAR(192) è compatibile con le query in parallelo mentre una colonna VARCHAR(193) non lo è.

Tabelle partizionate

È possibile utilizzare tabelle partizionate con query parallele in Aurora My versione 3. SQL Poiché le tabelle partizionate sono rappresentate internamente come più tabelle più piccole, una query che utilizza query parallele su una tabella non partizionata potrebbe non utilizzare query parallele su una tabella partizionata identica. Aurora My SQL valuta se ogni partizione è sufficientemente grande da qualificarsi per l'ottimizzazione delle query parallele, invece di valutare la dimensione dell'intera tabella. Controllare se la variabile di stato Aurora_pq_request_not_chosen_small_table viene incrementata se una query in una tabella partizionata non utilizza una query parallela quando ci si aspetta.

Ad esempio, si consideri una tabella partizionata con PARTITION BY HASH (*column*) PARTITIONS 2 e un'altra tabella partizionata con PARTITION BY HASH (*column*) PARTITIONS 10. Nella tabella con due partizioni, le partizioni sono cinque volte più grandi della tabella con dieci partizioni. Pertanto, è più probabile che la query parallela venga utilizzata per le query sulla tabella

con meno partizioni. Nell'esempio seguente, la tabella PART_BIG_PARTITIONS ha due partizioni e la PART_SMALL_PARTITIONS ha dieci partizioni. Con dati identici, è più probabile che la query parallela venga utilizzata per la tabella con meno partizioni di grandi dimensioni.

```
mysql> explain select count(*), p_brand from part_big_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------
+
| id | select_type | table               | partitions | Extra
                                                                                      |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------
+
|  1 | SIMPLE      | part_big_partitions | p0,p1      | Using where; Using temporary;
 Using parallel query (4 columns, 1 filters, 1 exprs; 0 extra; 1 group-bys, 1 aggrs) |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------
+

mysql> explain select count(*), p_brand from part_small_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+-----------------------+------------------------------
+------------------------------+
| id | select_type | table                 | partitions                   | Extra
                   |
+----+-------------+-----------------------+------------------------------
+------------------------------+
|  1 | SIMPLE      | part_small_partitions | p0,p1,p2,p3,p4,p5,p6,p7,p8,p9 | Using
 where; Using temporary |
+----+-------------+-----------------------+------------------------------
+------------------------------+
```

Funzioni aggregate, clausole BY e clausole GROUP HAVING

Le query che implicano funzioni di aggregazione sono spesso adatte alle query in parallelo, in quanto comportano l'analisi di un gran numero di righe in tabelle di grandi dimensioni.

In Aurora My SQL 3, la query parallela può ottimizzare le chiamate di funzioni aggregate nell'elenco di selezione e nella clausola. `HAVING`

Prima di Aurora My SQL 3, le chiamate alle funzioni aggregate nell'elenco di selezione o nella `HAVING` clausola non venivano trasferite al livello di archiviazione. Tuttavia, la funzionalità di query in parallelo può sempre migliorare le prestazioni di queste query con funzioni di aggregazione. A questo proposito, estrae dapprima in parallelo i valori di colonna dalle pagine di dati non elaborati nel livello di storage. Ritrasmette quindi quei valori al nodo head in un formato di tupla compatto anziché come pagine di dati complete. Come sempre, la query richiede almeno un predicato `WHERE` affinché la funzione di query in parallelo sia attivata.

Gli esempi semplici seguenti illustrano i tipi di query di aggregazione che possono beneficiare della funzionalità di query in parallelo restituendo risultati intermedi in formato compatto al nodo head, filtrando le righe non corrispondenti dai risultati intermedi oppure con entrambe le operazioni.

```
mysql> explain select sql_no_cache count(distinct p_brand) from part where p_mfgr =
 'Manufacturer#5';
+----+...+-------------------------------------------------------------------------------+
| id |...| Extra                                                                         |
+----+...+-------------------------------------------------------------------------------+
|  1 |...| Using where; Using parallel query (2 columns, 1 filters, 0 exprs; 0 extra) |
+----+...+-------------------------------------------------------------------------------+

mysql> explain select sql_no_cache p_mfgr from part where p_retailprice > 1000 group by
 p_mfgr having count(*) > 100;
+----+...
+-------------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                  |
+----+...
+-------------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (3
 columns, 0 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------------
+
```

Chiamate di funzione nella clausola WHERE

Aurora consente di applicare l'ottimizzazione mediante query in parallelo per le chiamate alla maggior parte delle funzioni integrate nella clausola `WHERE`. La parallelizzazione di queste chiamate di funzione scarica parte del CPU lavoro dal nodo principale. La valutazione delle funzioni di predicato in parallelo durante le prime fasi di una query consente a Aurora di ridurre al minimo la quantità di dati trasmessi ed elaborati nelle fasi successive.

Attualmente, la parallelizzazione non viene applicata alle chiamate di funzione nell'elenco di selezione. Queste funzioni sono valutate dal nodo head anche se chiamate di funzione identiche sono presenti nella clausola `WHERE`. I valori di origine delle colonne pertinenti sono inclusi nelle tuple ritrasmesse dai nodi di storage al nodo head. Il nodo head esegue tutte le trasformazioni come `UPPER`, `CONCATENATE`, ecc., allo scopo di generare i valori finali del set di risultati.

Nell'esempio seguente, una query in parallelo parallelizza la chiamata a `LOWER` in quanto è presente nella clausola `WHERE`. La query in parallelo non viene applicata alle chiamate a `SUBSTR` e `UPPER` in quanto queste sono presenti nell'elenco di selezione.

```
mysql> explain select sql_no_cache distinct substr(upper(p_name),1,5) from part
    -> where lower(p_name) like '%cornflower%' or lower(p_name) like '%goldenrod%';
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
                |
+----+...
+-------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using parallel query (2 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------
+
```

Le stesse considerazioni sono valide per altre espressioni, come espressioni `CASE` o operatori `LIKE`. L'esempio seguente mostra una query in parallelo che valuta l'espressione `CASE` e gli operatori `LIKE` nella clausola `WHERE`.

```
mysql> explain select p_mfgr, p_retailprice from part
    -> where p_retailprice > case p_mfgr
```

```
    ->   when 'Manufacturer#1' then 1000
    ->   when 'Manufacturer#2' then 1200
    ->   else 950
    -> end
    -> and p_name like '%vanilla%'
    -> group by p_retailprice;
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                |
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (4
 columns, 0 filters, 2 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

LIMITClausola

Le query in parallelo non sono correntemente utilizzate per blocchi di query che includono una clausola `LIMIT`. Le query in parallelo possono ancora essere utilizzate per fasi di query precedenti con le clausole `GROUPBY`, `ORDER BY` oppure con join.

Operatori di confronto

L'ottimizzatore stima il numero di righe da analizzare per valutare gli operatori di confronto e determina se utilizzare la funzionalità di query in parallelo in base a quella stima.

Il primo esempio qui sotto mostra che un confronto delle uguaglianze con la colonna di chiave primaria può essere eseguito in modo efficace senza la funzionalità di query in parallelo. Il secondo esempio mostra che un confronto simile con una colonna non indicizzata richiede l'analisi di milioni di righe e che di conseguenza può beneficiare della funzionalità di query in parallelo.

```
mysql> explain select * from part where p_partkey = 10;
+----+...+------+-------+
| id |...| rows | Extra |
+----+...+------+-------+
|  1 |...|    1 | NULL  |
+----+...+------+-------+
```

```
mysql> explain select * from part where p_type = 'LARGE BRUSHED BRASS';
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+
```

Le stesse considerazioni sono valide per i test di diseguaglianza e i confronti di intervalli come "minore di", "maggiore di" o "uguale a" oppure `BETWEEN`. L'ottimizzatore stima il numero di righe da analizzare e in base al volume globale di operazioni di I/O determina se l'utilizzo della funzionalità di query in parallelo è giustificato.

Join

Le query join con tabelle di grandi dimensioni comportano in genere operazioni con grandi quantità di dati che beneficiano dell'ottimizzazione mediante query in parallelo. I confronti dei valori di colonna tra molteplici tabelle (ovvero i predicati di join stessi) non sono attualmente parallelizzati. La funzionalità di query in parallelo può tuttavia trasferire una parte dell'elaborazione interna per altre fasi di join, come la costruzione del filtro Bloom durante un hash join. Inoltre, può essere applicata a query join anche senza una clausola `WHERE`. Di conseguenza, una query join è un'eccezione alla regola secondo cui una clausola `WHERE` è necessaria per utilizzare la query in parallelo.

Ogni fase dell'elaborazione di un join è valutata per determinare se è idonea per la funzionalità di query in parallelo. Se più fasi possono utilizzare tale funzionalità, sono eseguite in sequenza. Ogni query join viene pertanto contabilizzata come singola sessione di query in parallelo in termini di limiti di simultaneità.

Ad esempio, quando una query join include predicati `WHERE` per filtrare le righe di una delle tabelle unite in join, tale opzione di filtraggio può utilizzare la funzionalità di query in parallelo. Un altro esempio consiste in una query join che utilizza il meccanismo di hash join per unire in join una tabella di grandi dimensioni e una di piccole dimensioni. In tal caso, la funzionalità di query in parallelo potrebbe essere applicata all'analisi delle tabelle per generare la struttura di dati del filtro Bloom.

> **Note**
>
> La query in parallelo viene in genere utilizzata per quei tipi di query che richiedono un uso intensivo di risorse che traggono vantaggio dall'ottimizzazione dell'hash join. Il metodo per attivare l'ottimizzazione dell'hash join dipende dalla versione di Aurora MySQL. Per informazioni dettagliate su ogni versione, consultare Abilitazione dell'hash join per cluster di query parallele. Per informazioni su come utilizzare i join hash in modo efficace, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

```
mysql> explain select count(*) from orders join customer where o_custkey = c_custkey;
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
| id |...| table    | type  | possible_keys | key         |...| rows      | Extra

              |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
|  1 |...| customer | index | PRIMARY       | c_nationkey |...|  15051972 | Using index

              |
|  1 |...| orders   | ALL   | o_custkey     | NULL        |...| 154545408 | Using join
 buffer (Hash Join Outer table orders); Using parallel query (1 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
```

Per una query join che utilizza il meccanismo di ciclo nidificato, il blocco di ciclo nidificato più esterno può utilizzare la funzionalità di query in parallelo. L'utilizzo delle query in parallelo dipende dai fattori usuali, come la presenza di condizioni di filtro supplementari nella clausola `WHERE`.

```
mysql> -- Nested loop join with extra filter conditions can use parallel query.
mysql> explain select count(*) from part, partsupp where p_partkey != ps_partkey and
 p_name is not null and ps_availqty > 0;
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------+
| id | select_type | table    |...| rows     | Extra
                                  |
```

```
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part     |...| 20427936 | Using where; Using parallel query (2
 columns, 1 filters, 0 exprs; 0 extra) |
|  1 | SIMPLE      | partsupp |...| 78164450 | Using where; Using join buffer (Block
 Nested Loop)                          |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
```

Sottoquery

Il blocco di query esterno e il blocco di sottoquery interno possono utilizzare ciascuna query parallela o meno. Se lo fanno si basa sulle solite caratteristiche della tabella, clausola `WHERE` e così via, per ogni blocco. Ad esempio, la query seguente utilizza tale funzionalità per il blocco di sottoquery ma non per il blocco esterno.

```
mysql> explain select count(*) from part where
   --> p_partkey < (select max(p_partkey) from part where p_name like '%vanilla%');
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type |...| rows     | Extra
                        |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY     |...|     NULL | Impossible WHERE noticed after reading const tables
                       |
|  2 | SUBQUERY    |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

Attualmente, le sottoquery correlate non possono utilizzare l'ottimizzazione mediante query in parallelo.

UNION

Ogni blocco di query in una query `UNION` può utilizzare o meno la funzionalità di query in parallelo in base alle caratteristiche abituali della tabella, della clausola `WHERE`, ecc., per ogni parte della query `UNION`.

```
mysql> explain select p_partkey from part where p_name like '%choco_ate%'
    -> union select p_partkey from part where p_name like '%vanil_a%';
```

```
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type    |...| rows     | Extra
                           |
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY        |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
|  2 | UNION          |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
| NULL | UNION RESULT | <union1,2> |...|     NULL | Using temporary
                        |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Ogni clausola `UNION` nella query viene eseguita in modo sequenziale. Anche se la query include molteplici fasi che utilizzano la funzionalità di query in parallelo, esegue sempre un'unica query in parallelo. Di conseguenza, anche una query complessa con più fasi viene contabilizzata come una sola query per il limite di query in parallelo simultanee.

Visualizzazioni

L'ottimizzatore riscrive le query con una visualizzazione come query più lunga che utilizza le tabelle sottostanti. La query in parallelo funziona quindi nello stesso modo indipendentemente dal fatto che i riferimenti delle tabelle siano visualizzazioni o tabelle reali. Tutte le considerazioni sull'utilizzo o meno della funzionalità di query in parallelo per una query nonché su quali parti vengono trasferite, si applicano alla query riscritta finale.

Ad esempio, il piano query seguente mostra la definizione di una visualizzazione che in genere non utilizza le query in parallelo. Quando la vista viene interrogata con `WHERE` clausole aggiuntive, Aurora My utilizza la query parallela. SQL

```
mysql> create view part_view as select * from part;
mysql> explain select count(*) from part_view where p_partkey is not null;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
```

```
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 0 filters, 0 exprs;
 1 extra) |
+----+...+----------
+-------------------------------------------------------------------------+
```

Istruzioni del linguaggio di manipolazione dei dati () DML

L'istruzione `INSERT` può utilizzare la funzionalità di query in parallelo per la fase `SELECT` dell'elaborazione, se la parte `SELECT` soddisfa le altre condizioni relative a tale funzionalità.

```
mysql> create table part_subset like part;
mysql> explain insert into part_subset select * from part where p_mfgr =
 'Manufacturer#1';
+----+...+----------
+-------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------+
```

> **Note**
>
> In genere, dopo un'istruzione `INSERT`, i dati delle righe appena inserite si trovano nel pool di buffer. Di conseguenza, una tabella può non essere idonea per la funzionalità di query in parallelo subito dopo l'inserimento di un gran numero di righe. Successivamente, dopo la rimozione dei dati dal pool di buffer durante il funzionamento normale, le query sulla tabella possono iniziare a utilizzare di nuovo la funzionalità di query in parallelo.

L'istruzione `CREATE TABLE AS SELECT` non utilizza la funzionalità di query in parallelo anche se la porzione `SELECT` dell'istruzione fosse idonea a farlo. L'DDLaspetto di questa dichiarazione la rende incompatibile con l'elaborazione parallela delle interrogazioni. Nell'istruzione `INSERT ... SELECT`, la porzione `SELECT` può invece utilizzare la funzionalità di query in parallelo.

Le query in parallelo non sono mai utilizzate per le istruzioni `DELETE` o `UPDATE`, indipendentemente dalla dimensione della tabella e dei predicati nella clausola `WHERE`.

```
mysql> explain delete from part where p_name is not null;
+----+-------------+...+----------+-------------+
| id | select_type |...| rows     | Extra       |
+----+-------------+...+----------+-------------+
|  1 | SIMPLE      |...| 20427936 | Using where |
+----+-------------+...+----------+-------------+
```

Transazioni e blocco

Puoi utilizzare tutti i livelli di isolamento sull'istanza primaria Aurora.

Sulle istanze database di lettura Aurora, la query parallela si applica alle istruzioni eseguite nel livello di isolamento `REPEATABLE READ`. Aurora My SQL versione 2.09 o successiva può anche utilizzare il livello di `READ COMMITTED` isolamento sulle istanze Reader DB. `REPEATABLE READ`è il livello di isolamento predefinito per le istanze DB del lettore Aurora. Per utilizzare il livello di isolamento `READ COMMITTED` sulle istanze DB del lettore è necessario impostare l'opzione di configurazione `aurora_read_replica_read_committed` a livello di sessione. Il livello di `READ COMMITTED` isolamento per le istanze Reader è conforme al comportamento standard. SQL Tuttavia, l'isolamento è meno rigoroso per le istanze di lettura rispetto a quando le query utilizzano il livello di isolamento `READ COMMITTED` sull'istanza di scrittura.

Per ulteriori informazioni sui livelli di isolamento Aurora, in particolare sulle differenze su `READ COMMITTED` fra le istanze di scrittura e lettura, consultare Livelli di isolamento di Aurora MySQL.

Al termine di una transazione di grandi dimensioni, le statistiche della tabella possono non essere aggiornate. Queste statistiche possono necessitare di un'istruzione `ANALYZE TABLE` affinché Aurora sia in grado di stimare accuratamente il numero di righe. Un'DMListruzione su larga scala potrebbe anche portare una parte sostanziale dei dati della tabella nel buffer pool. La presenza di questi dati nel pool di buffer può implicare un utilizzo meno frequente della funzionalità di query in parallelo per quella tabella fino a che i dati non vengono rimossi dal pool.

Quando la tua sessione è in una transazione di lunga durata (per impostazione predefinita, 10 minuti), le altre query in quella sessione non utilizzano la funzionalità di query in parallelo. È anche possibile che si verifichi un timeout durante una singola query di lunga durata. Questo tipo di timeout può verificarsi se la durata della query è più lunga dell'intervallo massimo (attualmente 10 minuti) prima dell'inizio dell'elaborazione di query in parallelo.

Puoi limitare il rischio di un'esecuzione accidentale delle transazioni di lunga durata impostando `autocommit=1` nelle sessioni `mysql` in cui esegui query ad hoc (uniche). Anche un'istruzione `SELECT` relativa a una tabella inizia una transazione creando una visualizzazione di lettura. Una visualizzazione di lettura è un set di dati coerente per le query successive che viene conservato fino al commit della transazione. Tieni presente questa restrizione anche quando usi JDBC o ODBC applicazioni con Aurora, poiché tali applicazioni potrebbero essere eseguite con `autocommit` l'impostazione disattivata.

L'esempio seguente mostra come, con l'impostazione `autocommit` disattivata, l'esecuzione di una query su una tabella crea una visualizzazione di lettura che implicitamente avvia una transazione. Le query eseguite subito dopo possono ancora utilizzare la funzionalità di query in parallelo. Tuttavia, dopo una pausa di alcuni minuti, le query non sono più idonee per tale funzionalità. Termina la transazione con `COMMIT` o `ROLLBACK` per ripristinare l'idoneità.

```
mysql> set autocommit=0;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+

mysql> select sleep(720); explain select sql_no_cache count(*) from part where
 p_retailprice > 10.0;
+------------+
| sleep(720) |
+------------+
|          0 |
+------------+
1 row in set (12 min 0.00 sec)

+----+...+---------+-------------+
| id |...| rows    | Extra       |
+----+...+---------+-------------+
|  1 |...| 2976129 | Using where |
+----+...+---------+-------------+
```

```
mysql> commit;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+
```

Per determinare quante volte le query non erano idonee per le query in parallelo perché parte di transazioni di lunga durata, verifica la variabile di stato `Aurora_pq_request_not_chosen_long_trx`.

```
mysql> show global status like '%pq%trx%';
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| Aurora_pq_request_not_chosen_long_trx | 4     |
+---------------------------------+-------+
```

Qualsiasi istruzione `SELECT` che acquisisce blocchi, come la sintassi `SELECT FOR UPDATE` o `SELECT LOCK IN SHARE MODE`, non può utilizzare la funzionalità di query in parallelo.

Le query in parallelo possono essere utilizzate con una tabella bloccata da un'istruzione `LOCK TABLES`.

```
mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055';
+----+...+-----------
+-------------------------------------------------------------------------------+
| id |...| rows      | Extra
           |
+----+...+-----------
+-------------------------------------------------------------------------------+
|  1 |...| 154545408 | Using where; Using parallel query (3 columns, 1 filters, 0
 exprs; 0 extra) |
```

```
+----+...+-----------
+-------------------------------------------------------------------------------+

mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055' for update;
+----+...+-----------+-------------+
| id |...| rows      | Extra       |
+----+...+-----------+-------------+
|  1 |...| 154545408 | Using where |
+----+...+-----------+-------------+
```

Indici B-Tree

Le statistiche raccolte da un'istruzione `ANALYZE TABLE` consentono all'ottimizzatore di stabilire quando utilizzare la funzionalità di query in parallelo o le ricerche di indice in funzione delle caratteristiche dei dati di ogni colonna. Mantieni aggiornate le statistiche eseguendo `ANALYZE TABLE` DML operazioni che apportano modifiche sostanziali ai dati all'interno di una tabella.

Se le ricerche di indice sono in grado di eseguire efficacemente una query senza l'analisi di un grande volume di dati, Aurora può utilizzare le ricerche di indice. In questo modo, si evita l'overhead dell'elaborazione di query in parallelo. Esistono inoltre limiti di simultaneità sul numero di query in parallelo che possono essere eseguite simultaneamente su qualsiasi cluster di database Aurora. Assicurati di seguire le best practice per l'indicizzazione delle tabelle, di modo che le query più frequenti e che utilizzano maggiormente la simultaneità ricorrano alle ricerche di indice.

Indici di ricerca a testo completo (FTS)

Al momento, le query in parallelo non sono utilizzate per le tabelle che contengono un indice di ricerca full-text, indipendentemente dal fatto che la query faccia riferimento a tali colonne indicizzate o che utilizzi l'operatore `MATCH`.

Colonne virtuali

Attualmente, la query parallela non viene utilizzata per le tabelle che contengono una colonna virtuale, indipendentemente dal fatto che la query faccia riferimento a colonne virtuali.

Meccanismi di caching integrati

Aurora include meccanismi di caching integrati, ovvero il pool di buffer e la cache di query L'ottimizzatore Aurora sceglie tra questi meccanismi di caching e la funzionalità di query in parallelo a seconda di quale è più efficace per una determinata query.

Quando una query in parallelo filtra le righe e trasforma ed estrae i valori di colonna, i dati sono ritrasmessi al nodo head come tuple anziché pagine di dati. Di conseguenza, l'esecuzione di una query in parallelo non aggiunge pagine al pool di buffer o rimuove pagine presenti nel pool di buffer.

Aurora verifica il numero di pagine di dati della tabella presenti nel pool di buffer e quale proporzione di quei dati quel numero rappresenta. Aurora si basa su tali informazioni per determinare se è più efficace utilizzare la funzionalità di query parallela (e ignorare i dati nel pool di buffer). In alternativa, Aurora può utilizzare il percorso di elaborazione di query non in parallelo, che si avvale dei dati nella cache del pool di buffer. Le pagine memorizzate nella cache e l'impatto delle query con una grande quantità di dati sulle operazioni di caching e rimozione dipendono dalle impostazioni di configurazione relative al pool di buffer. Può quindi rivelarsi difficile prevedere se una specifica query utilizza la funzionalità di query in parallelo, poiché la scelta dipende dai dati che si trovano nel pool di buffer e che cambiano costantemente.

Inoltre, Aurora impone limiti di simultaneità per le query in parallelo. Poiché non tutte le query utilizzano la funzionalità di query in parallelo, una parte importante dei dati delle tabelle a cui hanno accesso più query simultaneamente si trova nel pool di buffer. Di conseguenza, Aurora spesso non sceglie queste tabelle per le query in parallelo.

Quando esegui una sequenza di query non in parallelo sulla stessa tabella, la prima query può risultare lenta in quanto i dati non sono presenti nel pool di buffer. La seconda query e quelle successive saranno molto più rapide in quanto il pool di buffer conterrà già dei dati. Le query in parallelo in genere forniscono prestazioni costanti sin dalla prima query su una tabella. Quando esegui test di prestazioni, compara le query non in parallelo con un pool di buffer a caldo e uno a freddo. In alcuni casi, i risultati con un pool di buffer a caldo sono comparabili a quelli delle query in parallelo. In questi casi, prendere in considerazione fattori quali la frequenza delle query rispetto a tale tabella. Considerare anche se vale la pena mantenere i dati per quella tabella nel pool di buffer.

La cache di query evita di dover rieseguire una query quando si invia una query identica e i dati di tabella sottostante non vengono modificati. Le query ottimizzate mediante la funzionalità di query in parallelo possono essere memorizzate nella cache di query, in modo da generare risultati istantanei all'esecuzione successiva.

> **Note**
>
> Quando si comparano le prestazioni, la cache di query può generare valori di durata artificialmente bassi. Di conseguenza, nelle comparazioni puoi utilizzare l'hint `sql_no_cache`. Questo hint impedisce la generazione dei risultati a partire dalla cache di query, anche se la stessa query è stata eseguita in precedenza. L'hint è posizionato subito

dopo l'istruzione `SELECT` in una query. Molti esempi di query in parallelo in questo argomento includono questo suggerimento per consentire la comparazione, in termini di durata, delle versioni della stessa query con e senza la funzionalità di query in parallelo abilitata. Assicurati di rimuovere questo hint dal codice sorgente quando utilizzi la funzionalità di query in parallelo in un ambiente di produzione.

Suggerimenti per l'ottimizzazione

Un altro modo per controllare l'ottimizzazione consiste nell'utilizzare i suggerimenti che possono essere specificati in singole istruzioni. Ad esempio, è possibile attivare l'ottimizzazione per una tabella in un'istruzione e quindi disattivare l'ottimizzazione per un'altra tabella. Per ulteriori informazioni su questi suggerimenti, consulta Optimizer Hints nel My Reference Manual. SQL

Puoi usare i SQL suggerimenti con Aurora Le SQL mie query per ottimizzare le prestazioni. È inoltre possibile utilizzare i suggerimenti per impedire che i piani di esecuzione per query importanti vengano modificati a causa di condizioni imprevedibili.

Abbiamo esteso la funzionalità dei SQL suggerimenti per aiutarti a controllare le scelte degli ottimizzatori per i tuoi piani di query. Questi suggerimenti si applicano alle query che utilizzano l'ottimizzazione delle query parallele. Per ulteriori informazioni, consulta Aurora I miei suggerimenti SQL.

Le mie tabelle temporanee ISAM

L'ottimizzazione mediante query in parallelo è applicabile solo alle tabelle InnoDB. Poiché Aurora My SQL utilizza My ISAM dietro le quinte per le tabelle temporanee, le fasi di interrogazione interne che coinvolgono tabelle temporanee non utilizzano mai la query parallela. Queste fasi di query sono indicate da `Using temporary` nell'output `EXPLAIN`.

Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL

Puoi utilizzare la funzionalità Advanced Auditing ad alte prestazioni di Amazon Aurora SQL My per controllare l'attività del database. A questo proposito, devi abilitare la raccolta dei registri di controllo impostando vari parametri del cluster di database. Quando l'audit avanzato è abilitato, puoi utilizzarlo per registrare qualsiasi combinazione di eventi supportati.

Puoi visualizzare o scaricare i registri di verifica per esaminare le informazioni di verifica per un'istanza database alla volta. Per farlo, puoi utilizzare le procedure in Monitoraggio dei file di log di Aurora.

> **Tip**
>
> Per un cluster Aurora DB contenente più istanze database, potrebbe essere più conveniente esaminare i registri di verifica per tutte le istanze del cluster. A tale scopo, puoi utilizzare Logs. CloudWatch È possibile attivare un'impostazione a livello di cluster per pubblicare i dati del registro di SQL controllo Aurora My su un gruppo di log in. CloudWatch È quindi possibile visualizzare, filtrare e cercare i log di controllo tramite l' CloudWatch interfaccia. Per ulteriori informazioni, consulta Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch .

Abilitazione dell'audit avanzato

Utilizza i parametri descritti in questa sezione per abilitare e configurare l'audit avanzato per il cluster di database.

Utilizza il parametro `server_audit_logging` per abilitare o disabilitare Advanced Auditing.

Utilizza il parametro `server_audit_events` per specificare quali eventi registrare.

Utilizzare i parametri `server_audit_incl_users` e `server_audit_excl_users` per specificare chi deve essere controllato. Per impostazione predefinita, vengono verificati tutti gli utenti. Per informazioni dettagliate su come funzionano questi parametri quando uno o entrambi sono lasciati vuoti o gli stessi nomi utente sono specificati in entrambi, consulta server_audit_incl_users e server_audit_excl_users.

Configura l'audit avanzato impostando questi parametri nel gruppo di parametri utilizzato dal tuo cluster di database. È possibile utilizzare la procedura mostrata in Modifica dei parametri in un gruppo di parametri DB in per modificare i parametri del cluster DB utilizzando il AWS Management Console. Puoi usare il modify-db-cluster-parameter-group AWS CLI comando o l'RDSAPIoperazione M odifyDBCluster ParameterGroup Amazon per modificare i parametri del cluster DB a livello di codice.

La modifica di questi parametri non richiede un riavvio del cluster database quando il gruppo di parametri è già associato al cluster. Quando associ il gruppo di parametri al cluster per la prima volta, devi riavviare il cluster.

Argomenti

- server_audit_logging
- server_audit_events
- server_audit_incl_users
- server_audit_excl_users

server_audit_logging

Abilita o disabilita l'audit avanzato. Il valore predefinito di questo parametro èOFF; impostalo su ON per abilitare Advanced Auditing.

Nessun dato di verifica viene visualizzato nei registri a meno che non si definiscano anche uno o più tipi di eventi da sottoporre a verifica utilizzando il parametro `server_audit_events`.

Per confermare che i dati di audit siano registrati per un'istanza database,
verifica che alcuni file di log per quell'istanza abbiano i nomi del modulo `audit/audit.log.`*`other_identifying_information`*. Per visualizzare i nomi dei file di log, segui la procedura descritta in Visualizzazione ed elenco dei file di log del database.

server_audit_events

Contiene un elenco di eventi delimitati da virgola da registrare. Gli eventi devono essere in maiuscolo e non devono esserci spazi bianchi tra gli elementi dell'elenco (ad esempio,: `CONNECT,QUERY_DDL`. L'impostazione predefinita di questo parametro è una stringa vuota.

Puoi registrare qualsiasi combinazione dei seguenti eventi:

- CONNECT— Registra sia le connessioni riuscite che quelle non riuscite e anche le disconnessioni. Questo evento include informazioni sull'utente.
- QUERY— Registra tutte le interrogazioni in formato testo semplice, incluse quelle che hanno esito negativo a causa di errori di sintassi o di autorizzazione.

> **Tip**
>
> Con questo tipo di evento attivato, i dati di verifica includono informazioni sul monitoraggio continuo e sul controllo dell'integrità eseguiti automaticamente da Aurora. Se sei interessato solo a particolari tipi di operazioni, puoi utilizzare i tipi di eventi più specifici. È inoltre possibile utilizzare l' CloudWatchinterfaccia per cercare nei log gli eventi relativi a database, tabelle o utenti specifici.

- QUERY_ DCL — Simile all'QUERYevento, ma restituisce solo le interrogazioni del linguaggio di controllo dei dati (GRANTREVOKE, e così viaDCL).
- QUERY_ DDL — Simile all'QUERYevento, ma restituisce solo le query del linguaggio di definizione dei dati (DDL) (CREATEALTER, e così via).
- QUERY_ DML — Simile all'QUERYevento, ma restituisce solo le interrogazioni del linguaggio di manipolazione dei dati (DML) (INSERTUPDATE, e così via e anche). SELECT
- TABLE— Registra le tabelle interessate dall'esecuzione delle query.

> **Note**
>
> In Aurora non esiste alcun filtro che escluda determinate query dai log di controllo. Per escludere le `SELECT` interrogazioni, è necessario escludere tutte le istruzioni. DML
> Se un determinato utente riporta queste `SELECT` interrogazioni interne nei log di controllo, puoi escluderlo impostando il parametro del cluster DB server_audit_excl_users. Tuttavia, se quell'utente viene utilizzato anche in altre attività e non può essere omesso, non esiste altra opzione per escludere le query. `SELECT`

server_audit_incl_users

Contiene l'elenco di nomi utente delimitati da virgole per gli utenti la cui attività è registrata.
Non ci devono essere spazi bianchi tra gli elementi dell'elenco. Ad esempio: `user_3,user_4`.
L'impostazione predefinita di questo parametro è una stringa vuota. La lunghezza massima è 1024

caratteri. I nomi utente specificati devono corrispondere ai valori nella colonna `User` della tabella `mysql.user`. Per ulteriori informazioni sui nomi utente, consulta Nomi utente e password degli account nella documentazione personale. SQL

Se `server_audit_incl_users` e `server_audit_excl_users` sono vuoti (impostazione predefinita), l'audit riguarderà tutti gli utenti.

Se aggiungi utenti a `server_audit_incl_users` e lasci vuoto `server_audit_excl_users`, saranno controllati solo quegli utenti.

Se aggiungi utenti a `server_audit_excl_users` e lasci vuoto `server_audit_incl_users`, saranno controllati tutti gli utenti, eccetto quelli elencati in `server_audit_excl_users`.

Se aggiungi gli stessi utenti a `server_audit_excl_users` e `server_audit_incl_users`, quegli utenti saranno controllati. Quando lo stesso utente è elencato in entrambe le impostazioni, a `server_audit_incl_users`viene data una priorità maggiore.

Gli eventi di connessione e disconnessione non sono interessati da questa variabile; sono sempre registrati se specificato. Un utente è registrato anche se è specificato nel parametro `server_audit_excl_users`, perché `server_audit_incl_users` ha una priorità maggiore.

server_audit_excl_users

Contiene l'elenco di nomi utente delimitati da virgole per gli utenti la cui attività non è registrata. Non ci devono essere spazi bianchi tra gli elementi dell'elenco. Ad esempio: `rdsadmin,user_1,user_2`. L'impostazione predefinita di questo parametro è una stringa vuota. La lunghezza massima è 1024 caratteri. I nomi utente specificati devono corrispondere ai valori nella colonna `User` della tabella `mysql.user`. Per ulteriori informazioni sui nomi utente, consulta Nomi utente e password degli account nella documentazione personaleSQL.

Se `server_audit_incl_users` e `server_audit_excl_users` sono vuoti (impostazione predefinita), l'audit riguarderà tutti gli utenti.

Se aggiungi utenti a `server_audit_excl_users` e lasci vuoto `server_audit_incl_users`, solo quegli utenti elencati in `server_audit_excl_users` non saranno controllati, mentre tutti gli altri lo saranno.

Se aggiungi gli stessi utenti a `server_audit_excl_users` e `server_audit_incl_users`, quegli utenti saranno controllati. Quando lo stesso utente è elencato in entrambe le impostazioni, a `server_audit_incl_users`viene data una priorità maggiore.

Gli eventi di connessione e disconnessione non sono interessati da questa variabile; sono sempre registrati se specificato. Un utente è registrato se è anche specificato nel parametro `server_audit_incl_users`, in quanto tale impostazione è prioritaria rispetto a `server_audit_excl_users`.

Visualizzazione dei log di audit

Puoi visualizzare e scaricare i log di audit utilizzando la console. Nella pagina Databases (Database), scegliere l'istanza database per visualizzarne i dettagli, quindi scorrere verso la sezione Logs (Log). I registri di verifica prodotti dalla caratteristica Advanced Auditing hanno i nomi del modulo `audit/audit.log.`*`other_identifying_information`*.

Per scaricare un file di log, selezionare il file nella sezione Logs (Log), quindi scegliere Download (Scarica).

È inoltre possibile ottenere un elenco dei file di registro utilizzando il describe-db-log-files AWS CLI comando. È possibile scaricare il contenuto di un file di registro utilizzando la porzione download-db-log-file- AWS CLI comando. Per ulteriori informazioni, consulta Visualizzazione ed elenco dei file di log del database e Download di un file di log di database.

Dettagli dei log di audit

I file di registro sono rappresentati come file di variabili (CSV) separati da virgole in UTF formato -8. Le query sono inoltre racchiuse tra virgolette singole (').

Il registro di controllo viene archiviato separatamente nella memoria locale di ogni istanza Aurora My SQL DB. Ogni istanza distribuisce le scritture su quattro file di log alla volta. La dimensione massima di un file di registro è 100 MB. Quando viene raggiunto questo limite non configurabile, Aurora ruota il file e genera un nuovo file.

> **Tip**
>
> Le voci dei log non sono in ordine sequenziale. Per ordinare le voci, utilizza il valore del timestamp. Per visualizzare gli ultimi eventi, potrebbe essere necessario esaminare tutti i file di log. Per una maggiore flessibilità nell'ordinamento e nella ricerca dei dati di registro, attiva l'impostazione per caricare i registri di controllo CloudWatch e visualizzarli utilizzando l'interfaccia. CloudWatch

Per visualizzare i dati di controllo con più tipi di campi e con output in JSON formato, puoi anche utilizzare la funzione Database Activity Streams. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

I file dei log di audit includono le seguenti informazioni delimitate da virgola, in righe, nell'ordine specificato:

Campo	Descrizione
timestamp	Il timestamp Unix per l'evento registrato con una precisione al microsecondo.
serverhost	Il nome dell'istanza per cui l'evento viene registrato.
username	Il nome utente connesso dell'utente.
host	L'host da cui l'utente ha effettuato la connessione.
connectionid	Il numero di ID di connessione per l'operazione registrata.
queryid	Il numero di ID di query che può essere utilizzato per trovare gli eventi di tabella relazionale e le query correlate. Per gli eventi `TABLE`, vengono aggiunte più righe.
operation	Il tipo di operazione registrata. I valori possibili sono `CONNECT`, `QUERY`, `READ`, `WRITE`, `CREATE`, `ALTER`, `RENAME` e `DROP`.
database	Il database attivo, come impostato dal comando `USE`.
oggetto	Per gli eventi `QUERY`, questo valore indica la query eseguita dal database. Per gli eventi `TABLE`, indica il nome di tabella.
retcode	Il codice restituito dell'operazione di registrazione.

Replica con Amazon Aurora MySQL

Le funzionalità di replica Aurora MySQL sono essenziali per garantire l'elevata disponibilità e le prestazioni avanzate del cluster. Aurora facilita la creazione o il ridimensionamento dei cluster con un massimo di 15 repliche Aurora.

Tutte le repliche utilizzano gli stessi dati sottostanti. Se alcune istanze database passano alla modalità offline, altre rimangono disponibili per continuare a elaborare le query o per essere utilizzate come scrittore, se necessario. Aurora ripartisce automaticamente le connessioni di sola lettura in più istanze database, consentendo a un cluster Aurora di supportare carichi di lavoro che comportano un grande numero di query.

Nei seguenti argomenti vengono fornite informazioni sul funzionamento della replica Aurora MySQL nonché sulla regolazione delle impostazioni di replica per garantire una disponibilità e prestazioni ottimali.

Argomenti

- Utilizzo delle repliche di Aurora
- Opzioni di replica per Amazon Aurora MySQL
- Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL
- Riavvio senza tempi di inattività (ZDR) per Amazon Aurora My SQL
- Configurazione dei filtri di replica con Aurora My SQL
- Monitoraggio della replica Amazon Aurora MySQL.
- Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS
- Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari)
- Utilizzo della replica GTID basata

Utilizzo delle repliche di Aurora

Le repliche Aurora sono endpoint indipendenti in un cluster DB Aurora, utilizzati al meglio per operazioni di dimensionamento della lettura e maggiore disponibilità. È possibile distribuire fino a 15 repliche di Aurora nelle zone di disponibilità sulle quali si estende un cluster di database in una Regione AWS. Sebbene il volume cluster database sia fatto di copie multiple di dati per il cluster database, i dati nel volume cluster sono rappresentati come un volume singolo e logico nell'istanza

primaria e nelle repliche Aurora nel cluster database. Per ulteriori informazioni sulle repliche di Aurora, consulta Repliche di Aurora.

Le repliche Aurora funzionano bene per il dimensionamento della lettura perché sono dedicate completamente a operazioni di lettura nel volume cluster. Le operazioni di lettura sono gestite dall'istanza primaria. Poiché il volume del cluster viene condiviso tra tutte le istanze database nel tuo cluster database Aurora MySQL, non è necessaria alcuna ulteriore azione per replicare una copia dei dati per ciascuna replica Aurora. Al contrario, le repliche di lettura MySQL devono riprodurre, su un singolo thread, tutte le operazioni di scrittura dall'istanza database master al datastore locale. Questa limitazione può influire sulla capacita delle repliche di lettura MySQL di supportare grandi volumi di traffico in lettura.

Con Aurora MySQL, quando una replica Aurora viene eliminata, l'endpoint dell'istanza viene rimosso immediatamente e la replica Aurora viene rimossa dall'endpoint di lettura. Se vi sono istruzioni in esecuzione nella replica Aurora da eliminare, si ha un periodo di tolleranza di tre minuti. Le istruzioni esistenti possono finire correttamente durante un periodo di tolleranza. Quando finisce il periodo di tolleranza, la replica Aurora viene chiusa ed eliminata.

Important

Le repliche Aurora per Aurora MySQL utilizzano sempre il livello di isolamento della transazione predefinita `REPEATABLE READ` per operazioni nelle tabelle InnoDB. Puoi utilizzare il comando `SET TRANSACTION ISOLATION LEVEL` per modificare il livello di transazione solo per l'istanza primaria di un cluster database Aurora MySQL. Questa restrizione evita blocchi a livello di utente nelle repliche Aurora e permette alle repliche Aurora di dimensionarsi per supportare migliaia di connessioni utente attive mantenendo il ritardo di replica al minimo.

Note

Le istruzioni DDL che vengono eseguite sull'istanza primaria possono interrompere le connessioni di database sulle repliche Aurora associate. Se una connessione di replica Aurora sta utilizzando attivamente un oggetto di database, come una tabella e quell'oggetto è modificato nell'istanza primaria utilizzando un'istruzione DDL, la connessione di replica Aurora viene interrotta.

> **Note**
>
> La regione Cina (Ningxia) non supporta le repliche di lettura fra regioni.

Opzioni di replica per Amazon Aurora MySQL

Puoi impostare repliche tra qualsiasi delle seguenti opzioni:

- Due cluster Aurora MySQL DB diversi Regioni AWS, creando una replica di lettura interregionale di un cluster Aurora MySQL DB.

  Per ulteriori informazioni, consulta Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS.
- Due cluster Aurora MySQL DB nello stesso ambiente, Regione AWS utilizzando la replica del log binario MySQL (binlog).

  Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).
- Una istanza database RDS for MySQL come origine e un cluster database Aurora MySQL, creando una replica di lettura Aurora di una istanza database RDS for MySQL.

  È possibile utilizzare questo approccio per apportare modifiche ai dati esistenti e in corso in Aurora MySQL durante la migrazione ad Aurora. Per ulteriori informazioni, consulta Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora.

  È inoltre possibile utilizzare questo approccio per aumentare la scalabilità delle query di lettura per i dati. A tale scopo, eseguire query sui dati utilizzando una o più istanze database all'interno di un cluster Aurora MySQL di sola lettura. Per ulteriori informazioni, consulta Ridimensionamento delle letture per il tuo SQL database My con Amazon Aurora.
- Un cluster Aurora MySQL DB in uno e Regione AWS fino a cinque cluster Aurora MySQL DB di sola lettura Aurora in diverse regioni, creando un database globale Aurora.

  È possibile utilizzare un database globale Aurora per supportare applicazioni con un footprint globale. Il cluster database Aurora MySQL primario dispone di un'istanza di scrittura e fino a 15 repliche Aurora. I cluster database Aurora MySQL secondari di sola lettura possono essere

composti di un massimo di 16 repliche Aurora. Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.

> **Note**
>
> Il riavvio dell'istanza primaria di un cluster database Amazon Aurora riavvia automaticamente anche le repliche Aurora per il cluster database, per ristabilire un punto di ingresso che garantisce coerenza di lettura/scrittura nel cluster database.

Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL

Le funzionalità seguenti consentono di ottimizzare le prestazioni della replica Aurora MySQL.

La funzionalità di compressione dei log di replica riduce automaticamente la larghezza di banda di rete per i messaggi di replica. Poiché ogni messaggio è trasmesso a tutte le repliche Aurora, i vantaggi sono maggiori per i cluster più voluminosi. Questa funzionalità implica un certo overhead della CPU sul nodo scrittore per eseguire la compressione. È sempre abilitato in Aurora MySQL versione 2 e versione 3.

La funzionalità di filtraggio binlog riduce automaticamente la larghezza di banda di rete per i messaggi di replica. Poiché le repliche Aurora non utilizzano informazioni di binlog incluse nei messaggi di replica, quei dati sono omessi dai messaggi inviati a tali nodi.

In Aurora MySQL versione 2, puoi controllare questa funzionalità modificando il parametro `aurora_enable_repl_bin_log_filtering`. Per impostazione predefinita, questo parametro è abilitato. Poiché questa ottimizzazione è concepita per essere trasparente, puoi disattivare questa impostazione solo durante la diagnosi o la risoluzione dei problemi relativi alla replica. Ad esempio per corrispondere al comportamento di un cluster Aurora MySQL meno recente dove questa funzionalità non era disponibile.

In Aurora MySQL versione 3, il filtraggio dei file binlog è sempre abilitato.

Riavvio senza tempi di inattività (ZDR) per Amazon Aurora My SQL

La funzione zero-downtime restart (ZDR) può preservare alcune o tutte le connessioni attive alle istanze DB durante determinati tipi di riavvii. ZDRsi applica ai riavvii eseguiti automaticamente da Aurora per risolvere le condizioni di errore, ad esempio quando una replica inizia a rimanere troppo indietro rispetto all'origine.

> **Important**
>
> Il ZDR meccanismo funziona con la massima diligenza possibile. Le SQL versioni di Aurora My, le classi di istanze, le condizioni di errore, SQL le operazioni compatibili e altri fattori che determinano dove ZDR si applica sono soggetti a modifiche in qualsiasi momento.

ZDRper Aurora My SQL 2.x richiede la versione 2.10 e successive. ZDRè disponibile in tutte le versioni minori di Aurora My SQL 3.x. In Aurora My SQL versione 2 e 3, il ZDR meccanismo è attivato per impostazione predefinita e Aurora non utilizza il parametro. `aurora_enable_zdr`

Aurora riporta sulla pagina Eventi le attività relative al riavvio con tempi di inattività pari a zero. Aurora registra un evento quando tenta di riavviare utilizzando il ZDR meccanismo. Questo evento indica perché Aurora esegue il riavvio. Quindi Aurora registra un altro evento al termine del riavvio. Questo evento finale esegue un report su quanto tempo è durato il processo e quante connessioni sono state conservate o interrotte durante il riavvio. È possibile consultare il registro degli errori del database per visualizzare ulteriori dettagli su ciò che è successo durante il riavvio.

Sebbene le connessioni rimangano intatte dopo un'ZDRoperazione riuscita, alcune variabili e funzionalità vengono reinizializzate. I seguenti tipi di informazioni non vengono conservati tramite un riavvio causato da zero-downtime restart:

- Variabili globali. Aurora ripristina le variabili di sessione, ma non ripristina le variabili globali dopo il riavvio.
- Variabili di stato. In particolare, il valore di attività riportato dallo stato del motore viene ripristinato.
- `LAST_INSERT_ID`.
- Stato `auto_increment` in memoria per le tabelle. Lo stato di incremento automatico in memoria viene reinizializzato. Per ulteriori informazioni sui valori di incremento automatico, vedere My SQL Reference Manual.
- Informazioni diagnostiche dalle tabelle `INFORMATION_SCHEMA` e `PERFORMANCE_SCHEMA`. Queste informazioni diagnostiche vengono visualizzate anche nell'output di comandi come `SHOW PROFILE` e `SHOW PROFILES`.

La tabella seguente mostra le versioni, i ruoli delle istanze e altre circostanze che determinano se Aurora può utilizzare il ZDR meccanismo per il riavvio delle istanze DB nel cluster.

Aurora La mia versione SQL	ZDRvale per lo scrittore?	ZDRvale per i lettori?	ZDRsempre abilitato?	Note
2.x, inferiore a 2.10.0	No	No	N/D	ZDRnon è disponibile per queste versioni.
2.10.0—2.11.0	Sì	Sì	Sì	Aurora ripristina le transazioni in corso sulle connessioni attive. La tua domanda deve riprovare le transazioni. Aurora annulla tutte le connessioni che utilizzan oTLS/SSL, tabelle temporanee, blocchi tabella o blocchi utente.
2.11.1 e versioni successive	Sì	Sì	Sì	Aurora ripristina le transazioni in corso sulle connessioni attive. La tua domanda deve riprovare le transazioni. Aurora annulla tutte le connessioni che utilizzan o tabelle temporanee, blocchi tabella o blocchi utente.
3,01—3,03	Sì	Sì	Sì	Aurora ripristina le transazioni in corso sulle connessioni attive. La tua domanda deve riprovare le transazioni. Aurora annulla tutte le connessioni che utilizzan oTLS/SSL, tabelle temporanee, blocchi tabella o blocchi utente.
3.04 e versioni successive	Sì	Sì	Sì	Aurora ripristina le transazioni in corso sulle connessioni attive. La tua domanda deve riprovare le transazioni.

Aurora La mia versione SQL	ZDRvale per lo scrittore?	ZDRvale per i lettori?	ZDRsempre abilitato?	Note
				Aurora annulla tutte le connessioni che utilizzano tabelle temporanee, blocchi tabella o blocchi utente.

Configurazione dei filtri di replica con Aurora My SQL

Puoi utilizzare i filtri di replica per specificare quali database e tabelle vengono replicati con una replica di lettura. I filtri di replica possono includere database e tabelle nella replica o escluderli dalla replica.

Di seguito sono riportati alcuni casi d'uso per i filtri di replica:

- Per ridurre le dimensioni di una replica di lettura. Con il filtro di replica è possibile escludere i database e le tabelle che non sono necessari nella replica di lettura.
- Per escludere database e tabelle dalle repliche di lettura per motivi di sicurezza.
- Per replicare database e tabelle diversi per casi d'uso specifici in repliche di lettura diverse. Ad esempio, è possibile utilizzare repliche di lettura specifiche per l'analisi o la condivisione.
- Per un cluster DB che ha repliche di lettura in diversi Regioni AWS, per replicare database o tabelle diversi in diversi Regioni AWS.
- Per specificare quali database e tabelle vengono replicati con un cluster Aurora SQL My DB configurato come replica in una topologia di replica in entrata. Per ulteriori informazioni su questa configurazione, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Argomenti

- Impostazione dei parametri di filtro di replica per Aurora My SQL
- Limitazioni del filtro di replica per Aurora My SQL
- Esempi di filtri di replica per Aurora My SQL
- Visualizzazione dei filtri di replica per una replica di lettura

Impostazione dei parametri di filtro di replica per Aurora My SQL

Per configurare i filtri di replica, imposta i seguenti parametri:

- `binlog-do-db` – Replica le modifiche ai log binari specificati. Quando impostate questo parametro per un cluster di origine binlog, vengono replicati solo i log binari specificati nel parametro.
- `binlog-ignore-db` – Non replicare le modifiche ai log binari specificati. Quando il `binlog-do-db` parametro è impostato per un cluster di origine binlog, questo parametro non viene valutato.
- `replicate-do-db` – Replicare le modifiche ai database specificati. Quando si imposta questo parametro per un cluster di replica binlog, vengono replicati solo i database specificati nel parametro.
- `replicate-ignore-db` – Non replicare le modifiche ai database specificati. Quando il `replicate-do-db` parametro è impostato per un cluster di replica binlog, questo parametro non viene valutato.
- `replicate-do-table` – Replicare le modifiche alle tabelle specificate. Quando si imposta questo parametro per una replica di lettura, vengono replicate solo le tabelle specificate nel parametro. Inoltre, quando il `replicate-ignore-db` parametro `replicate-do-db` or è impostato, assicuratevi di includere il database che include le tabelle specificate nella replica con il cluster di replica binlog.
- `replicate-ignore-table` – Non replicare le modifiche alle tabelle specificate. Quando il `replicate-do-table` parametro è impostato per un cluster di replica binlog, questo parametro non viene valutato.
- `replicate-wild-do-table` – Replicare le tabelle in base ai modelli di nome del database e della tabella specificati. I caratteri jolly `%` e `_` sono supportati. Quando il `replicate-ignore-db` parametro `replicate-do-db` or è impostato, assicuratevi di includere il database che include le tabelle specificate nella replica con il cluster di replica binlog.
- `replicate-wild-ignore-table` – Non replicare le tabelle in base ai modelli di nomi di database e tabella specificati. I caratteri jolly `%` e `_` sono supportati. Quando il `replicate-wild-do-table` parametro `replicate-do-table` or è impostato per un cluster di replica binlog, questo parametro non viene valutato.

I parametri vengono valutati nell'ordine in cui sono elencati. Per ulteriori informazioni su come funzionano questi parametri, consulta la mia documentazione: SQL

- Per informazioni generali, consulta Opzioni e variabili del server di replica.

- Per informazioni sulla modalità di valutazione dei parametri di filtro della replica del database, consulta Valutazione delle opzioni di replica a livello di database e registrazione binaria.
- Per informazioni sulla modalità di valutazione dei parametri di filtro replica delle tabelle, consulta Valutazione delle opzioni di replica a livello di tabella.

Per impostazione predefinita, ognuno di questi parametri ha un valore vuoto. In ogni cluster binlog, è possibile utilizzare questi parametri per impostare, modificare ed eliminare i filtri di replica. Quando viene impostato uno di questi parametri, è necessario separare ogni filtro dagli altri con una virgola.

È possibile utilizzare i caratteri jolly `%` e `_` nei parametri `replicate-wild-do-table` e `replicate-wild-ignore-table`. Il carattere jolly `%` corrisponde a un numero qualsiasi di caratteri e il carattere jolly `_` corrisponde a un solo carattere.

Il formato di registrazione binaria dell'istanza database di origine è importante per la replica perché determina il record delle modifiche ai dati. L'impostazione del parametro `binlog_format` determina se la replica è basata su righe o basata su dichiarazione. Per ulteriori informazioni, consulta Configurazione di Aurora per database Single-AZ.

> **Note**
>
> Tutte le istruzioni Data Definition Language (DDL) vengono replicate come istruzioni, indipendentemente dall'`binlog_format`impostazione sull'istanza DB di origine.

Limitazioni del filtro di replica per Aurora My SQL

Le seguenti limitazioni si applicano al filtro di replica per Aurora My: SQL

- I filtri di replica sono supportati solo per Aurora SQL My versione 3.
- Ogni parametro di filtro della replica ha un limite di 2.000 caratteri.
- Le virgole non sono supportate nei filtri di replica.
- Il filtro delle repliche non supporta le transazioni XA.

  Per ulteriori informazioni, consulta Restrizioni sulle transazioni XA nella documentazione personale. SQL

Esempi di filtri di replica per Aurora My SQL

Per configurare il filtro per una replica di lettura, modifica i parametri di filtro replica nel gruppo di parametri del cluster di database associato alla replica di lettura.

> **Note**
>
> Non è consentito modificare un gruppo di parametri del cluster di database predefinito. Se la replica di lettura usa un gruppo di parametri predefinito, creare un nuovo gruppo di parametri e associarlo alla replica di lettura. Per ulteriori informazioni sui gruppi di parametri del cluster di database, consulta .

È possibile impostare i parametri in un gruppo di parametri del cluster DB utilizzando AWS Management Console, AWS CLI, oppure RDSAPI. Per informazioni sull'estensione dei parametri consulta Modifica dei parametri in un gruppo di parametri DB in . Quando si impostano parametri in un gruppo di parametri del cluster di database, tutti i cluster di database associati al gruppo di parametri utilizzano le impostazioni dei parametri. Se imposti i parametri di filtro replica in un gruppo di parametri del cluster di database, assicurati che il gruppo di parametri sia associato solo alle repliche di lettura. Lasciare vuoti i parametri di filtro di replica per le istanze database di origine.

I seguenti esempi impostano i parametri utilizzando AWS CLI. Questi esempi sono impostati `ApplyMethod` `immediate` in modo che le modifiche ai parametri avvengano immediatamente dopo il completamento del CLI comando. Se si desidera applicare una modifica in sospeso dopo il riavvio della replica di lettura, impostare `ApplyMethod` su `pending-reboot`.

Gli esempi seguenti impostano i filtri di replica:

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Excluding databases from replication
- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example Inclusione dei database nella replica

Nell'esempio seguente sono inclusi i database `mydb1` e `mydb2` nella replica.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Example Inclusione delle tabelle nella replica

Nell'esempio seguente sono incluse le tabelle `table1` e `table2` nel database `mydb1` nella replica.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Example Inclusione di tabelle nella replica utilizzando caratteri jolly

Nell'esempio seguente sono incluse tabelle con nomi che iniziano con `order` e `return` nel database `mydb` nella replica.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Example Esclusione di database dalla replica

Nell'esempio seguente vengono esclusi i database `mydb5` e `mydb6` dalla replica.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6,ApplyMethod=immediate"
```

Example Esclusione di tabelle dalla replica

Nell'esempio seguente vengono escluse dalla replica le tabelle `table1` nel database `mydb5` e `table2` nel database `mydb6`.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
```

```
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Example Esclusione di tabelle dalla replica utilizzando caratteri jolly

Nell'esempio seguente vengono escluse le tabelle con nomi che iniziano con `order` e `return` nel database `mydb7` dalla replica.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Visualizzazione dei filtri di replica per una replica di lettura

È possibile visualizzare i filtri di replica per una replica di lettura nei seguenti modi:

- Controllare le impostazioni dei parametri di filtro replica nel gruppo di parametri associato alla replica di lettura.

  Per istruzioni, consulta Visualizzazione dei valori dei parametri per un gruppo di parametri DB in Aurora.
- In un My SQL client, connettiti alla replica di lettura ed esegui l'`SHOW REPLICA STATUS`istruzione.

Nell'output, i campi seguenti mostrano i filtri di replica per la replica di lettura:

- `Binlog_Do_DB`
- `Binlog_Ignore_DB`
- `Replicate_Do_DB`
- `Replicate_Ignore_DB`
- `Replicate_Do_Table`
- `Replicate_Ignore_Table`
- `Replicate_Wild_Do_Table`
- `Replicate_Wild_Ignore_Table`

Per ulteriori informazioni su questi campi, vedere Verifica dello stato della replica nella documentazione personaleSQL.

Monitoraggio della replica Amazon Aurora MySQL.

Il dimensionamento di lettura e l'alta disponibilità dipendono da un periodo di ritardo minimo. Puoi monitorare il ritardo di una replica Aurora rispetto all'istanza principale del tuo cluster Aurora MySQL DB monitorando la metrica Amazon. CloudWatch `AuroraReplicaLag` La metrica `AuroraReplicaLag` viene registrata in ogni replica Aurora.

L'istanza DB principale registra anche le CloudWatch metriche `AuroraReplicaLagMaximum` e `AuroraReplicaLagMinimum` Amazon. La metrica `AuroraReplicaLagMaximum` registra la quantità massima di ritardo tra l'istanza DB primaria e ogni replica Aurora nel cluster DB. La metrica `AuroraReplicaLagMinimum` registra la quantità minima di ritardo tra l'istanza DB primaria e ogni replica Aurora nel cluster DB.

Se hai bisogno del valore più recente per il ritardo di Aurora Replica, puoi controllare la metrica `AuroraReplicaLag` in Amazon. CloudWatch Il ritardo replica di Aurora viene anche registrato su ogni replica Aurora del cluster database Aurora MySQL nella tabella `information_schema.replica_host_status`. Per ulteriori informazioni su questa tabella, consultare information_schema.replica_host_status.

Per ulteriori informazioni sul monitoraggio delle istanze e dei parametri RDS, consulta. CloudWatch Monitoraggio dei parametri in un cluster di database Amazon Aurora

Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS

Puoi creare un cluster Amazon Aurora MySQL DB come replica di lettura in un cluster DB diverso da quello di origine. Regione AWS L'adozione di questo approccio può migliorare le capacità di disaster recovery, consentire di scalare le operazioni di lettura per renderle più vicine agli utenti e semplificare la migrazione da una Regione AWS all'altra. Regione AWS

Puoi creare repliche di lettura dei cluster di database crittografati e non crittografati. La replica di lettura deve essere crittografata se il cluster di database di origine è crittografato.

Per ogni cluster di database di origine, puoi avere fino a cinque cluster database tra regioni che sono repliche di lettura.

> **Note**
>
> In alternativa alle repliche di lettura tra regioni, è possibile dimensionare le operazioni di lettura con ritardi minimi utilizzando un database globale Aurora. Un database globale Aurora ha un cluster Aurora DB primario in uno Regione AWS e fino a 10 cluster DB secondari di sola lettura in diverse regioni. Ogni cluster database secondario può includere fino a 16 repliche Aurora (anziché 15). La replica dal cluster database primario a tutti i secondari viene gestita dal livello di archiviazione di Aurora anziché dal motore del database, pertanto il ritardo per la replica delle modifiche è minimo, in genere inferiore a 1 secondo. Mantenere il motore di database fuori dal processo di replica significa che il motore di database è dedicato all'elaborazione dei carichi di lavoro. Significa inoltre che non è necessario configurare o gestire la replica binlog (registrazione dei log binari) di Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.

Quando si crea una replica di lettura del cluster Aurora MySQL DB in un altro Regione AWS, è necessario tenere presente quanto segue:

- Il cluster di database di origine e il cluster di database di replica di lettura tra regione può avere fino a 15 repliche Aurora assieme all'istanza primaria per il cluster database. Utilizzando questa funzionalità, è possibile scalare le operazioni di lettura sia per l'origine che per la destinazione di Regione AWS replica. Regione AWS
- In uno scenario con più regioni, si verifica un ritardo maggiore tra il cluster di database di origine e la replica di lettura a causa della maggiore lunghezza dei canali di rete tra Regioni AWS.

- I dati trasferiti per repliche tra regioni comportano costo di trasferimento dei dati Amazon RDS. Le seguenti operazioni di replica tra regioni generano addebiti per i dati trasferiti fuori dalla Regione AWS di origine:
  - Quando crei la replica di lettura, Amazon RDS acquisisce uno snapshot del cluster di origine e lo trasferisce a Regione AWS quello che contiene la replica di lettura.
  - Per ogni modifica dei dati effettuata nei database di origine, Amazon RDS trasferisce i dati dalla regione di origine a Regione AWS quella che contiene la replica di lettura.

  Per ulteriori informazioni sui prezzi del trasferimento dati Amazon RDS, consulta Prezzi di Amazon Aurora.
- Puoi eseguire più operazioni di creazione o eliminazione simultanee per repliche di lettura che fanno riferimento allo stesso cluster di database di origine. Tuttavia, devi rimanere entro il limite di cinque repliche di lettura per ogni cluster di database di origine.
- Per un efficace funzionamento della replica, ciascuna replica di lettura deve avere la stessa quantità di risorse di calcolo e storage del cluster di database di origine. Se si dimensiona il cluster di database di origine, si devono dimensionare anche le repliche di lettura.

Argomenti

- Prima di iniziare
- Creazione di un cluster DB di replica di lettura interregionale per Aurora My SQL
- Promozione di una replica di lettura su un cluster DB per Aurora My SQL
- Risoluzione dei problemi delle repliche interregionali per Amazon Aurora My SQL

Prima di iniziare

Prima di poter creare un cluster di database di Aurora MySQL che è una replica di lettura tra regioni, è necessario abilitare l'accesso binario al cluster di database fonte di Aurora MySQL. La replica tra regioni per Aurora MySQL utilizza la replica binaria MySQL per riprodurre le modifiche nel cluster di database di replica di lettura tra regioni.

Per abilitare l'accesso binario a un cluster database Aurora MySQL, aggiornare il parametro `binlog_format` per il cluster database fonte. Il parametro `binlog_format` è un parametro a livello di cluster che si trova nel gruppo di parametri cluster predefinito. Se il cluster database utilizza il gruppo di parametri cluster database predefinito, crea un nuovo gruppo di parametri cluster database per modificare le impostazioni `binlog_format`. Consigliamo di impostare `binlog_format` su

`MIXED`. Tuttavia, puoi anche impostare `binlog_format` su `ROW` o `STATEMENT` se hai bisogno di un formato binlog specifico. Riavvia il cluster database Aurora affinché venga applicata la modifica.

Per ulteriori informazioni sull'utilizzo di registrazione binaria con Aurora, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari). Per ulteriori informazioni sulla modifica dei parametri di configurazione di Aurora MySQL, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora e .

Creazione di un cluster DB di replica di lettura interregionale per Aurora My SQL

Puoi creare un cluster Aurora DB che è una replica di lettura interregionale utilizzando AWS Management Console, the AWS Command Line Interface (AWS CLI) o Amazon. RDS API Puoi creare repliche di lettura tra regioni da cluster di database crittografati e non crittografati.

Quando crei una replica di lettura interregionale per Aurora My SQL utilizzando, AWS Management Console Amazon RDS crea un cluster DB nella destinazione Regione AWS, quindi crea automaticamente un'istanza DB che è l'istanza principale per quel cluster DB.

Quando crei una replica di lettura interregionale utilizzando AWS CLI o RDSAPI, devi prima creare il cluster DB nella destinazione Regione AWS e attendere che diventi attivo. Quando è attivo, crei un'istanza database che è l'istanza primaria per il cluster database.

La replica inizia quando l'istanza primaria del cluster di database di replica di lettura diventa disponibile.

Utilizzare le seguenti procedure per creare una replica di lettura interregionale da un cluster Aurora My DB. SQL Queste procedure funzionano per la creazione di repliche di lettura da cluster di database crittografati o non crittografati.

Console

Per creare un cluster Aurora My SQL DB che sia una replica di lettura interregionale con AWS Management Console

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra di AWS Management Console, seleziona Regione AWS quello che ospita il tuo cluster DB di origine.
3. Nel riquadro di navigazione, scegli Databases (Database).
4. Seleziona il cluster database per cui vuoi creare una replica di lettura tra Regioni.

5. In Actions (Operazioni), scegli Create cross region read replica (Crea replica di lettura tra regioni).
6. Nella pagina Create cross region read replica (Crea replica di lettura tra regioni), scegliere le impostazioni di opzione per il cluster di database di replica di lettura tra regioni, come descritto nella seguente tabella.

Opzione	Descrizione
Regione di destinazione	Scegli Regione AWS per ospitare il nuovo cluster DB di replica di lettura interregionale.
Destination DB subnet group (Gruppo di sottoreti del database di destinazione)	Seleziona il gruppo di sottoreti di database da utilizzar e per il cluster di database di replica di lettura tra regioni.
Accessibile pubblicamente	Seleziona Yes (Sì) per assegnare al cluster di database di replica di lettura tra regioni un indirizzo IP pubblico; altrimenti, seleziona No.
Encryption (Crittografia)	Selezionare Enable encryption (Abilita crittografia) per abilitare la crittografia dei dati inattivi del cluster database. Per ulteriori informazioni, consulta Crittogra fia delle risorse Amazon Aurora.
AWS KMS key	Disponibile solo se Encryption (Crittografia) è impostato su Enable encryption (Abilita crittografia). Seleziona AWS KMS key da utilizzare per crittografare questo cluster DB. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
DB instance class (Classe istanza database)	Scegliere una classe di istanza database che definisca i requisiti di elaborazione e di memoria per l'istanza primaria nel cluster DB. Per ulteriori informazioni sulle opzioni di classe di istanza database, consulta Classi di istanze DB Amazon Aurora.
Multi-AZ deployment (Implemen tazione Multi-AZ)	Scegli Sì per creare una replica di lettura del nuovo cluster DB in un'altra zona di disponibilità nella

Opzione	Descrizione
	destinazione Regione AWS per il supporto del failover. Per altre informazioni sulle zone di disponibilità multiple, consulta Regioni e zone di disponibilità.
Read replica source (Origine replica di lettura)	Seleziona il cluster di database di origine per il quale creare una replica di lettura tra regioni.
DB instance identifier (Identificatore istanze DB)	Digita un nome per l'istanza primaria nel cluster di database di replica di lettura tra regioni. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint per l'istanza primaria del nuovo cluster database. L'identificatore istanze database presenta i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere univoca per tutte le istanze DB Account AWS, per ciascuna. Regione AWS In quanto il cluster di database di replica di lettura tra regione viene creato da una snapshot del cluster database origine, il nome utente master e la password master per la replica di lettura corrispondono al nome utente master e alla password master per il cluster di database di origine.

Opzione	Descrizione
DB Cluster Identifier (Identificatore cluster DB)	Digita un nome per il cluster database di replica di lettura tra Regioni che sia univoco per l'account nella Regione AWS di destinazione per la replica. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint del cluster per il cluster database. Per informazioni sull'endpoint del cluster, consulta Connessioni endpoint Amazon Aurora. L'identificatore del cluster DB prevede i seguenti vincoli: • Deve contenere da 1 a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere unico per tutti i cluster DB per ciascuno Account AWS, per ciascuno. Regione AWS
Priorità	Seleziona una priorità di failover per l'istanza primaria del nuovo cluster database. Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Se non si specifica alcun valore, l'impostazione predefinita è tier-1 (livello 1). Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. Per impostazione predefinita, i cluster Aurora DB utilizzano la SQL porta My, 3306. I firewall presso alcune aziende bloccano le connessioni a questa porta. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster database.

Opzione	Descrizione
Enhanced Monitoring (Monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster database. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Scegli il IAM ruolo che hai creato per consentire RDS ad Amazon di comunicare con Amazon CloudWatch Logs per te oppure scegli Default per RDS creare un ruolo a te assegnato. `rds-monitoring-role` Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se Enhanced Monitoring (Monitoraggio avanzato) è impostato su Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Questa impostazione non si applica ai cluster Aurora My SQL DB. Per ulteriori informazioni sugli aggiornamenti del motore per Aurora MySQL, consulta. Aggiornamenti del motore di database per Amazon Aurora My SQL

7. Seleziona Create (Crea) per creare la replica di lettura tra regioni per Aurora.

AWS CLI

Per creare un cluster Aurora My SQL DB che sia una replica di lettura interregionale con CLI

1. Chiama il AWS CLI create-db-clustercomando nel punto in Regione AWS cui desideri creare il cluster DB di replica di lettura. Includi l'`--replication-source-identifier`opzione e specifica l'Amazon Resource Name (ARN) del cluster DB di origine per cui creare una replica di lettura.

   Per la replica tra regioni dove il cluster di database identificato tramite `--replication-source-identifier` è crittografato, è necessario specificare anche le opzioni `--kms-key-id` e `--storage-encrypted`.

   > **Note**
   >
   > Puoi impostare repliche tra regioni da un cluster di database crittografato a una replica di lettura crittografata specificando `--storage-encrypted` e fornendo un valore per `--kms-key-id`.

   Non puoi specificare i parametri `--master-username` e `--master-user-password`. Tali valori vengono presi dal cluster di database di origine.

   Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database non crittografato nella regione us-west-2. Il comando viene chiamato nella regione Stati Uniti orientali 1. In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-cluster \
     --db-cluster-identifier sample-replica-cluster \
     --engine aurora-mysql \
     --engine-version 8.0.mysql_aurora.3.08.0 \
     --replication-source-identifier arn:aws:rds:us-
   west-2:123456789012:cluster:sample-master-cluster
   ```

In Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database crittografato nella regione us-west-2. Il comando viene chiamato nella regione Stati Uniti orientali 1.

In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster \
  --kms-key-id my-us-east-1-key \
  --storage-encrypted
```

In Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster ^
  --kms-key-id my-us-east-1-key ^
  --storage-encrypted
```

L'`--source-region`opzione è necessaria per la replica interregionale tra le regioni AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali), in cui il cluster DB identificato da è crittografato. `--replication-source-identifier` Per`--source-region`, specificare il cluster DB Regione AWS di origine.

Se non si specifica `--source-region`, è necessario specificare un valore per `--pre-signed-url`. Un prefirmato URL è un comando URL che contiene una richiesta firmata Signature Version 4 per il `create-db-cluster` comando richiamato nell'origine Regione AWS. Per ulteriori informazioni sull'`pre-signed-url`opzione, consulta create-db-clusterla sezione AWS CLI Command Reference.

2. Verificate che il cluster DB sia diventato disponibile per l'uso utilizzando il AWS CLI describe-db-clusterscomando, come illustrato nell'esempio seguente.

```
aws rds describe-db-clusters --db-cluster-identifier sample-replica-cluster
```

Quando i risultati **`describe-db-clusters`** mostrano uno stato `available`, crea l'istanza primaria per il cluster database affinché la replica possa iniziare. A tale scopo, utilizzate il AWS CLI create-db-instancecomando come illustrato nell'esempio seguente.

In Linux, macOS, oppure Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier sample-replica-cluster \
  --db-instance-class db.r5.large \
  --db-instance-identifier sample-replica-instance \
  --engine aurora-mysql
```

In Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier sample-replica-cluster ^
  --db-instance-class db.r5.large ^
  --db-instance-identifier sample-replica-instance ^
  --engine aurora-mysql
```

Quando l'istanza database viene creata e diventa disponibile, inizia la replica. È possibile determinare se l'istanza DB è disponibile chiamando il AWS CLI describe-db-instancescomando.

RDS API

Per creare un cluster Aurora My SQL DB che sia una replica di lettura interregionale con API

1. Chiama l'reateDBClusteroperazione RDS API C nel punto in Regione AWS cui desideri creare il cluster DB di replica di lettura. Includi il `ReplicationSourceIdentifier` parametro e specifica l'Amazon Resource Name (ARN) del cluster DB di origine per cui creare una replica di lettura.

   Per la replica tra regioni dove il cluster di database identificato da `ReplicationSourceIdentifier` è crittografato, specifica il parametro `KmsKeyId` e impostare il parametro `StorageEncrypted` su `true`.

   > **Note**
   >
   > Puoi configurare repliche tra regioni da un cluster di database crittografato a una replica di lettura crittografata specificando `StorageEncrypted` come **`true`** e fornendo un valore per `KmsKeyId`. In questo caso, non devi specificare `PreSignedUrl`.

   Non devi includere i parametri `MasterUsername` e `MasterUserPassword`, perché quei valori sono presi dal cluster database origine.

   Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database non crittografato nella regione us-west-2. Questa operazione viene chiamata nella regione us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
```

```
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Il seguente esempio di codice crea una replica di lettura nella regione us-east-1 da uno snapshot di cluster di database crittografato nella regione us-west-2. Questa operazione viene chiamata nella regione us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &KmsKeyId=my-us-east-1-key
  &StorageEncrypted=true
  &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBCluster
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526ReplicationSourceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster%25253Asample-master-cluster
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-
west-2%252Frds%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-
amz-content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Per la replica interregionale tra le regioni AWS GovCloud (Stati Uniti orientali) e AWS GovCloud (Stati Uniti occidentali), in cui il cluster DB identificato da `ReplicationSourceIdentifier` è crittografato, specifica anche il parametro. `PreSignedUrl` La prefirmata URL deve essere una richiesta valida per l'`CreateDBCluster`APIoperazione che può essere eseguita nell'origine Regione AWS che contiene il cluster DB crittografato da replicare. L'identificatore di KMS chiave viene utilizzato per crittografare la replica di lettura e deve essere una KMS chiave valida per la destinazione. Regione AWS Per generare automaticamente anziché manualmente un predefinitoURL, utilizzate invece il AWS CLI create-db-clustercomando con l'opzione. `--source-region`

2. Verificate che il cluster DB sia diventato disponibile per l'uso utilizzando l'escribeDBClustersoperazione RDS API D, come illustrato nell'esempio seguente.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=DescribeDBClusters
  &DBClusterIdentifier=sample-replica-cluster
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T002223Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=84c2e4f8fba7c577ac5d820711e34c6e45ffcd35be8a6b7c50f329a74f35f426
```

Quando lo stato dei risultati di `DescribeDBClusters` è `available`, creare l'istanza primaria per il cluster di database affinché la replica possa iniziare. A tale scopo, utilizzate l'reateDBInstanceazione RDS API C come illustrato nell'esempio seguente.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBInstance
  &DBClusterIdentifier=sample-replica-cluster
  &DBInstanceClass=db.r5.large
  &DBInstanceIdentifier=sample-replica-instance
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
```

```
&X-Amz-Date=20160201T003808Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=125fe575959f5bbcebd53f2365f907179757a08b5d7a16a378dfa59387f58cdb
```

Quando l'istanza database viene creata e diventa disponibile, inizia la replica. È possibile determinare se l'istanza DB è disponibile chiamando il escribeDBInstances comando AWS CLI D.

Visualizzazione delle repliche di Amazon Aurora My SQL in più regioni

Puoi visualizzare le relazioni di replica tra regioni per i tuoi cluster Amazon Aurora SQL My DB chiamando describe-db-clustersAWS CLI il comando o l'operazione D. escribeDBClusters RDS API Nella risposta, fai riferimento al campo `ReadReplicaIdentifiers` per gli identificatori di cluster di database di cluster di qualsiasi cluster di database di replica di lettura tra regioni. Fai riferimento all'`ReplicationSourceIdentifier`elemento del cluster DB ARN di origine che è la fonte di replica.

Promozione di una replica di lettura su un cluster DB per Aurora My SQL

È possibile promuovere una replica Aurora My SQL read a un cluster DB autonomo. Quando promuovi una replica Aurora My SQL read, le relative istanze DB vengono riavviate prima che diventino disponibili.

In genere, si promuove una replica Aurora My SQL read su un cluster DB autonomo come schema di recupero dati in caso di errore del cluster DB di origine.

A tale scopo, crea prima di tutto una replica di lettura e quindi monitora il cluster di database di origine per individuare eventuali errori. In caso di errore, sono necessarie le operazioni seguenti:

1. Promuovi la replica di lettura.
2. Indirizzare il traffico di database al cluster database promosso.
3. Creare una replica di lettura sostitutiva con il cluster di database promosso come origine.

Quando promuovi una replica di lettura, questa diventa un cluster di database Aurora standalone. Il processo di promozione può richiedere alcuni minuti per il completamento, che possono aumentare a seconda delle dimensioni della replica di lettura. Dopo aver promosso la replica di lettura a nuovo cluster di database, il cluster si comporta come qualsiasi altro cluster di database. Ad esempio, è

possibile creare repliche di lettura da esso ed eseguire operazioni di ripristino. point-in-time È anche possibile creare repliche Aurora per il cluster database.

Poiché il cluster di database promosso non è più una replica di lettura, non è possibile usarlo come target di replica.

Le fasi seguenti illustrano il processo generale di promozione di una replica di lettura a cluster di database:

1. Arrestare la scrittura delle transazioni nel cluster di database di origine della replica di lettura e quindi attendere il completamento di tutti gli aggiornamenti nella replica di lettura. Gli aggiornamenti del database vengono eseguiti nella replica di lettura dopo essere stati completati nel cluster di database di origine e questo ritardo di replica può variare significativamente. Utilizzare il parametro `ReplicaLag` per determinare quando sono stati applicati tutti gli aggiornamenti alla replica di lettura. Il parametro `ReplicaLag` consente di registrare il tempo di ritardo di un'istanza database di replica di lettura rispetto all'istanza database di origine. Quando il parametro `ReplicaLag` diventa `0`, la replica di lettura ha raggiunto l'istanza database di origine.
2. Promuovi la replica di lettura utilizzando l'opzione Promote sulla RDS console Amazon, il AWS CLI comando promote-read-replica-db-cluster o l'operazione PromoteReadReplicaDBClusterAmazon RDSAPI.

   Scegli un'istanza Aurora My SQL DB per promuovere la replica di lettura. Dopo la promozione della replica di lettura, il cluster Aurora SQL My DB viene promosso a cluster DB autonomo. L'istanza database con la priorità di failover maggiore viene promossa a istanza database primaria per il cluster database. Le altre istanze database diventano repliche Aurora.

   > **Note**
   >
   > Per il completamento del processo di promozione sono necessari alcuni minuti. Quando promuovi una replica di lettura, la replica viene arrestata e le istanze database vengono riavviate. Al termine del riavvio, la replica di lettura è disponibile come un nuovo cluster di database.

Console

Per promuovere una replica Aurora My SQL read su un cluster DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nella console scegliere Instances (Istanze).

   Viene visualizzato il riquadro Instance (Istanza).
3. Nel riquadro Instances (Istanze), scegliere la replica di lettura che si vuole promuovere.

   Le repliche lette vengono visualizzate come istanze Aurora SQL My DB.
4. Per Actions (Operazioni), scegliere Promote read replica (Promuovere replica di lettura).
5. Nella pagina di conferma scegliere Promote read replica (Promuovi replica di lettura).

AWS CLI

Per promuovere una replica di lettura in un cluster DB, usa il comando -cluster. AWS CLI promote-read-replica-db

Example

PerLinux, omacOS: Unix

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier mydbcluster
```

Per Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier mydbcluster
```

RDS API

Per promuovere una replica di lettura in un cluster DB, chiama PromoteReadReplicaDBCluster.

Risoluzione dei problemi delle repliche interregionali per Amazon Aurora My SQL

Di seguito è riportato un elenco di messaggi di errore comuni che si possono verificare durante la creazione di una replica di lettura tra regioni Amazon Aurora e le istruzioni per risolvere gli errori specificati.

Il cluster di origine [cluster DB] non ha i binlog ARN abilitati

Per risolvere questo problema, abilitare l'accesso binario nel cluster database origine. Per ulteriori informazioni, consulta Prima di iniziare.

Il cluster di origine [cluster DBARN] non ha il gruppo di parametri del cluster sincronizzato su writer

Ricevi questo errore se hai aggiornato il parametro `binlog_format` del cluster database, ma non hai riavviato l'istanza primaria per il cluster database. Riavvia l'istanza primaria (ovvero, lo scrittore) per il cluster database e prova di nuovo.

Il cluster di origine [cluster DBARN] ha già una replica di lettura in questa regione

Per ogni cluster di database di origine in qualsiasi Regione AWS, puoi avere fino a cinque cluster di database tra regioni che sono repliche di lettura. Se disponi già del numero massimo di repliche di lettura per un cluster di database in una particolare Regione AWS, devi eliminarne una esistente prima di poter creare un nuovo cluster di database tra regioni in tale regione.

Il cluster DB [DB clusterARN] richiede un aggiornamento del motore di database per il supporto della replica tra regioni

Per risolvere questo problema, aggiorna la versione del motore di database per tutte le istanze nel cluster di database di origine alla versione di motore di database più recente e prova a creare di nuovo la replica di lettura tra regioni.

Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari)

Poiché Amazon Aurora My SQL è compatibile con MySQL, puoi configurare la replica tra un SQL database My e un cluster Amazon Aurora My DB. SQL Questo tipo di replica utilizza la replica del log SQL binario My, nota anche come replica binlog. Se utilizzi la replica dei log binari con Aurora, ti consigliamo che il tuo database SQL My esegua La SQL mia versione 5.5 o successiva. È possibile configurare la replica laddove il cluster Aurora SQL My DB è l'origine della replica o la replica. Puoi eseguire la replica con un'istanza Amazon RDS My SQL DB, un SQL database My esterno ad Amazon RDS o un altro cluster Aurora SQL My DB.

> **Note**
>
> Non è possibile utilizzare la replica binlog da o verso determinati tipi di cluster di database Aurora. In particolare, la replica binlog non è disponibile per Aurora Serverless v1 i cluster. Se l'istruzione `SHOW MASTER STATUS` and `SHOW SLAVE STATUS` (Aurora My SQL version 2) o (`SHOW REPLICA STATUS`Aurora My SQL version 3) non restituisce alcun output, verifica che il cluster che stai utilizzando supporti la replica binlog.

È inoltre possibile eseguire la replica con un'istanza RDS for My SQL DB o un cluster Aurora SQL My DB in un altro Regione AWS. Quando si esegue la replica in tutto il mondo Regioni AWS, assicurati che i cluster e le istanze DB siano accessibili al pubblico. Se i cluster Aurora My SQL DB si trovano in sottoreti private del tuoVPC, usa il peering tra VPC Regioni AWS. Per ulteriori informazioni, vedereUn cluster di DB in un'istanza a VPC cui accede un'EC2istanza in un'altra VPC.

Se si desidera configurare la replica tra un cluster Aurora SQL My DB e un cluster SQL Aurora My DB in un altro Regione AWS, puoi creare un cluster Aurora My SQL DB come replica di lettura in un altro Regione AWS dal cluster DB di origine. Per ulteriori informazioni, consulta Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS.

Con Aurora My SQL versione 2 e 3, è possibile eseguire la replica tra Aurora My SQL e un'origine o una destinazione esterna che utilizza gli identificatori di transazione globali () per la replica. GTIDs Assicurati che i parametri GTID relativi nel cluster Aurora SQL My DB abbiano impostazioni compatibili con lo stato GTID del database esterno. Per informazioni su come effettuare questa operazione, consulta Utilizzo della replica GTID basata. In Aurora My SQL versione 3.01 e successive, puoi scegliere come GTIDs assegnare transazioni replicate da una fonte che non utilizza. GTIDs Per informazioni sulla procedura archiviata che controlla tale impostazione, vedere mysql.rds_assign_gtids_to_anonymous_transactions (Aurora My versione 3) SQL.

> **Warning**
>
> Quando esegui la replica tra Aurora SQL My e SQL My, assicurati di utilizzare solo tabelle InnoDB. Se hai le mie ISAM tabelle che desideri replicare, puoi convertirle in InnoDB prima di impostare la replica con il seguente comando.
>
> ```
> alter table <schema>.<table_name> engine=innodb, algorithm=copy;
> ```

Nelle sezioni seguenti, configura la replica, interrompi la replica, ridimensiona le letture per il database, ottimizza la replica binlog e configura binlog avanzato.

Argomenti

- Configurazione della replica dei log binari per Aurora My SQL
- Interruzione della replica dei log binari per Aurora My SQL
- Ridimensionamento delle letture per il tuo SQL database My con Amazon Aurora
- Ottimizzazione della replica dei log binari per Aurora MySQL
- Configurazione di binlog avanzato per Aurora MySQL

Configurazione della replica dei log binari per Aurora My SQL

La configurazione di My SQL replication con Aurora SQL My prevede i seguenti passaggi, che vengono discussi in dettaglio:

Indice

- 1. Abilitare l'accesso binario nella fonte di replica
- 2. Mantieni i log binari nel master di replica fino a quando non sono più necessari
- 3. Crea una copia o un dump della fonte di replica
- 4. Carica il dump nella destinazione della replica (se necessario)
- 5. Creazione di un utente di replica sull'origine di replica
- 6. Abilitare la replica nel target di replica
  - Definire una posizione per arrestare la replica su una replica di lettura
- 7. Monitora la replica
- Sincronizzazione delle password tra origine di replica e destinazione

1. Abilitare l'accesso binario nella fonte di replica

Seguono le istruzioni su come abilitare l'accesso binario nella fonte di replica per il motore del database.

Motore del database	Istruzioni
Aurora Mia SQL	Per attivare la registrazione binaria su un cluster Aurora My DB SQL Imposta il parametro del cluster di database `binlog_format` su `ROW`, `STATEMENT` o `MIXED`. `MIXED` è consigliato a meno che tu abbia bisogno di un formato binlog specifico. Il valore predefinito è `OFF`. Per modificare il parametro `binlog_format` , crea un gruppo di parametri del cluster di database personalizzato e associalo al tuo cluster di database. Non puoi modificare i parametri in un gruppo di parametri del cluster di database predefinito. Se stai modificando il parametro `binlog_format` da `OFF` a un altro valore, riavvia il cluster di database Aurora per rendere effettiva la modifica. Per ulteriori informazioni, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora e .
RDSper My SQL	Per attivare la registrazione binaria su un'istanza Amazon RDS DB Non puoi attivare la registrazione binaria direttamente per un'istanza Amazon RDS DB, ma puoi attivarla effettuando una delle seguenti operazioni: • Abilitare i backup automatici per l'istanza database. È possibile abilitare i backup automatici quando si crea un'istanza database o si possono abilitare i backup modificando un'istanza database esistente. Per ulteriori informazioni, consulta la sezione Creazione di un'istanza DB nella Amazon RDS User Guide. • Crea una replica di lettura per l'istanza database. Per ulteriori informazioni, consulta Working with read replicas nella Amazon RDS User Guide.
Il mio SQL (esterno)	Per impostare la replica crittografata Per replicare i dati in modo sicuro con Aurora My SQL versione 2, puoi utilizzare la replica crittografata.

Motore del database	Istruzioni

> **Note**
>
> Se non hai bisogno di utilizzare la replica crittografata, puoi saltare queste fasi.

Seguono i prerequisiti per l'utilizzo della replica crittografata:

- Secure Sockets Layer (SSL) deve essere abilitato sul database My source esterno. SQL
- È necessario preparare una chiave client e un certificato client per il cluster Aurora My SQL DB.

Durante la replica crittografata, il cluster Aurora SQL My DB funge da client verso il server del database SQL My. I certificati e le chiavi per il SQL client Aurora My sono in file in formato.pem.

1. Assicurati di essere preparato per la replica crittografata:

   - Se non hai SSL abilitato il database My SQL source esterno e non hai preparato una chiave client e un certificato client, SSL accendi il server My SQL database e genera la chiave client e il certificato client richiesti.
   - Se SSL è abilitato sulla fonte esterna, fornisci una chiave client e un certificato per il cluster Aurora My SQL DB. Se non li possiedi, genera una nuova chiave e un nuovo certificato per il cluster Aurora My SQL DB. Per firmare il certificato client, devi disporre della chiave di autorità di certificazione che hai usato per configurare SSL sul database esterno My SQL source.

   Per ulteriori informazioni, consulta Creazione di SSL certificati e chiavi utilizzando openssl nella documentazione personaleSQL.

   Hai bisogno del certificato autorità certificato, della chiave client e del certificato client.

Motore del database	Istruzioni

2. Connect al cluster Aurora My SQL DB come utente principale utilizzando. SSL

   Per informazioni sulla connessione a un cluster Aurora My SQL DB conSSL, vedere. Connessioni TLS ai cluster Aurora MySQL DB

3. Esegui la `mysql.rds_import_binlog_ssl_material` stored procedure per importare le SSL informazioni nel cluster Aurora My SQL DB.

   Per il `ssl_material_value` parametro, inserisci le informazioni dai file in formato.pem per il cluster Aurora My SQL DB nel payload corretto. JSON

   L'esempio seguente importa SSL informazioni in un cluster Aurora My SQL DB. Nei file in formato .pem, il codice del corpo è in genere più lungo del codice del corpo riportato nell'esempio.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICA
TE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE
 KEY-----
```

Motore del database	Istruzioni

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pc
jqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSF
JuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJ
tjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMu
cxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Per ulteriori informazioni, consulta mysql.rds_import_binlog_ssl_material e Connessioni TLS ai cluster Aurora MySQL DB.

> **Note**
>
> Dopo aver eseguito la procedura, i segreti vengono archiviati in file. Per eliminare i file in un secondo momento, puoi eseguire la stored procedure mysql.rds_remove_binlog_ssl_material.

Per attivare la registrazione binaria su un database My esterno SQL

1. Da un shell comando, interrompi il servizio mysql.

   ```
   sudo service mysqld stop
   ```

2. Modificare il file `my.cnf` (posto in genere sotto `/etc`).

   ```
   sudo vi /etc/my.cnf
   ```

   Aggiungere le opzioni `log_bin` e `server_id` alla sezione `[mysqld]`. L'opzione `log_bin` fornisce un identificatore di nome file per i file di log binari. L'opzione `server_id` fornisce un identificatore univoco per il server in relazioni master-replica.

Motore del database	Istruzioni

	Se la replica crittografata non è richiesta, assicuratevi che il SQL database My esterno sia avviato con i binlogs abilitati e SSL che sia disattivato.
	
	Seguono le voci rilevanti nel file `/etc/my.cnf` per dati non crittografati.
	
	`log-bin=mysql-bin`
	`server-id=2133421`
	`innodb_flush_log_at_trx_commit=1`
	`sync_binlog=1`
	
	Se è necessaria la replica crittografata, assicuratevi che il SQL database My esterno sia avviato con SSL e che i binlogs siano abilitati.
	
	Le voci del `/etc/my.cnf` file includono le posizioni dei file.pem per il server di database My. SQL
	
	`log-bin=mysql-bin`
	`server-id=2133421`
	`innodb_flush_log_at_trx_commit=1`
	`sync_binlog=1`
	
	`# Setup SSL.`
	`ssl-ca=/home/sslcerts/ca.pem`
	`ssl-cert=/home/sslcerts/server-cert.pem`
	`ssl-key=/home/sslcerts/server-key.pem`
	
	Inoltre, l'`sql_mode`opzione per l'istanza My SQL DB deve essere impostata su 0 o non deve essere inclusa nel file my.cnf.
	
	Mentre sei connesso al SQL database My esterno, registra la posizione del log binario del SQL database My esterno.
	
	`mysql> SHOW MASTER STATUS;`
	
	L'output visualizzato dovrebbe essere simile al seguente:

Motore del database	Istruzioni

```
+------------------+----------+--------------+------------------
+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
 Executed_Gtid_Set |
+------------------+----------+--------------+------------------
+-------------------+
| mysql-bin.000031 |      107 |              |                  |
              |
+------------------+----------+--------------+------------------
+-------------------+
1 row in set (0.00 sec)
```

Per ulteriori informazioni, consulta Setting the replication source configurationnella sezione La mia SQL documentazione.

3. Avvia il servizio mysql.

```
sudo service mysqld start
```

2. Mantieni i log binari nel master di replica fino a quando non sono più necessari

Quando usi My SQL binary log replication, Amazon RDS non gestisce il processo di replica. Come risultato, devi assicurarti che i file binlog nel master di replica siano mantenuti finché le modifiche vengono applicate alla replica. Questa manutenzione aiuta a ripristinare il database di origine in caso di errore.

Utilizza le istruzioni seguenti per mantenere i registri binari per il motore di database.

Motore del database	Istruzioni
Aurora Mia SQL	Per conservare i log binari su un cluster Aurora My DB SQL

Motore del database	Istruzioni
	Non hai accesso ai file binlog per un cluster Aurora SQL My DB. Di conseguenza, è necessario scegliere un periodo di tempo per conservare i file binlog sulla fonte di replica abbastanza a lungo da garantire che le modifiche siano state applicate alla replica prima che il file binlog venga eliminato da Amazon. RDS È possibile conservar e i file binlog su un cluster Aurora SQL My DB per un massimo di 90 giorni. Se stai configurando la replica con un'istanza My SQL database o My SQL DB come replica e il database RDS per cui stai creando una replica è molto grande, scegli un intervallo di tempo ampio per conservare i file binlog fino al completamento della copia iniziale del database sulla replica e al raggiungimento del ritardo di replica pari a 0. Per impostare il periodo di conservazione dei log binari, usa la procedura mysql.rds _set_configuration e specifica il parametro di configurazione `'binlog retention hours'` insieme al numero di ore di conservazione dei file binlog nel cluster di database. Il valore massimo per Aurora My SQL versione 2.11.0 e successive e versione 3 è 2160 (90 giorni). L'esempio seguente imposta il periodo di conservazione dei file binlog su 6 giorni: `CALL mysql.rds_set_configuration('binlog retention hours', 144);` Dopo l'avvio della replica, è possibile verificare che le modifiche siano state applicate alla replica eseguendo il comando (`SHOW SLAVE STATUS`Aurora My SQL versione 2) o (`SHOW REPLICA STATUS`Aurora My SQL versione 3) sulla replica e controlla ndo il campo. `Seconds behind master` Se il campo `Seconds behind master` è 0, non vi è alcun ritardo di replica. Quando non c'é ritardo di replica, riduci il periodo di tempo di conservazione dei file binlog impostando il parametro di configurazione `binlog retention hours` su un intervallo di tempo più breve. Se questa impostazione non è specificata, l'impostazione predefinita per Aurora My SQL è 24 (1 giorno).

Motore del database	Istruzioni
	Se si specifica un valore `'binlog retention hours'` superiore al valore massimo, Aurora My SQL utilizza il valore massimo.
RDSper My SQL	Per conservare i log binari su un'istanza Amazon RDS DB Puoi conservare i file di log binari su un'istanza Amazon RDS DB impostando le ore di conservazione dei binlog proprio come con un cluster Aurora SQL My DB, descritto nella riga precedente. Puoi anche conservare i file binlog su un'istanza Amazon RDS DB creando una replica di lettura per l'istanza DB. La replica di lettura è temporanea e ha solamente l'obiettivo di mantenere i file binlog. Dopo che la replica di lettura è stata creata, chiama la procedura mysql.rds_stop_replication nella replica di lettura. Mentre la replica viene interrotta, Amazon RDS non elimina nessuno dei file binlog sull'origine della replica. Dopo aver impostato la replica con la replica permanente, puoi eliminare la replica di lettura quando il ritardo di replica (campo `Seconds behind master`) tra il master di replica a la replica permanente diventa 0.
Il mio SQL (esterno)	Per conservare i log binari su un database My SQL esterno Poiché i file binlog su un SQL database My esterno non sono gestiti da AmazonRDS, vengono conservati fino a quando non vengono eliminati. Dopo l'avvio della replica, è possibile verificare che le modifiche siano state applicate alla replica eseguendo il comando (`SHOW SLAVE STATUS`Aurora My SQL versione 2) o (`SHOW REPLICA STATUS`Aurora My SQL versione 3) sulla replica e controllando il campo. `Seconds behind master` Se il campo `Seconds behind master` è 0, non vi è alcun ritardo di replica. Quando non c'é ritardo di replica, puoi eliminare i vecchi file binlog.

3. Crea una copia o un dump della fonte di replica

Si utilizza un'istantanea, un clone o un dump della fonte di replica per caricare una copia di base dei dati sulla replica. Quindi si inizia a replicare da quel punto.

Utilizza le seguenti istruzioni per creare una copia o un dump della fonte di replica per il tuo motore di database.

Motore del database	Istruzioni
Aurora Mia SQL	Per creare una copia di un cluster Aurora My DB SQL Seleziona uno dei seguenti metodi: • Ripristina da un'istantanea del cluster DB: 1. Crea una snapshot di un cluster di database del cluster di database Amazon Aurora. Per ulteriori informazioni, consulta Creazione di uno snapshot del cluster database. 2. Crea un nuovo cluster di database Aurora ripristinandolo dalla snapshot cluster di database che hai appena creato. Assicurati di mantenere il gruppo di parametri database per il cluster di database ripristinato come con il cluster di database originale. Ciò assicura che la copia del cluster di database abbia l'accesso binario abilitato. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database. • Clona il tuo cluster DB. Per ulteriori informazioni, consulta Clonazione di un volume per un cluster di database Amazon Aurora. Per determinare il nome e la posizione del file binlog Seleziona uno dei seguenti metodi: • Nel: AWS Management Console 1. Scegli Databases, quindi scegli l'istanza principale (writer) per il tuo nuovo cluster Aurora DB per mostrarne i dettagli. 2. Scorri fino a Recent Events (Eventi recenti). Viene visualizzato un messaggio di evento che include il nome e la posizione del file binlog. Il messaggio evento è nel formato seguente.

Motore del database	Istruzioni

```
Binlog position from crash recovery is binlog-file-name  binlog-po
sition
```

3. Salva i valori del nome e della posizione del file binlog per quando avvierai la replica.

- Chiamate il AWS CLI comando describe-events, come nell'esempio seguente.

```
aws rds describe-events

{
    "Events": [
        {
            "EventCategories": [],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:
db:sample-restored-instance",
            "Date": "2016-10-28T19:43:46.862Z",
            "Message": "Binlog position from crash recovery is mysql-
bin-changelog.000003 4278",
            "SourceIdentifier": "sample-restored-instance"
        }
    ]
}
```

- Controlla il registro degli SQL errori My per l'ultima posizione del file My SQL binlog.

Per creare un dump di un cluster Aurora My DB SQL

Se la destinazione della replica è un SQL database My esterno o un'istanza RDS for My SQL DB, è necessario creare un file di dump dal cluster Aurora DB.

Assicurati di eseguire il `mysqldump` comando sulla copia del cluster DB di origine che hai creato. Questo serve a evitare considerazioni bloccanti quando si esegue il dump. Se il dump fosse eseguito direttamente sul cluster DB di origine, sarebbe necessario bloccare le tabelle di origine per evitare scritture simultanee su di esse mentre il dump è in corso.

Motore del database	Istruzioni
	1. Connect al cluster DB utilizzando un SQL client My. 2. Emetti il `mysqldump` comando. Per esempio: `PROMPT> mysqldump --databases database_name --single-transaction --order-by-primary -r backup.sql -u local_user s -p` 3. Dopo aver creato il file di dump, è possibile eliminare la copia del cluster DB.
RDSper My SQL	Per creare uno snapshot di un'istanza Amazon RDS DB Crea una replica di lettura della tua istanza Amazon RDS DB. Per ulteriori informazi oni, consulta Creazione di una replica di lettura nella Guida per l'utente di Amazon Relational Database Service. 1. Esegui la connessione alla replica di lettura e interrompi la replica eseguendo la procedura mysql.rds_stop_replication. 2. Mentre la replica di lettura è interrotta, connettiti alla replica di lettura ed `SHOW SLAVE STATUS` esegui il comando (Aurora My SQL versione 2) o (`SHOW REPLICA STATUS`Aurora My versione 3). SQL Recupera il nome dei file di log binario corrente dal campo `Relay_Master_Log_File` e la posizione del file di log dal campo `Exec_Master_Log_Pos` . Salva questi valori per quando avvierai la replica. 3. Mentre la replica di lettura rimane nello stato Stopped (Interrotto), crea una snapshot DB di una replica di lettura. Per ulteriori informazioni, consulta Creazione di uno snapshot DB nella Guida per l'utente di Amazon Relational Database Service. 4. Elimina la replica di lettura.

Motore del database	Istruzioni

My (esterno) SQL	Per creare un dump di un database My SQL esterno
	1. Prima di creare un dump, devi assicurarti che la posizione del binlog per il dump sia aggiornata con i dati nell'istanza di origine. Per fare ciò, devi prima fermare qualsiasi operazione di scrittura all'istanza con il seguente comando:
	`mysql> FLUSH TABLES WITH READ LOCK;`
	2. Crea un dump del tuo SQL database My utilizzando il `mysqldump` comando illustrato di seguito:
	`PROMPT> sudo mysqldump --databases database_name --master-data=2 --single-transaction \ --order-by-primary -r backup.sql -u local_user -p`
	3. Dopo aver creato il dump, sblocca le tabelle nel tuo SQL database My con il seguente comando:
	`mysql> UNLOCK TABLES;`

4. Carica il dump nella destinazione della replica (se necessario)

Se prevedi di caricare dati da un dump di un SQL database My esterno ad AmazonRDS, potresti voler creare un'EC2istanza in cui copiare i file di dump. Quindi puoi caricare i dati nel tuo cluster DB o nell'istanza DB da quell'EC2istanza. Utilizzando questo approccio, puoi comprimere i file di dump prima di copiarli sull'EC2istanza per ridurre i costi di rete associati alla copia dei dati su Amazon. RDS Puoi anche crittografare il file o i file dump per assicurare i dati mentre vengono trasferiti nella rete.

> **Note**
>
> Se crei un nuovo cluster Aurora My SQL DB come destinazione della replica, non è necessario caricare un file di dump:

- È possibile eseguire il ripristino da uno snapshot del cluster DB per creare un nuovo cluster DB. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database.
- È possibile clonare il cluster DB di origine per creare un nuovo cluster DB. Per ulteriori informazioni, consulta Clonazione di un volume per un cluster di database Amazon Aurora.
- È possibile migrare i dati da uno snapshot di un'istanza DB in un nuovo cluster DB. Per ulteriori informazioni, consulta Migrazione dei dati verso un cluster Amazon Aurora My DB SQL.

Utilizza le seguenti istruzioni per caricare il dump della fonte di replica nella destinazione di replica per il motore di database.

Motore del database	Istruzioni

Aurora Mia SQL	Per caricare un dump in un cluster Aurora My DB SQL
	1. Copia l'output del `mysqldump` comando dalla fonte di replica in una posizione che può anche connettersi al cluster Aurora SQL My DB.
	2. Connect al cluster Aurora My SQL DB utilizzando il `mysql` comando. Di seguito è riportato un esempio.
	`PROMPT> mysql -h host_name -port=3306 -u db_master_user -p`
	3. Al `mysql` prompt, esegui il `source` comando e passagli il nome del file di dump del database per caricare i dati nel cluster Aurora My SQL DB, ad esempio:
	`mysql> source backup.sql;`

RDSper My SQL	Per caricare un dump in un'istanza Amazon RDS DB 1. Copia l'output del `mysqldump` comando dalla fonte di replica in una posizione che possa connettersi anche alla tua istanza My SQL DB. 2. Connect alla propria istanza My SQL DB utilizzando il `mysql` comando. Di seguito è riportato un esempio.

Motore del database	Istruzioni
	``` PROMPT> mysql -h host_name -port=3306 -u db_master_user -p ``` 3. Al `mysql` prompt, esegui il `source` comando e passagli il nome del file di dump del database per caricare i dati nell'istanza My SQL DB, ad esempio: ``` mysql> source backup.sql; ```
My SQL (esterno)	Per caricare un dump in un database My SQL esterno Non è possibile caricare uno snapshot DB o uno snapshot del cluster DB in un database My esterno. SQL Devi utilizzare invece l'output dal comando `mysqldump` . 1. Copia l'output del `mysqldump` comando dalla tua fonte di replica in una posizione che può anche connettersi al tuo database My. SQL 2. Connect al SQL database My utilizzando il `mysql` comando. Di seguito è riportato un esempio. ``` PROMPT> mysql -h host_name -port=3306 -u db_master_user -p ``` 3. Al `mysql` prompt, esegui il `source` comando e passagli il nome del file di dump del database per caricare i dati nel tuo database MySQL. Di seguito è riportato un esempio. ``` mysql> source backup.sql; ```

5. Creazione di un utente di replica sull'origine di replica

Crea un ID utente sull'origine utilizzato solamente per la replica. L'esempio seguente è RDS per database My SQL o My SQL source esterni.

```
mysql> CREATE USER 'repl_user'@'domain_name' IDENTIFIED BY 'password';
```

Per i database Aurora My SQL source, il parametro del cluster `skip_name_resolve` DB è impostato su 1 (0N) e non può essere modificato, quindi è necessario utilizzare un indirizzo IP per l'host anziché un nome di dominio. Per ulteriori informazioni, consulta skip_name_resolve nella documentazione My. SQL

```
mysql> CREATE USER 'repl_user'@'IP_address' IDENTIFIED BY 'password';
```

L'utente richiede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE`. Concedi questi privilegi all'utente.

Se è necessario utilizzare la replica crittografata, è necessario disporre di SSL connessioni per l'utente di replica. Ad esempio, è possibile utilizzare una delle seguenti istruzioni per richiedere SSL connessioni sull'account utente. `repl_user`

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'IP_address';
```

```
GRANT USAGE ON *.* TO 'repl_user'@'IP_address' REQUIRE SSL;
```

> **Note**
>
> Se `REQUIRE SSL` non è incluso, la connessione di replica potrebbe ridiventare una connessione non crittografata.

6. Abilitare la replica nel target di replica

Prima di attivare la replica, si consiglia di scattare un'istantanea manuale del cluster Aurora My SQL DB o della destinazione di replica dell'RDSistanza My SQL DB. Se c'è un problema e devi ristabilire la replica con il cluster di database o il target di replica dell'istanza database, puoi ripristinare il cluster di database o l'istanza database da questa snapshot invece di dover importare di nuovo i dati nel target di replica.

Utilizza le istruzioni seguenti per attivare la replica del motore di database.

Motore del database	Istruzioni
Aurora Mia SQL	Per attivare la replica da un cluster Aurora My DB SQL 1. Individua il punto di partenza per la replica. Hai bisogno del nome e della posizione del file binlog. Se la destinazione della replica del cluster di database è stata creata da: • Istantanea o clone del cluster DB: recupera il nome e la posizione del file binlog dagli eventi recenti per il nuovo cluster DB, come mostrato in. 3. Crea una copia o un dump della fonte di replica • Istantanea DB: hai recuperato il nome e la posizione del file binlog dal comando (`SHOW SLAVE STATUS`Aurora My SQL versione 2) o (`SHOW REPLICA STATUS`Aurora My SQL versione 3) quando hai creato l'istantanea della tua origine di replica. 2. Connettiti al cluster di database e richiama le seguenti procedure per avviare la replica con l'origine utilizzando il nome e il percorso del file di log binario del passaggio precedente: • mysql.rds_set_external_source () SQL • mysql.rds_set_external_master SQL • mysql.rds_start_replication (tutte le versioni) L'esempio seguente è per Aurora My SQL versione 3. `CALL mysql.rds_set_external_source ('mydbinstance.123456789012 .us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Per utilizzare SSL la crittografia, imposta il valore finale su 1 invece di`0`.
RDSper My SQL	Per attivare la replica da un'istanza Amazon RDS DB

Motore del database	Istruzioni
	1. Se il target di replica dell'istanza database è stato creato da una snapshot DB, hai bisogno del file e della posizione binlog che sono il punto di partenza per la replica. Hai recuperato questi valori dal comando `SHOW SLAVE STATUS` (Aurora SQL My versione 2) o `SHOW REPLICA STATUS` (Aurora SQL My versione 3) quando hai creato l'istantanea della tua origine di replica. 2. Esegui la connessione all'istanza database e chiama le procedure mysql.rds_set_external_master SQL o mysql.rds_set_external_source () SQL e mysql.rds_start_replication per avviare la replica con l'origine. Utilizza il nome e il percorso del file di log binario del passaggio precedente. Di seguito è riportato un esempio. `CALL mysql.rds_set_external_master ('mydbcluster.cluster-12345 6789012.us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Per utilizzare la SSL crittografia, imposta il valore finale su invece di. `1` `0`

Motore del database	Istruzioni

Il mio SQL (esterno)	Per attivare la replica da un database My SQL esterno
	
	1. Recupera il file e la posizione binlog che sono il punto di inizio per la replica. Hai recuperato questi valori dal comando `SHOW SLAVE STATUS` (Aurora SQL My versione 2) o `SHOW REPLICA STATUS` (Aurora SQL My versione 3) quando hai creato l'istantanea della tua origine di replica. Se la destinazione esterna My SQL replica è stata compilata dall'output del `mysqldump` comando con l'`--master-data=2` opzione, il file binlog e la posizione binlog vengono inclusi nell'output. Di seguito è riportato un esempio.
	
	``
	--
	-- Position to start replication or point-in-time recovery from
	--
	
	-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
	 MASTER_LOG_POS=107;
	``
	
	2. Connettiti alla destinazione esterna My SQL replica ed emetti `CHANGE MASTER TO` and `START SLAVE` (Aurora SQL My versione 2) o `START REPLICA` (Aurora SQL My versione 3) per avviare la replica con l'origine di replica utilizzando il nome e la posizione del file di log binario del passaggio precedente, ad esempio:
	
	``
	CHANGE MASTER TO
	  MASTER_HOST = 'mydbcluster.cluster-123456789012.us-east-1.r
	ds.amazonaws.com',
	  MASTER_PORT = 3306,
	  MASTER_USER = 'repl_user',
	  MASTER_PASSWORD = 'password',
	  MASTER_LOG_FILE = 'mysql-bin-changelog.000031',
	  MASTER_LOG_POS = 107;
	-- And one of these statements depending on your engine version:
	START SLAVE; -- Aurora MySQL version 2
	START REPLICA; -- Aurora MySQL version 3
	``

Se la replica ha esito negativo, può verificarsi un notevole aumento di I/O non intenzionale sulla replica, che può compromettere le prestazioni. Se la replica non riesce o non è più necessaria, è possibile eseguire la stored procedure mysql.rds_reset_external_master Aurora My versione 2) SQL SQL o mysql.rds_reset_external_source SQL per rimuovere la configurazione della replica.

Definire una posizione per arrestare la replica su una replica di lettura

In Aurora My SQL versione 3.04 e successive, è possibile avviare la replica e quindi interromperla in una posizione specificata del file di registro binario utilizzando la stored procedure. mysql.rds_start_replication_until (Aurora My versione 3) SQL

Per avviare la replica su una replica di lettura e arrestare la replica in corrispondenza di una posizione specifica

1. Utilizzando un SQL client My, connettiti al cluster di replica Aurora SQL My DB come utente principale.
2. Eseguire la procedura archiviata mysql.rds_start_replication_until (Aurora My versione 3) SQL.

   L'esempio seguente avvia la replica e replica le modifiche fino a raggiungere la posizione `120` nel file di log binario `mysql-bin-changelog.000777`. In caso di disaster recovery, presumere che la posizione `120` si riferisca al momento immediatamente precedente l'errore.

   ```
   call mysql.rds_start_replication_until(
     'mysql-bin-changelog.000777',
     120);
   ```

La replica si arresta automaticamente quando viene raggiunto il punto di arresto. Viene generato il seguente RDS evento:. `Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure`

Se si utilizza la replica GTID basata, utilizzare la mysql.rds_start_replication_until_gtid (Aurora La mia versione 3) SQL stored procedure anziché la mysql.rds_start_replication_until (Aurora My versione 3) SQL stored procedure. Per ulteriori informazioni sulla replica GTID basata, vedere. Utilizzo della replica GTID basata

7. Monitora la replica

Quando si configura La mia SQL replica con un cluster Aurora SQL My DB, è necessario monitorare gli eventi di failover per il cluster Aurora SQL My DB quando è la destinazione della replica. Se accade un failover, il cluster di database che è il target di replica potrebbe essere ricreato in un nuovo host con un indirizzo di rete diverso. Per informazioni su come monitorare gli eventi di failover, consulta Utilizzo delle notifiche di RDS eventi di Amazon.

È inoltre possibile monitorare la distanza tra la destinazione della replica e l'origine della replica collegandosi alla destinazione della replica ed eseguendo il comando (`SHOW SLAVE STATUS`Aurora My SQL versione 2) o (`SHOW REPLICA STATUS`Aurora My versione 3). SQL Nell'output del comando, il campo `Seconds Behind Master` indica quanto è indietro il target di replica rispetto al master di replica.

> **Important**
>
> Se aggiorni il cluster DB e specifichi un gruppo di parametri personalizzato, assicurati di riavviare manualmente il cluster al termine dell'aggiornamento. In questo modo il cluster utilizza le nuove impostazioni personalizzate dei parametri e riavvia la replica binlog.

Sincronizzazione delle password tra origine di replica e destinazione

Quando si modificano gli account utente e le password sull'origine di replica utilizzando le SQL istruzioni, tali modifiche vengono replicate automaticamente nel target di replica.

Se si utilizza il AWS Management Console, il o il AWS CLI RDS API per modificare la password principale sull'origine della replica, tali modifiche non vengono replicate automaticamente nella destinazione di replica. Se si desidera sincronizzare l'utente principale e la password principale tra il sistema di origine e quello di destinazione, è necessario apportare la stessa modifica alla destinazione di replica.

Interruzione della replica dei log binari per Aurora My SQL

Per interrompere la replica dei log binari con un'istanza My SQL DB, un SQL database My esterno o un altro cluster Aurora DB, segui questi passaggi, descritti in dettaglio più avanti in questo argomento.

1. Arrestare la replica dei log binari nella destinazione di replica

2. Disabilitare l'accesso binario alla fonte di replica

1. Arrestare la replica dei log binari nella destinazione di replica

Usa le seguenti istruzioni per arrestare la replica dei log binari del motore di database.

Motore del database	Istruzioni
Aurora Mia SQL	Per interrompere la replica dei log binari su una destinazione di replica del cluster Aurora SQL My DB Esegui la connessione al cluster di database Aurora che è la destinazione della replica e chiama la procedura mysql.rds_stop_replication.
RDSper My SQL	Per interrompere la replica dei log binari su un'istanza Amazon RDS DB Connect all'istanza RDS DB che è la destinazione della replica e richiama la mysql.rds_stop_replication procedura.
Il mio SQL (esterno)	Per interrompere la replica dei log binari su un database My SQL esterno Connect al SQL database My ed esegui il comando `STOP SLAVE` (versione 5.7) o `STOP REPLICA` (versione 8.0).

2. Disabilitare l'accesso binario alla fonte di replica

Usa le seguenti istruzioni per disattivare la registrazione dei log binari nell'origine della replica per il motore del database.

Motore del database	Istruzioni
Aurora Mia SQL	Per disabilitare l'accesso binario in un cluster di database Amazon Aurora 1. Connettiti al cluster di database Aurora che è l'origine della replica. 2. Utilizza la procedura mysql.rds_set_configuration e specifica il parametro di configurazione `binlog retention hours`, con il valore `NULL`, come illustrato nell'esempio seguente.

Motore del database	Istruzioni
	`CALL mysql.rds_set_configuration('binlog retention hours', NULL);` **Note** Non puoi utilizzare il valore `0` per `binlog retention hours`. 3. Imposta il parametro `binlog_format` su `OFF` nel master di replica. Il parametro `binlog_format` è nel gruppo di parametri di database del cluster di database personalizzato associato al cluster di database. Dopo aver modificato il valore del parametro `binlog_format` , riavvia il cluster di database per rendere effettiva la modifica. Per ulteriori informazioni, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora e Modifica dei parametri in un gruppo di parametri DB in .
RDSper My SQL	Per disattivare la registrazione binaria su un'istanza Amazon RDS DB Non puoi disattivare la registrazione binaria direttamente per un'istanza Amazon RDS DB, ma puoi disattivarla effettuando le seguenti operazioni: 1. Disabilitare i backup automatici per l'istanza database. Puoi disabilitare i backup automatici modificando un'istanza database esistente e impostando Backup Retention Period (Tempo di conservazione del backup) su 0. Per ulteriori informazioni, consulta Modificare un'istanza Amazon RDS DB e Lavorare con i backup nella Amazon Relational Database Service User Guide. 2. Elimina tutte le repliche di lettura per l'istanza database. Per ulteriori informazioni, consulta Lavorare con le repliche di lettura delle istanze DB MariaDB, SQL My e SQL Postgre nella Amazon Relational Database Service User Guide.

Motore del database	Istruzioni

My (esterno) SQL	Per disattivare la registrazione binaria su un database My SQL esterno
	
	Connect al SQL database My e richiama il `STOP REPLICATION` comando.
	
	1. Da una shell dei comandi, arrestare il servizio `mysqld`,
	
	`sudo service mysqld stop`
	
	2. Modificare il file `my.cnf` (posto in genere sotto `/etc`).
	
	`sudo vi /etc/my.cnf`
	
	Elimina le opzioni `log_bin` e `server_id` dalla sezione `[mysqld]`.
	
	Per ulteriori informazioni, consulta Setting the replication source configurationnella sezione La mia SQL documentazione.
	
	3. Avvia il servizio mysql.
	
	`sudo service mysqld start`

Ridimensionamento delle letture per il tuo SQL database My con Amazon Aurora

Puoi usare Amazon Aurora con la tua istanza My SQL DB per sfruttare le funzionalità di scalabilità di lettura di Amazon Aurora ed espandere il carico di lavoro di lettura per la tua istanza My DB. SQL Per usare Aurora per ridimensionare le letture per la tua istanza My SQL DB, crea un cluster Amazon Aurora My SQL DB e rendilo una replica di lettura della tua istanza My DB. SQL Questo vale per un'istanza RDS for My SQL DB o un SQL database My in esecuzione all'esterno di AmazonRDS.

Per informazioni su come creare un cluster di database Amazon Aurora, consulta Creazione di un cluster database Amazon Aurora.

Quando configuri la replica tra l'istanza My SQL DB e il cluster DB Amazon Aurora, assicurati di seguire queste linee guida:

- Usa l'indirizzo dell'endpoint del cluster Amazon Aurora DB quando fai riferimento al tuo cluster Amazon Aurora My DB. SQL Se si verifica un failover, la replica Aurora promossa a istanza principale per il cluster Aurora SQL My DB continua a utilizzare l'indirizzo dell'endpoint del cluster DB.
- Conservare i binlog sull'istanza di scrittura finché non si ha la conferma che siano stati applicati alla replica di Aurora. Questa manutenzione assicura il ripristino dell'istanza di lettura nel caso di errori.

> **Important**
>
> Quando si utilizza la replica auto-gestita, si ha la responsabilità di monitorare e risolvere eventuali problemi di replica. Per ulteriori informazioni, consulta Diagnosi e risoluzione del ritardo tra repliche di lettura.

> **Note**
>
> Le autorizzazioni necessarie per avviare la replica su un cluster Aurora My SQL DB sono limitate e non disponibili per il tuo utente master Amazon. RDS Pertanto, è necessario utilizzare le mysql.rds_start_replication procedure mysql.rds_set_external_master SQL or mysql.rds_set_external_source () SQL and per configurare la replica tra il cluster Aurora SQL My DB e l'istanza SQL My DB.

Avvia la replica tra un'istanza di origine esterna e un cluster Aurora My DB SQL

1. Rendi di sola lettura l'istanza My SQL DB di origine:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Esegui il `SHOW MASTER STATUS` comando sull'istanza My SQL DB di origine per determinare la posizione del binlog. Viene restituito un output simile all'esempio seguente:

```
File                      Position
-----------------------------------
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. Copia il database dall'istanza My SQL DB esterna al cluster Amazon Aurora My SQL DB utilizzando. `mysqldump` Per database molto grandi, potresti voler utilizzare la procedura descritta in Importazione di dati in un'istanza DB My SQL o MariaDB con tempi di inattività ridotti nella Amazon Relational Database Service User Guide.

   Per, o: Linux macOS Unix

   ```
   mysqldump \
       --databases <database_name> \
       --single-transaction \
       --compress \
       --order-by-primary \
       -u local_user \
       -p local_password | mysql \
           --host aurora_cluster_endpoint_address \
           --port 3306 \
           -u RDS_user_name \
           -p RDS_password
   ```

   Per Windows:

   ```
   mysqldump ^
       --databases <database_name> ^
       --single-transaction ^
       --compress ^
       --order-by-primary ^
       -u local_user ^
       -p local_password | mysql ^
           --host aurora_cluster_endpoint_address ^
           --port 3306 ^
           -u RDS_user_name ^
           -p RDS_password
   ```

   > **Note**
   >
   > Assicurati che non siano presenti spazi tra l'opzione `-p` e la password immessa.

   Utilizza le opzioni `--host`, `--user (-u)`, `--port` e `-p` nel comando `mysql` per specificare il nome host, il nome utente e la password per eseguire la connessione al cluster di database

Aurora. Il nome host è il DNS nome dell'endpoint del cluster Amazon Aurora DB, ad esempio. `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com` Puoi trovare il valore dell'endpoint nei dettagli del cluster nella Console di RDS gestione Amazon.

4. Rendi nuovamente scrivibile l'istanza My SQL DB di origine:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

Per ulteriori informazioni sulla creazione di backup da utilizzare con la replica, consulta la documentazione Backing up a source or replica by making it read onlypersonale. SQL

5. Nella Console di RDS gestione Amazon, aggiungi l'indirizzo IP del server che ospita il SQL database My di origine al gruppo VPC di sicurezza per il cluster Amazon Aurora DB. Per ulteriori informazioni sulla modifica di un gruppo VPC di sicurezza, consulta Security groups for your VPC nella Amazon Virtual Private Cloud User Guide.

Potrebbe anche essere necessario configurare la rete locale per consentire le connessioni dall'indirizzo IP del cluster Amazon Aurora DB, in modo che possa comunicare con la tua istanza My SQL di origine. Per individuare l'indirizzo IP del cluster di database Amazon Aurora, utilizzare il comando `host`.

```
host aurora_endpoint_address
```

Il nome host è il DNS nome dell'endpoint del cluster Amazon Aurora DB.

6. Utilizzando il client di tua scelta, connettiti all'SQListanza My esterna e crea un SQL utente My da utilizzare per la replica. Questo account viene utilizzato unicamente per la replica e deve essere limitato al dominio personale per aumentare la sicurezza. Di seguito è riportato un esempio.

```
CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
```

7. Per l'SQListanza My esterna, concedi `REPLICATION CLIENT` e `REPLICATION SLAVE` privilegi al tuo utente di replica. Per concedere ad esempio i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente "`repl_user`" del proprio dominio, eseguire questo comando.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com'
    IDENTIFIED BY 'password';
```

8. Scatta un'istantanea manuale del cluster Aurora SQL My DB come replica di lettura prima di configurare la replica. Se è necessario ristabilire la replica con il cluster DB come replica di lettura, è possibile ripristinare il cluster Aurora My SQL DB da questa istantanea invece di dover importare i dati dall'SQListanza My DB in un nuovo cluster Aurora My DB. SQL
9. Configurare il cluster di database Amazon Aurora come replica. Connettiti al cluster Amazon Aurora DB come utente principale e identifica il SQL database My di origine come origine di replica utilizzando le procedure mysql.rds_set_external_master SQL or mysql.rds_set_external_source () SQL and. mysql.rds_start_replication

   Usa il nome e la posizione del file binlog che hai determinato nel passaggio 2. Di seguito è riportato un esempio.

   ```
   For Aurora MySQL version 2:
   CALL mysql.rds_set_external_master ('mymasterserver.example.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);

   For Aurora MySQL version 3:
   CALL mysql.rds_set_external_source ('mymasterserver.example.com', 3306,
       'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);
   ```

10. Nel cluster di database Amazon Aurora chiama la procedura mysql.rds_start_replication per avviare la replica.

    ```
    CALL mysql.rds_start_replication;
    ```

Dopo aver stabilito la replica tra l'istanza My SQL DB di origine e il cluster Amazon Aurora DB, puoi aggiungere repliche Aurora al tuo cluster Amazon Aurora DB. È possibile quindi connettere le repliche di Aurora per il dimensionamento della lettura dei dati. Per informazioni sulla creazione di una replica di Aurora, consulta Aggiunta di repliche di Aurora a un cluster di database.

Ottimizzazione della replica dei log binari per Aurora MySQL

Di seguito, è possibile imparare a ottimizzare le prestazioni di replica dei log binari e risolvere i problemi correlati in Aurora MySQL.

> **Tip**
>
> Questa discussione presuppone che tu abbia familiarità con il meccanismo di replica dei log binari MySQL e del suo funzionamento. Per informazioni di base, consulta Implementazione della replica nella documentazione di MySQL.

Replica di log binari multithread

Con la replica del log binario multithread, un thread SQL legge gli eventi dal log di inoltro e li mette in coda per l'applicazione dei thread SQL worker. I thread di lavoro SQL sono gestiti dal thread coordinatore. Gli eventi di log binari vengono applicati in parallelo quando possibile. Il livello di parallelismo dipende da fattori quali versione, parametri, progettazione dello schema e caratteristiche del carico di lavoro.

La replica di log binari multithread è supportata in Aurora MySQL versione 3 e in Aurora MySQL versione 2.12.1 e successive. Affinché una replica multithread elabori in modo efficiente gli eventi binlog in parallelo, è necessario configurare l'origine per la replica multithread del log binario e l'origine deve utilizzare una versione che includa le informazioni sul parallelismo nei suoi file di log binari.

Quando un'istanza DB Aurora MySQL è configurata per utilizzare la replica binaria dei log, per impostazione predefinita l'istanza di replica utilizza la replica a thread singolo per le versioni di Aurora MySQL precedenti alla 3.04. Per abilitare la replica multithread, si aggiorna il parametro a un valore maggiore rispetto al gruppo di parametri personalizzato. `replica_parallel_workers` `1`

Per Aurora MySQL versione 3.04 e successive, la replica è multithread per impostazione predefinita, con impostazione predefinita su. `replica_parallel_workers` `4` È possibile modificare questo parametro nel gruppo di parametri personalizzato.

Per aumentare la resilienza del database in caso di arresti imprevisti, si consiglia di abilitare la replica GTID sull'origine e di GTIDs consentirla sulla replica. Per consentire la replica GTID, imposta `gtid_mode` su sia sull'origine che sulla replica. `ON_PERMISSIVE` Per ulteriori informazioni sulla replica basata su GTID, consultare Utilizzo della replica GTID basata.

Le opzioni di configurazione seguenti consentono di ottimizzare la replica multithread. Per informazioni sull'utilizzo, consulta Opzioni e variabili di replica e registrazione binaria nel Manuale di riferimento MySQL. Per ulteriori informazioni sulla replica multithread, consulta il blog MySQL Improving the Parallel Applier with Writeset-based Dependency Tracking.

I valori ottimali dei parametri dipendono da diversi fattori. Ad esempio, le prestazioni per la replica dei log binari sono influenzate dalle caratteristiche del carico di lavoro del database e dalla classe di istanza database su cui è in esecuzione la replica. Pertanto, si consiglia di testare a fondo tutte le modifiche a questi parametri di configurazione prima di applicare nuove impostazioni dei parametri a un'istanza di produzione:

- `binlog_format recommended value`— impostato su `ROW`
- `binlog_group_commit_sync_delay`
- `binlog_group_commit_sync_no_delay_count`
- `binlog_transaction_dependency_history_size`
- `binlog_transaction_dependency_tracking`— il valore consigliato è `WRITESET`
- `replica_preserve_commit_order`
- `replica_parallel_type`— il valore consigliato è `LOGICAL_CLOCK`
- `replica_parallel_workers`
- `replica_pending_jobs_size_max`
- `transaction_write_set_extraction`— il valore consigliato è `XXHASH64`

Lo schema e le caratteristiche del carico di lavoro sono fattori che influiscono sulla replica in parallelo. I fattori più comuni sono i seguenti.

- Assenza di chiavi primarie: RDS non è in grado di stabilire la dipendenza da writeset per le tabelle senza chiavi primarie. Con `ROW` format, è possibile eseguire una singola istruzione su più righe con un'unica scansione completa della tabella sull'origine, ma si ottiene una scansione completa della tabella per riga modificata sulla replica. L'assenza di chiavi primarie riduce in modo significativo il throughput di replica.
- Presenza di chiavi esterne: se sono presenti chiavi esterne, Amazon RDS non può utilizzare la dipendenza writeset per il parallelismo delle tabelle con la relazione FK.
- Dimensioni delle transazioni: se una singola transazione si estende su dozzine o centinaia di megabyte o gigabyte, il thread coordinatore e uno dei thread di lavoro potrebbero impiegare molto tempo a elaborare solo quella transazione. Durante questo periodo, tutti gli altri thread di lavoro potrebbero rimanere inattivi dopo aver completato l'elaborazione delle transazioni precedenti.

In Aurora MySQL versione 3.06 e successive, è possibile migliorare le prestazioni per le repliche di log binari durante la replica di transazioni per tabelle di grandi dimensioni con più di un indice

secondario. Questa funzionalità introduce un pool di thread per applicare le modifiche all'indice secondario in parallelo su una replica binlog. La funzionalità è controllata dal parametro del cluster `aurora_binlog_replication_sec_index_parallel_workers` DB, che controlla il numero totale di thread paralleli disponibili per applicare le modifiche all'indice secondario. Il parametro è impostato su `0` (disabilitato) per impostazione predefinita. L'attivazione di questa funzionalità non richiede il riavvio dell'istanza. Per abilitare questa funzionalità, interrompi la replica in corso, imposta il numero desiderato di thread di lavoro paralleli e quindi riavvia la replica.

Ottimizzazione della replica binlog

In Aurora MySQL 2.10 e versioni successive, Aurora applica automaticamente un'ottimizzazione nota come cache I/O binlog alla replica del log binario. Inserendo nella cache gli eventi binlog più recenti, questa ottimizzazione è progettata per migliorare le prestazioni del thread di dump di binlog limitando al contempo l'impatto sulle transazioni in primo piano sull'istanza di origine binlog.

> **Note**
>
> Questa memoria utilizzata per questa funzione è indipendente dall'impostazione `binlog_cache` MySQL.
> Questa funzione non si applica alle istanze DB Aurora che utilizzano le classi di istanza `db.t2` e `db.t3`.

Non è necessario regolare alcun parametro di configurazione per attivare questa ottimizzazione. In particolare, se avevi regolato il parametro di configurazione `aurora_binlog_replication_max_yield_seconds` su un valore diverso da zero nelle versioni precedenti di Aurora MySQL, reimpostalo su zero per le versioni attualmente disponibili.

Le variabili `aurora_binlog_io_cache_reads` di stato `aurora_binlog_io_cache_read_requests` consentono di monitorare la frequenza con cui i dati vengono letti dalla cache I/O binlog.

- `aurora_binlog_io_cache_read_requests`: mostra il numero di richieste di lettura I/O binlog dalla cache.
- `aurora_binlog_io_cache_reads`: mostra il numero di letture I/O binlog che recuperano informazioni dalla cache.

La seguente query SQL calcola la percentuale di richieste di lettura binlog che sfruttano le informazioni memorizzate nella cache. In questo caso, più il rapporto è vicino a 100, migliore è.

```
mysql> SELECT
  (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_reads')
  / (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_read_requests')
  * 100
  as binlog_io_cache_hit_ratio;
+---------------------------+
| binlog_io_cache_hit_ratio |
+---------------------------+
|         99.99847949080622 |
+---------------------------+
```

La funzione di cache I/O binlog include anche nuovi parametri relativi ai thread di dump di binlog. I thread di dump sono i thread creati quando le nuove repliche di binlog sono collegate all'istanza fonte binlog.

I parametri del thread di dump vengono stampati nel log del database ogni 60 secondi con il prefisso `[Dump thread metrics]`. I parametri includono informazioni per ogni replica binlog, ad esempio `Secondary_id`, `Secondary_uuid`, il nome del file binlog e la posizione in cui ogni replica sta leggendo. I parametri includono anche `Bytes_behind_primary` che rappresenta la distanza in byte tra l'origine della replica e la replica. Questo parametro misura il ritardo del thread I/O di replica. Tale cifra è diversa dal ritardo del thread dell'applicatore SQL della replica, rappresentato dal parametro `seconds_behind_master` sulla replica binlog. È possibile determinare se le repliche di binlog stiano recuperando l'origine o rimangano indietro controllando se la distanza diminuisce o aumenta.

Configurazione di binlog avanzato per Aurora MySQL

Il file di log binario avanzato riduce il sovraccarico delle prestazioni di elaborazione causato dall'attivazione del file di log binario, che in alcuni casi può arrivare fino al 50%. Con il file di log binario avanzato, questo sovraccarico può essere ridotto a circa il 13%. Per ridurre il sovraccarico, il file di log binario avanzato scrive i log binari e i log delle transazioni nello spazio di archiviazione in parallelo, il che riduce al minimo i dati scritti al momento del commit della transazione.

L'utilizzo del file di log binario avanzato migliora anche i tempi di ripristino del database dopo riavvii e failover fino al 99% rispetto al file di log binario della community MySQL. Il file di log binario avanzato

è compatibile con i carichi di lavoro esistenti basati sul file di log binario e viene utilizzato nello stesso modo in cui si utilizza il file di log binario della community MySQL.

Enhanced binlog è disponibile su Aurora MySQL versione 3.03.1 e successive.

Argomenti

- Configurazione dei parametri del file di log binario avanzato
- Altri parametri correlati
- Differenze tra file di log binario avanzato e file di log binario avanzato della community MySQL
- CloudWatch Metriche Amazon per binlog migliorato
- Limitazioni del file di log binario avanzato

Configurazione dei parametri del file di log binario avanzato

È possibile passare dal file di log binario della community MySQL al file di log binario avanzato attivando/disattivando i relativi parametri. Gli utenti esistenti di file di log binario possono continuare a leggere e consumare i file di log binario senza interruzioni nella sequenza di file di log binario.

Per attivare il binlog avanzato, imposta i seguenti parametri:

Parametro	Predefinito	Descrizione
`binlog_format`	–	Impostare il parametro `binlog_format` sul formato di registrazione binaria desiderato per attivare il file di log binario avanzato. Assicurarsi che `binlog_format parameter` non sia impostato su OFF. Per ulteriori informazioni, consultare Configurazione del log binario di Aurora MySQL.
`aurora_enhanced_binlog`	`0`	Impostare il valore di questo parametro su 1 nel gruppo di

Parametro	Predefinito	Descrizione
		parametri del cluster database associato al cluster Aurora MySQL. Quando si modifica il valore di questo parametro, è necessario riavviare l'istanza di scrittura quando il valore di `DBClusterParameterGroupStatus` viene visualizzato come `pending-reboot` .
`binlog_backup`	`1`	Disattivare questo parametro per attivare il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su `0`.
`binlog_replication_globaldb`	`1`	Disattivare questo parametro per attivare il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su `0`.

> **Important**
>
> È possibile disattivare i parametri `binlog_backup` e `binlog_replication_globaldb` solo in caso di utilizzo del file di log binario avanzato.

Per disattivare il binlog avanzato, imposta i seguenti parametri:

Parametro	Descrizione
`aurora_enhanced_binlog`	Impostare il valore di questo parametro su `0` nel gruppo di parametri del cluster database

Parametro	Descrizione
	associato al cluster Aurora MySQL. Ogni volta che si modifica il valore di questo parametro, è necessario riavviare l'istanza di scrittura quando il valore di `DBCluster ParameterGroupStatus` viene visualizzato come `pending-reboot` .
`binlog_backup`	Attivare questo parametro quando si disattiva il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su 1.
`binlog_replication_globaldb`	Attivare questo parametro quando si disattiva il file di log binario avanzato. A tale scopo, impostare il valore di questo parametro su 1.

Per verificare se il file di log binario avanzato è attivo, usare il seguente comando nel client MySQL:

```
mysql>show status like 'aurora_enhanced_binlog';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| aurora_enhanced_binlog | ACTIVE |
+------------------------+--------+
1 row in set (0.00 sec)
```

Quando il file di log binario avanzato è attivo, l'output mostra `ACTIVE` per `aurora_enhanced_binlog`.

Altri parametri correlati

Quando si attiva il file di log binario avanzato, questa operazione interessa i seguenti parametri:

- Il parametro `max_binlog_size` è visibile ma non modificabile. Il relativo valore predefinito `134217728` viene impostato automaticamente su `268435456` quando il file di log binario avanzato è attivato.

- A differenza file di log binario avanzato della community MySQL, `binlog_checksum` non agisce come parametro dinamico quando il file di log binario avanzato è attivato. Affinché la modifica a questo parametro abbia effetto, è necessario riavviare manualmente il cluster database anche quando `ApplyMethod` è `immediate`.
- Il valore impostato per il parametro `binlog_order_commits` non ha alcun effetto sull'ordine dei commit quando il file di log binario avanzato è attivato. I commit vengono sempre ordinati senza ulteriori implicazioni in termini di prestazioni.

Differenze tra file di log binario avanzato e file di log binario avanzato della community MySQL

Il binlog avanzato interagisce in modo diverso con i cloni, i backup e il database globale Aurora rispetto al binlog MySQL della community. Si consiglia di analizzare le seguenti differenze prima di utilizzare il file di log binario avanzato.

- I file binlog avanzati del cluster DB di origine non sono disponibili su un cluster DB clonato.
- I file binlog avanzati non sono inclusi nei backup di Aurora. Pertanto, i file di log binario avanzati del cluster database di origine non sono disponibili dopo il ripristino di un cluster database nonostante il relativo periodo di conservazione impostato.
- Se utilizzati con un database globale Aurora, i file di log binario avanzati del cluster database primario non vengono replicati nel cluster database nelle regioni secondarie.

Examples (Esempi)

Negli esempi seguenti vengono illustrate le differenze tra file di log binario avanzati e file di log binario della community MySQL.

Su un cluster database ripristinato o clonato

Quando il file di log binario avanzato è attivato, i file di log binario storici non sono disponibili nel cluster database ripristinato o clonato. Dopo un'operazione di ripristino o clonazione, se binlog è attivato, il nuovo cluster DB inizia a scrivere la propria sequenza di file binlog, a partire da 1 (.000001). mysql-bin-changelog

Per attivare il file di log binario avanzato dopo un'operazione di ripristino o clonazione, impostare i parametri del cluster database richiesti sul cluster database ripristinato o clonato. Per ulteriori informazioni, consulta Configurazione dei parametri del file di log binario avanzato.

Example Esempio: operazione di clonazione o ripristino eseguita quando è attivato il binlog avanzato

Cluster database di origine:

```
mysql> show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog turned on
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

In un cluster database ripristinato o clonato, non viene eseguito il backup dei file di log binario quando è attivato il file di log binario avanzato. Per evitare discontinuità nei dati del file di log binario, non sono disponibili nemmeno i file di log binario scritti prima di attivare il file di log binario avanzato.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> New sequence of Binlog files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Esempio: operazione di clonazione o ripristino eseguita quando il binlog avanzato è disattivato

Cluster DB di origine:

```
mysql>show binary logs;
```

```
+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Il binlog avanzato viene disabilitato dopo`mysql-bin-changelog.000003`. In un cluster database ripristinato o clonato, sono disponibili file di log binario scritti dopo aver disattivato il file di log binario avanzato.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Su un database globale Amazon Aurora

Su un database globale Amazon Aurora, i dati del file di log binario del cluster database primario non vengono replicati nei cluster database secondari. Dopo un processo di failover tra regioni, i dati del file di log binario non sono disponibili nel cluster database primario appena promosso. Se binlog è attivato, il cluster DB appena promosso avvia la propria sequenza di file binlog, a partire da 1 (mysql-bin-changelog.000001).

Per attivare il file di log binario avanzato dopo il failover, è necessario impostare i parametri del cluster database richiesti sul cluster database secondario. Per ulteriori informazioni, consulta Configurazione dei parametri del file di log binario avanzato.

Example Esempio: l'operazione di failover globale del database viene eseguita quando è attivato il binlog avanzato

Vecchio cluster database primario (prima del failover):

```
mysql>show binary logs;

+---------------------------+-----------+-----------+
| Log_name                  | File_size | Encrypted |
+---------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog enabled
+---------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Nuovo cluster database primario (dopo il failover):

I file di log binario non vengono replicati nelle regioni secondarie quando il file di log binario avanzato è attivato. Per evitare discontinuità nei dati del file di log binario, i file di log binario scritti prima di attivare il file di log binario avanzato non sono disponibili.

```
mysql>show binary logs;

+---------------------------+-----------+-----------+
| Log_name                  | File_size | Encrypted |
+---------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> Fresh sequence of Binlog
 files
+---------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Esempio: l'operazione di failover globale del database viene eseguita quando il binlog avanzato è disattivato

Cluster database di origine:

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Cluster database ripristinato o clonato:

Il binlog avanzato viene disabilitato dopo. `mysql-bin-changelog.000003` I file di log binario scritti dopo aver disattivato il file di log binario avanzato vengono replicati e sono disponibili nel cluster database appena promosso.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
3 rows in set (0.00 sec)
```

CloudWatch Metriche Amazon per binlog migliorato

Le seguenti CloudWatch metriche di Amazon vengono pubblicate solo quando è attivato il binlog avanzato.

CloudWatch parametro	Descrizione	unità
ChangeLogBytesUsed	Quantità di spazio di archiviazione in byte utilizzato dal file di log binario avanzato.	Byte
ChangeLogReadIOPs	Numero di operazioni I/O di lettura eseguite nel file di log binario avanzato in intervalli di 5 minuti.	Conteggio per 5 minuti
ChangeLogWriteIOPs	Numero di operazioni I/O di scrittura su disco eseguite nel file di log binario avanzato in intervalli di 5 minuti.	Conteggio per 5 minuti

Limitazioni del file di log binario avanzato

Le seguenti limitazioni si applicano ai cluster database Amazon Aurora quando il file di log binario avanzato è attivato.

- Enhanced binlog è supportato solo su Aurora MySQL versione 3.03.1 e successive.
- I file di log binario avanzati scritti sul cluster database primario non vengono copiati nei cluster database clonati o ripristinati.
- Se utilizzati con un database globale Amazon Aurora, i file di log binario avanzati del cluster database primario non vengono replicati nei cluster database secondari. Pertanto, dopo il processo di failover, i dati dei file di log binario storici non sono disponibili nel nuovo cluster database primario.
- I seguenti parametri di configurazione del file di log binario vengono ignorati:
  - `binlog_group_commit_sync_delay`
  - `binlog_group_commit_sync_no_delay_count`
  - `binlog_max_flush_queue_time`
- Non è possibile eliminare o rinominare una tabella danneggiata in un database. Per eliminare queste tabelle, puoi contattare. Supporto

- La cache I/O del file di log binario è disabilitata quando il file di log binario avanzato è attivato. Per ulteriori informazioni, consulta Ottimizzazione della replica dei log binari per Aurora MySQL.

> **Note**
>
> Il file di log binario avanzato è caratterizzato da miglioramenti delle prestazioni di lettura simili a quelli della cache I/O del file di log binario e da ulteriori ottimizzazioni delle prestazioni di scrittura.

- La funzionalità Backtrack non è supportata. Il file di log binario avanzato non può essere attivato in un cluster database nelle seguenti condizioni:
  - Cluster database con la funzionalità Backtrack abilitata.
  - Cluster DB in cui la funzionalità backtrack era precedentemente abilitata, ma ora è disabilitata.
  - Cluster database ripristinato da un cluster database di origine o da uno snapshot con la funzionalità Backtrack abilitata.

Utilizzo della replica GTID basata

Il seguente contenuto spiega come utilizzare gli identificatori di transazione globali (GTIDs) con la replica di log binari (binlog) tra le istanze di . tra un SQL cluster Aurora My e una fonte esterna.

> **Note**
>
> Per Aurora, è possibile utilizzare questa funzionalità solo con i SQL cluster Aurora My che utilizzano la replica binlog da o verso un database My esterno. SQL L'altro database potrebbe essere un'SQListanza Amazon RDS My, un SQL database My locale o un cluster Aurora DB in un altro. Regione AWS Per sapere come configurare questo tipo di replica, consultare Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Se utilizzi la replica binlog e non hai familiarità con la replica GTID basata su MySQL, consulta Replica con identificatori di transazione globali nella documentazione My. SQL

GTIDla replica basata è supportata per Aurora SQL My versione 2 e 3.

Argomenti

- Panoramica degli identificatori globali delle transazioni () GTIDs

- Parametri per la replica basata GTID
- Abilitazione della replica GTID basata per un cluster Aurora My SQL
- Disabilitazione della replica GTID basata su Aurora My DB SQL

Panoramica degli identificatori globali delle transazioni () GTIDs

Gli identificatori di transazione globali (GTIDs) sono identificatori univoci generati per le transazioni My confermate. SQL È possibile utilizzarlo GTIDs per rendere la replica binlog più semplice e facile da risolvere.

> **Note**
>
> Quando Aurora sincronizza i dati tra le istanze database in un cluster, tale meccanismo di replica non coinvolge i log binari (binlog). Per Aurora MySQL, la replica GTID basata su Aurora My si applica solo quando si utilizza anche la replica binlog per eseguire la replica all'interno o all'esterno di un cluster Aurora My SQL DB da un database esterno compatibile con My. SQL

My SQL utilizza due diversi tipi di transazioni per la replica binlog:

- GTIDtransazioni: transazioni identificate da un. GTID
- Transazioni anonime: transazioni a cui non è GTID assegnato un.

In una configurazione di replica, GTIDs sono uniche in tutte le istanze DB. GTIDssemplificano la configurazione della replica perché quando li si utilizza, non è necessario fare riferimento alle posizioni dei file di registro. GTIDssemplificano inoltre il monitoraggio delle transazioni replicate e determinano se l'istanza di origine e le repliche sono coerenti.

In genere si utilizza la replica GTID basata con Aurora quando si esegue la replica da un database esterno compatibile con SQL My in un cluster Aurora. Puoi configurare questa configurazione di replica come parte di una migrazione da un RDS database locale o Amazon ad Aurora My. SQL Se il database esterno lo utilizza giàGTIDs, l'abilitazione della replica GTID basata per il cluster Aurora semplifica il processo di replica.

È possibile configurare la replica GTID basata per un cluster Aurora SQL My impostando innanzitutto i parametri di configurazione pertinenti in un gruppo di parametri del cluster DB. Quindi associ tale gruppo di parametri al cluster.

Parametri per la replica basata GTID

Utilizzate i seguenti parametri per configurare la replica GTID basata.

Parametro	Valori validi	Descrizione
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF`specifica che le nuove transazioni sono transazioni anonime (ovvero non necessari eGTIDs) e che una transazione deve essere anonima per essere replicata. `OFF_PERMISSIVE` indica che le nuove transazioni sono anonime, ma tutte le transazio ni possono essere replicate. `ON_PERMISSIVE` specifica che le nuove transazioni sono transazioni, ma tutte GTID le transazioni possono essere replicate. `ON`specifica che le nuove transazioni sono GTID transazioni e che una transazione deve essere una GTID transazione da replicare.
`enforce_gtid_consistency`	`OFF`, `ON`, `WARN`	`OFF`consente alle transazioni di violare la coerenzaGTID. `ON`impedisce che le transazioni GTID violino la coerenza. `WARN`consente alle transazioni di violare la GTID coerenza ma genera un avviso quando si verifica una violazione.

> **Note**
>
> Nel AWS Management Console, il `gtid_mode` parametro appare come`gtid-mode`.

Per la replica GTID basata, usa queste impostazioni per il gruppo di parametri del cluster DB per il tuo cluster Aurora SQL My DB:

- `ON`e `ON_PERMISSIVE` si applicano solo alla replica in uscita da un cluster Aurora My. SQL Entrambi questi valori fanno sì che il cluster Aurora DB venga utilizzato GTIDs per le transazioni replicate su un database esterno. `ON`richiede che il database esterno utilizzi anche la GTID replica basata. `ON_PERMISSIVE`rende facoltativa la replica GTID basata sul database esterno.
- `OFF_PERMISSIVE`, se impostato, significa che il cluster di database Aurora può accettare la replica in ingresso da un database esterno. Può farlo indipendentemente dal fatto che il database esterno utilizzi la replica GTID basata o meno.
- `OFF`, se impostato, significa che il cluster Aurora DB accetta solo la replica in ingresso da database esterni che non utilizzano la replica basata. GTID

> **Tip**
>
> La replica in entrata è lo scenario di replica binlog più comune per i cluster Aurora My. SQL Per la replica in entrata, si consiglia di impostare la modalità su. GTID `OFF_PERMISSIVE` Tale impostazione consente la replica in entrata da database esterni indipendentemente dalle GTID impostazioni dell'origine di replica.

Per ulteriori informazioni sui gruppi di parametri, consultare .

Abilitazione della replica GTID basata per un cluster Aurora My SQL

Quando la replica GTID basata è abilitata per un cluster Aurora SQL My DB, GTID le impostazioni si applicano alla replica binlog in entrata e in uscita.

Per abilitare la replica GTID basata su un cluster Aurora My SQL

1. Creare o modificare un gruppo di parametri del cluster di database utilizzando le seguenti impostazioni dei parametri:

- `gtid_mode` – `ON` o `ON_PERMISSIVE`
- `enforce_gtid_consistency` – `ON`

2. Associa il gruppo di parametri del cluster DB al cluster Aurora MySQL. A tale scopo, seguire la procedura descritta in .
3. (Facoltativo) Specificate la modalità di assegnazione GTIDs alle transazioni che non le includono. Per eseguire questa operazione, chiama la procedura archiviata in mysql.rds_assign_gtids_to_anonymous_transactions (Aurora My versione 3) SQL.

Disabilitazione della replica GTID basata su Aurora My DB SQL

È possibile disabilitare la replica GTID basata su un cluster Aurora SQL My DB. Ciò significa che il cluster Aurora non può eseguire la replica binlog in entrata o in uscita con database esterni che utilizzano la replica basata. GTID

> **Note**
>
> Nella seguente procedura, replica di lettura indica la destinazione della replica in una configurazione Aurora con replica basata sui log binari verso o da un database esterno. Non indica istanze database di replica a sola lettura di Aurora. Ad esempio, quando un cluster Aurora accetta la replica in ingresso da un'origine esterna, l'istanza primaria Aurora funge da replica di lettura per la replica basata sui log binari.

Per ulteriori dettagli sulle procedure memorizzate citate in questa sezione, consultare Aurora Il mio riferimento alla procedura SQL memorizzata.

Per disabilitare la replica GTID basata su un cluster Aurora My DB, un'istanza SQL My DB repliche di lettura

1. Sulle repliche Aurora, esegui la seguente procedura:

   Per la versione 3

   ```
   CALL mysql.rds_set_source_auto_position(0);
   ```

   Per la versione 2

```
CALL mysql.rds_set_master_auto_position(0);
```

2. Reimpostare `gtid_mode` su `ON_PERMISSIVE`.

   a. Assicurati che il gruppo di parametri del cluster DB associato al SQL cluster Aurora My sia `gtid_mode` impostato su. `ON_PERMISSIVE`

      Per ulteriori informazioni sull'impostazione dei parametri di configurazione mediante i gruppi di parametri, consultare .

   b. Riavviare il cluster Aurora My SQL DB.

3. Reimpostare `gtid_mode` su `OFF_PERMISSIVE`.

   a. Assicurati che il gruppo di parametri del cluster DB associato al SQL cluster Aurora My sia `gtid_mode` impostato su. `OFF_PERMISSIVE`

   b. Riavviare il cluster Aurora My SQL DB.

4. Attendi che tutte le GTID transazioni vengano applicate all'istanza primaria di Aurora. Per verificare che vengano applicate, procedi nel seguente modo:

   a. Sull'istanza primaria di Aurora, esegui il `SHOW MASTER STATUS` comando.

      L'output dovrebbe essere simile al seguente.

```
File                        Position
-------------------------------------
mysql-bin-changelog.000031      107
-------------------------------------
```

      Annotare il file e la posizione nell'output.

   b. In ogni replica letta, utilizzate le informazioni sul file e sulla posizione dalla relativa istanza di origine nel passaggio precedente per eseguire la seguente query:

      Per la versione 3

```
SELECT SOURCE_POS_WAIT('file', position);
```

      Per la versione 2

```
SELECT MASTER_POS_WAIT('file', position);
```

Ad esempio, se il nome del file è `mysql-bin-changelog.000031` e la posizione è`107`, esegui la seguente istruzione:

Per la versione 3

```
SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
```

Per la versione 2

```
SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
```

5. Reimposta i GTID parametri per disabilitare la replica GTID basata.

   a. Assicurati che il gruppo di parametri del cluster DB associato al SQL cluster Aurora My abbia le seguenti impostazioni dei parametri:

      - `gtid_mode` – `OFF`
      - `enforce_gtid_consistency` – `OFF`

   b. Riavviare il cluster Aurora My SQL DB.

Utilizzo dell'inoltro di scrittura locale in un cluster database Amazon Aurora MySQL

L'inoltro di scrittura locale (all'interno del cluster) consente alle applicazioni di eseguire transazioni di lettura/scrittura direttamente su una replica di Aurora. Queste transazioni vengono quindi inoltrate all'istanza database di scrittura in attesa che venga eseguito il commit. È possibile utilizzare l'inoltro locale di scrittura quando le applicazioni richiedono read-after-write coerenza, ovvero la capacità di leggere l'ultima scrittura in una transazione.

Le repliche di lettura ricevono aggiornamenti in modo asincrono dall'istanza di scrittura. Senza inoltro di scrittura, è necessario effettuare transazioni di lettura che richiedono read-after-write coerenza sull'istanza Writer DB. Oppure è necessario sviluppare una logica dell'applicazione personalizzata complessa per sfruttare più repliche di lettura per la scalabilità. Le applicazioni devono suddividere completamente tutto il traffico in lettura e in scrittura, mantenendo due gruppi di connessioni al database per inviare il traffico all'endpoint corretto. Questi costi di sviluppo complicano la progettazione dell'applicazione quando le query fanno parte di una singola sessione logica, o transazione, all'interno dell'applicazione. Inoltre, poiché il ritardo di replica può variare tra le repliche di lettura, è difficile ottenere una coerenza di lettura globale tra tutte le istanze nel database.

L'inoltro di scrittura evita la necessità di suddividere tali transazioni o inviarle esclusivamente all'istanza di scrittura, semplificando lo sviluppo dell'applicazione. Questa nuova funzionalità consente di ottenere facilmente il dimensionamento della lettura per carichi di lavoro che devono leggere l'ultima scrittura in una transazione e non sono sensibili alla latenza di scrittura.

L'inoltro di scrittura locale è diverso dall'inoltro di scrittura globale, che inoltra le scritture da un cluster database secondario al cluster database primario in un database globale Aurora. Puoi utilizzare l'inoltro di scrittura locale in un cluster database che fa parte di un database globale Aurora. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

L'inoltro di scrittura locale richiede Aurora MySQL versione 3.04 o successive.

Argomenti

- Abilitazione dell'inoltro di scrittura locale
- Verifica se l'inoltro di scrittura è abilitato un cluster database
- Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura
- Livelli di isolamento per l'inoltro di scrittura
- Coerenza di lettura per l'inoltro di scrittura

- Esecuzione di istruzioni a più parti con inoltro scrittura
- Transazioni con inoltro di scrittura
- Parametri di configurazione per l'inoltro di scrittura
- Amazon CloudWatch Metrics e Aurora Le SQL mie variabili di stato per l'inoltro di scrittura
- Identificazione delle transazioni e delle query inoltrate

Abilitazione dell'inoltro di scrittura locale

Per impostazione predefinita, l'inoltro di scrittura locale non è abilitato per i cluster Aurora My DB. SQL Abilitare l'inoltro di scrittura locale a livello di cluster, non a livello di istanza.

> **Important**
>
> È inoltre possibile abilitare l'inoltro di scrittura locale per le repliche di lettura interregionali che utilizzano la registrazione binaria, ma le operazioni di scrittura non vengono inoltrate all'origine Regione AWS. Vengono inoltrati all'istanza Writer DB del cluster di replica di lettura binlog.
> Usa questo metodo solo se hai un caso d'uso per scrivere sulla replica di lettura binlog nella replica di lettura secondaria Regione AWS. Altrimenti, potresti ritrovarti con uno scenario «split-brain» in cui i set di dati replicati non sono coerenti tra loro.
> Si consiglia di utilizzare l'inoltro di scrittura globale con database globali, anziché l'inoltro di scrittura locale su repliche di lettura tra regioni, a meno che non sia assolutamente necessario. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Console

Utilizzo di AWS Management Console, seleziona la casella di controllo Attiva l'inoltro di scrittura locale in Read replica write forwarding quando crei o modifichi un cluster DB.

AWS CLI

Per abilitare l'inoltro di scrittura con AWS CLI, usa l'opzione. `--enable-local-write-forwarding` Questa opzione funziona quando si crea un nuovo cluster database secondario tramite il comando `create-db-cluster`. Inoltre, funziona quando si modifica un cluster database esistente

tramite il comando `modify-db-cluster`. È possibile disabilitare l'inoltro di scrittura utilizzando l'`--no-enable-local-write-forwarding`opzione con questi stessi CLI comandi.

L'esempio seguente crea un cluster Aurora My SQL DB con l'inoltro di scrittura abilitato.

```
aws rds create-db-cluster \
  --db-cluster-identifier write-forwarding-test-cluster \
  --enable-local-write-forwarding \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.04.0 \
  --master-username myuser \
  --master-user-password mypassword \
  --backup-retention 1
```

Vengono quindi create istanze database di scrittura e lettura in modo da poter utilizzare l'inoltro di scrittura. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

RDS API

Per abilitare l'inoltro di scrittura tramite Amazon RDSAPI, imposta il `EnableLocalWriteForwarding` parametro su. `true` Questo parametro funziona quando si crea un nuovo cluster database secondario utilizzando l'operazione `CreateDBCluster`. Funziona anche quando si modifica un cluster database esistente utilizzando l'operazione `ModifyDBCluster`. È possibile disattivare l'inoltro di scrittura impostando il parametro `EnableLocalWriteForwarding` su `false`.

Abilitazione dell'inoltro di scrittura per le sessioni di database

Il parametro `aurora_replica_read_consistency` è un parametro database e un parametro del cluster database che abilita l'inoltro di scrittura. È possibile specificare `EVENTUAL`, `SESSION` o `GLOBAL` per il livello di coerenza di lettura. Per ulteriori informazioni sui livelli di coerenza, consulta Coerenza di lettura per l'inoltro di scrittura.

A questo parametro si applicano le seguenti regole:

- Il valore predefinito è '' (null).
- L'inoltro di scrittura è disponibile solo se `aurora_replica_read_consistency` è impostato su `EVENTUAL`, `SESSION` o `GLOBAL`. Questo parametro è rilevante solo nelle istanze di lettura di cluster database secondari con l'inoltro di scrittura abilitato.

- Non è possibile importare questo parametro (quando vuoto) o annullarne l'impostazione (quando già impostato) all'interno di una transazione con più istruzioni. È possibile modificarlo da un valore valido a un altro valore valido durante una transazione di questo tipo, ma questa azione non è consigliata.

Verifica se l'inoltro di scrittura è abilitato un cluster database

Per determinare se è possibile utilizzare l'inoltro di scrittura in un cluster database, verifica che l'attributo `LocalWriteForwardingStatus` del cluster sia impostato su `enabled`.

Nella AWS Management Console scheda Configurazione della pagina dei dettagli del cluster, viene visualizzato lo stato Enabled for Local read replica write forwarding.

Per visualizzare lo stato dell'impostazione di inoltro della scrittura per tutti i cluster, esegui il comando seguente. AWS CLI

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
    {
        "LocalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-1"
    },
    {
        "LocalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "requested",
        "DBClusterIdentifier": "test-global-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "null",
        "DBClusterIdentifier": "aurora-mysql-v2-cluster"
    }
]
```

Un cluster database può avere i seguenti valori per `LocalWriteForwardingStatus`:

- `disabled`: l'inoltro di scrittura è disabilitato.
- `disabling`: l'inoltro di scrittura è in fase di disabilitazione.
- `enabled`: l'inoltro di scrittura è abilitato.
- `enabling`: l'inoltro di scrittura è in fase di abilitazione.
- `null`: l'inoltro di scrittura non è disponibile per questo cluster database.
- `requested`: l'inoltro di scrittura è stato richiesto, ma non è ancora attivo.

Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura

È possibile utilizzare i seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`. Esistono alcune restrizioni sulle proprietà di queste istruzioni che è possibile utilizzare con l'inoltro di scrittura, come descritto di seguito.
- Istruzioni `SELECT ... LOCK IN SHARE MODE` e `SELECT FOR UPDATE`.
- Istruzioni `PREPARE` e `EXECUTE`.

Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in un cluster database con inoltro di scrittura. Inoltre, le funzioni e le procedure definite dall'utente non sono supportate. Pertanto, l'impostazione `EnableLocalWriteForwarding` è disabilitata per impostazione predefinita per i cluster database. Prima di abilitarla, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.

Le seguenti restrizioni si applicano alle istruzioni SQL utilizzate con l'inoltro di scrittura. In alcuni casi, è possibile utilizzare le istruzioni su cluster database con inoltro di scrittura abilitato. Questo approccio funziona se l'inoltro di scrittura non è abilitato all'interno della sessione dal parametro di configurazione `aurora_replica_read_consistency`. Se si prova a utilizzare un'istruzione quando non è consentita a causa dell'inoltro di scrittura, verrà visualizzato un messaggio di errore simile al seguente:

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

DDL (Data Definition Language)

Connettersi all'istanza database di scrittura per eseguire le istruzioni DDL. Non è possibile eseguirle da istanze database di lettura.

Aggiornamento di una tabella permanente utilizzando i dati di una tabella temporanea

È possibile utilizzare tabelle temporanee in cluster database con l'inoltro di scrittura abilitato. Tuttavia, non è possibile utilizzare un'istruzione DML per modificare una tabella permanente se l'istruzione fa riferimento a una tabella temporanea. Ad esempio, non è possibile utilizzare un'istruzione `INSERT ... SELECT` che accetta i dati da una tabella temporanea.

Transazioni XA

Non è possibile utilizzare le istruzioni seguenti in un cluster database quando l'inoltro di scrittura è abilitato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster database per i quali non è abilitato l'inoltro di scrittura o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Prima di abilitare l'inoltro di scrittura all'interno di una sessione, occorre verificare se il codice utilizza queste istruzioni.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Istruzioni LOAD per tabelle permanenti

Non è possibile utilizzare le istruzioni seguenti in un cluster database con l'inoltro di scrittura abilitato.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Istruzioni plugin

Non è possibile utilizzare le istruzioni seguenti in un cluster database con l'inoltro di scrittura abilitato.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Istruzioni SAVEPOINT

Non è possibile utilizzare le istruzioni seguenti in un cluster database quando l'inoltro di scrittura è abilitato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster database in cui l'inoltro di scrittura non è abilitato o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Prima di abilitare l'inoltro di scrittura all'interno di una sessione, occorre verificare se il codice utilizza queste istruzioni.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Livelli di isolamento per l'inoltro di scrittura

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo il livello di isolamento `REPEATABLE READ`. Sebbene sia possibile utilizzare il livello di isolamento `READ COMMITTED` anche con repliche di Aurora, questa operazione non funziona con l'inoltro di scrittura. Per informazioni sui livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`, consulta Livelli di isolamento di Aurora MySQL.

Coerenza di lettura per l'inoltro di scrittura

È possibile controllare il grado di coerenza di lettura in un cluster database. Il livello di coerenza di lettura determina il tempo di attesa di un cluster database prima di ogni operazione di lettura per garantire che alcune o tutte le modifiche vengano replicate dall'istanza di scrittura. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nel cluster database prima di qualsiasi query successiva. Inoltre, è possibile utilizzare questa impostazione per garantire che le query sul cluster database visualizzino sempre gli aggiornamenti più recenti dall'istanza di scrittura. Questa impostazione si applica anche alle query inviate da altre sessioni o da altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegli un valore per il parametro database `aurora_replica_read_consistency` o il parametro cluster database.

Important

Imposta sempre il parametro database `aurora_replica_read_consistency` o il parametro cluster database quando desideri inoltrare le scritture. In caso contrario,

Aurora non inoltra le scritture. Questo parametro ha un valore vuoto per impostazione predefinita, quindi scegli un valore specifico quando utilizzi questo parametro. Il parametro `aurora_replica_read_consistency` influisce solo sui cluster o sulle istanze database in cui l'inoltro di scrittura è abilitato.

All'aumentare del livello di coerenza, l'applicazione rimane più tempo in attesa della propagazione delle modifiche tra le istanze database. È possibile scegliere il giusto equilibrio tra tempi di risposta rapidi e garanzia che le modifiche apportate in altre istanze database siano completamente disponibili prima dell'esecuzione delle query.

È possibile specificare i valori seguenti per il parametro `aurora_replica_read_consistency`:

- `EVENTUAL`: i risultati delle operazioni di scrittura nella stessa sessione non sono visibili finché l'operazione di scrittura non viene eseguita sull'istanza database di scrittura. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati più vecchi o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica. Questa è la stessa coerenza dei cluster Aurora My SQL DB che non utilizzano l'inoltro di scrittura.
- `SESSION`: tutte le query che utilizzano l'inoltro di scrittura visualizzano i risultati di tutte le modifiche apportate in tale sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati.
- `GLOBAL`: una sessione visualizza tutte le modifiche confermate in tutte le sessioni e le istanze nel cluster database. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando il cluster DB contiene tutti up-to-date i dati salvati dall'autore, a partire dal momento in cui è iniziata la query.

Per informazioni su tutti i parametri di configurazione coinvolti nell'inoltro di scrittura, consulta Parametri di configurazione per l'inoltro di scrittura.

Note

È anche possibile usare `aurora_replica_read_consistency` come una variabile di sessione, ad esempio:

```
mysql> set aurora_replica_read_consistency = 'session';
```

Esempi di utilizzo dell'inoltro di scrittura

Negli esempi seguenti vengono illustrati gli effetti del parametro `aurora_replica_read_consistency` sull'esecuzione delle istruzioni `INSERT` seguite dalle istruzioni `SELECT`. I risultati possono variare, a seconda del valore di `aurora_replica_read_consistency` e della tempistica delle istruzioni.

Per ottenere una maggiore coerenza, è possibile attendere brevemente prima di rilasciare l'istruzione `SELECT`. Oppure Aurora può attendere automaticamente il completamento della replica dei risultati prima di procedere con `SELECT`.

Per informazioni sull'impostazione dei parametri database, consulta .

Example con **`aurora_replica_read_consistency`** impostato su **`EVENTUAL`**

L'esecuzione di un'istruzione `INSERT`, immediatamente seguita da un'istruzione `SELECT`, restituisce un valore per `COUNT(*)` con il numero di righe prima dell'inserimento della nuova riga. Se poco dopo si esegue nuovamente `SELECT`, viene restituito il conteggio delle righe aggiornato. Le istruzioni `SELECT` non attendono.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
```

```
1 row in set (0.00 sec)
```

Example con **`aurora_replica_read_consistency`** impostato su **`SESSION`**

Un'istruzione `SELECT` subito dopo l'istruzione `INSERT` attende fino a quando le modifiche apportate dall'istruzione `INSERT` diventano visibili. Le istruzioni `SELECT` successive non attendono.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)

mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Con l'impostazione di coerenza di lettura ancora impostata su `SESSION`, l'introduzione di una breve attesa dopo l'esecuzione di un'istruzione `INSERT` rende disponibile il conteggio delle righe aggiornate dal momento dell'esecuzione dell'istruzione `SELECT` successiva.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
```

```
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Example con **`aurora_replica_read_consistency`** impostato su **`GLOBAL`**

Ogni istruzione `SELECT` attende che tutte le modifiche ai dati, a partire dall'ora di inizio dell'istruzione, diventino visibili prima di eseguire la query. Il tempo di attesa per ciascuna istruzione `SELECT` varia, a seconda dell'entità di ritardo di replica.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Esecuzione di istruzioni a più parti con inoltro scrittura

Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o un'istruzione `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata all'istanza database di scrittura ed eseguita lì.

Transazioni con inoltro di scrittura

Se la modalità di accesso alle transazioni è impostata su sola lettura, l'inoltro di scrittura non viene utilizzato. È possibile specificare la modalità di accesso per la transazione utilizzando l'istruzione `SET TRANSACTION` o `START TRANSACTION`. È anche possibile specificare la modalità di accesso alle transazioni modificando il valore della variabile di sessione transaction_read_only. È possibile modificare questo valore di sessione solo quando si è connessi a un cluster database in cui l'inoltro di scrittura è abilitato.

Se una transazione di lunga durata non emette alcuna dichiarazione per un periodo di tempo considerevole, potrebbe superare il periodo di timeout inattivo. Questo periodo ha un valore predefinito di un minuto. È possibile impostare il parametro `aurora_fwd_writer_idle_timeout` per aumentarlo fino a un giorno. Una transazione che supera il timeout di inattività viene annullata dall'istanza di scrittura. L'istruzione successiva inviata riceve un errore di timeout. Quindi Aurora esegue il rollback della transazione.

Questo tipo di errore può verificarsi in altri casi in cui l'inoltro di scrittura non diventa disponibile. Ad esempio, Aurora annulla le eventuali transazioni che utilizzano l'inoltro di scrittura se si riavvia il cluster database o si disabilita l'inoltro di scrittura.

Quando un'istanza di scrittura in un cluster che utilizza l'inoltro di scrittura locale viene riavviata, le eventuali transazioni e query attive, inoltrate sulle istanze di lettura che utilizzano l'inoltro di scrittura locale vengono chiuse automaticamente. Dopo che l'istanza di scrittura sarà nuovamente disponibile, è possibile riprovare a eseguire queste transazioni.

Parametri di configurazione per l'inoltro di scrittura

I gruppi di parametri database Aurora contengono impostazioni per la funzionalità di inoltro di scrittura. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`aurora_fwd_writer_idle_timeout`	Cluster	Intero senza segno	60	1–86.400

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`aurora_fwd_writer_max_connections_pct`	Cluster	Intero lungo senza segno	10	0–90
`aurora_replica_read_consistency`	Cluster o istanza	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Per controllare le richieste di scrittura in entrata, usare queste impostazioni:

- `aurora_fwd_writer_idle_timeout`: il numero di secondi per cui l'istanza database di scrittura rimane in attesa su una connessione inoltrata da un'istanza di lettura prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, la sessione viene annullata da Aurora.
- `aurora_fwd_writer_max_connections_pct`: il limite superiore delle connessioni al database che può essere utilizzato su un'istanza database di scrittura per gestire le query inoltrate da istanze di lettura. È espresso come una percentuale dell'impostazione `max_connections` per l'istanza di scrittura. Ad esempio, se `max_connections` è 800 e `aurora_fwd_master_max_connections_pct` o `aurora_fwd_writer_max_connections_pct` è 10, allora lo scrittore consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione `max_connections`.

  Questa impostazione si applica solo all'istanza di scrittura quando l'inoltro di scrittura è abilitato. Se si riduce il valore, le connessioni esistenti non vengono influenzate. Aurora tiene in considerazione il nuovo valore dell'impostazione durante il tentativo di creazione di una nuova connessione da un cluster database. Il valore predefinito è 10, che rappresenta il 10% del valore `max_connections`.

> **Note**
>
> Poiché `aurora_fwd_writer_idle_timeout` e `aurora_fwd_writer_max_connections_pct` s
> parametri del cluster database, tutte le istanze database in ogni cluster hanno gli stessi valori per questi parametri.

Per ulteriori informazioni su `aurora_replica_read_consistency`, consulta Coerenza di lettura per l'inoltro di scrittura.

Per ulteriori informazioni sui gruppi di parametri database, consulta .

Amazon CloudWatch Metrics e Aurora Le SQL mie variabili di stato per l'inoltro di scrittura

Le seguenti CloudWatch metriche di Amazon e le variabili di stato di Aurora SQL My si applicano quando utilizzi l'inoltro di scrittura per Aurora My. SQL Per ulteriori informazioni sulle metriche per le istanze DB di Aurora SQL My writer and reader, consulta i seguenti argomenti.

Argomenti

- Metriche per l'inoltro di scrittura per le istanze Aurora My writer DB SQL
- Metriche per l'inoltro di scrittura per le istanze DB di Aurora My reader SQL

Metriche per l'inoltro di scrittura per le istanze Aurora My writer DB SQL

I seguenti CloudWatch parametri di Amazon si applicano quando utilizzi l'inoltro di scrittura su uno o più cluster DB. Queste metriche sono tutte misurate sull'istanza Writer DB.

CloudWatch parametro	Unità	Descrizione
`ForwardingWriterDMLLatency`	Millisecondi	Tempo medio di elaborazione di ogni DML istruzione inoltrata sull'istanza Writer DB. Non include il tempo impiegato dal cluster database per inoltrare la richiesta di scrittura

CloudWatch parametro	Unità	Descrizione
		o il tempo necessario per replicare le modifiche all'istanza di scrittura.
`ForwardingWriterDMLThroughput`	Conteggio al secondo	Numero di DML istruzioni inoltrate elaborate ogni secondo da questa istanza Writer DB.
`ForwardingWriterOpenSessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer.

Le seguenti variabili di SQL stato Aurora My si applicano quando si utilizza l'inoltro di scrittura su uno o più cluster DB. Queste variabili di stato sono tutte misurate sull'istanza Writer DB.

Aurora La mia SQL variabile di stato	Unità	Descrizione
`Aurora_fwd_writer_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate a questa istanza database writer.
`Aurora_fwd_writer_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate a questa istanza database writer.
`Aurora_fwd_writer_open_sessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer.
`Aurora_fwd_writer_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate a questa istanza database writer.
`Aurora_fwd_writer_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate a questa istanza database writer.

Metriche per l'inoltro di scrittura per le istanze DB di Aurora My reader SQL

Le seguenti CloudWatch metriche vengono misurate su ogni istanza Reader DB in un cluster DB con l'inoltro di scrittura abilitato.

CloudWatch parametro	Unità	Descrizione
`ForwardingReplicaDMLLatency`	Millisecondi	Tempo di risposta medio dell'inoltro DMLs sulla replica.
`ForwardingReplicaDMLThroughput`	Conteggio al secondo	Numero di dichiarazioni inoltrate elaborate ogni secondo. DML
`ForwardingReplicaOpenSessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`ForwardingReplicaReadWaitLatency`	Millisecondi	Tempo medio di attesa di un'istruzione `SELECT` su un'istanza database di lettura prima di raggiungere l'istanza di scrittura. Il grado di attesa dell'istanza database del lettore prima di elaborare una query dipende dall'impostazione `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Conteggio al secondo	Numero totale di istruzioni `SELECT` elaborate ogni secondo in tutte le sessioni che inoltrano scritture.
`ForwardingReplicaSelectLatency`	Millisecondi	Latenza `SELECT` inoltrata, mediata su tutte le istruzion

CloudWatch parametro	Unità	Descrizione
		i SELECT inoltrate all'interno del periodo di monitoraggio.
ForwardingReplicaSelectThroughput	Conteggio al secondo	Velocità di trasmissione effettiva per secondo SELECT inoltrata media all'interno del periodo di monitoraggio.

Le seguenti variabili di SQL stato Aurora My sono misurate su ogni istanza Reader DB in un cluster DB con l'inoltro di scrittura abilitato.

Aurora La mia SQL variabile di stato	Unità	Descrizione
Aurora_fwd_replica_dml_stmt_count	Conteggio	Numero totale di istruzioni DML inoltrate dall'istanza database del lettore.
Aurora_fwd_replica_dml_stmt_duration	Microsecondi	Durata totale di tutte le DML istruzioni inoltrate da questa istanza DB Reader.
Aurora_fwd_replica_errors_session_limit	Conteggio	Numero di sessioni rifiutate dal cluster primario a causa di una delle seguenti condizioni di errore: • istanza di scrittura piena • Troppe istruzioni inoltrate in corso.
Aurora_fwd_replica_open_sessions	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura

Aurora La mia SQL variabile di stato	Unità	Descrizione
		su un'istanza database di lettura.
`Aurora_fwd_replica_read_wait_count`	Conteggio	Numero totale di read-after-write attese su questa istanza DB del lettore.
`Aurora_fwd_replica_read_wait_duration`	Microsecondi	Durata totale delle attese a causa dell'impostazione di coerenza di lettura su questa istanza database del lettore.
`Aurora_fwd_replica_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate dall'istanza database del lettore.
`Aurora_fwd_replica_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate da questa istanza database del lettore.

Identificazione delle transazioni e delle query inoltrate

È possibile usare la tabella `information_schema.aurora_forwarding_processlist` per identificare le transazioni e le query inoltrate. Per ulteriori informazioni su questa tabella, consultare information_schema.aurora_forwarding_processlist.

L'esempio seguente mostra tutte le connessioni inoltrate su un'istanza database di scrittura.

```
mysql> select * from information_schema.AURORA_FORWARDING_PROCESSLIST where
 IS_FORWARDED=1 order by REPLICA_SESSION_ID;

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

```
| ID  | USER     | HOST               | DB       | COMMAND | TIME | STATE        |
 INFO                                    | IS_FORWARDED | REPLICA_SESSION_ID |
 REPLICA_INSTANCE_IDENTIFIER     | REPLICA_CLUSTER_NAME | REPLICA_REGION |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+--------------+--------------------
+---------------------------------+---------------------------------------+
| 648 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest58 SET k=k+1 WHERE id=4802579 |            1 |                637 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
| 650 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest54 SET k=k+1 WHERE id=2503953 |            1 |                639 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+--------------+--------------------
+---------------------------------+----------------------+----------------+
```

Nell'istanza database di lettura di inoltro, è possibile visualizzare i thread associati a queste connessioni database di scrittura eseguendo`SHOW PROCESSLIST`. Il valori `REPLICA_SESSION_ID` sull'istanza di scrittura, 637 e 639, sono identici ai valori`Id` sull'istanza di lettura.

```
mysql> select @@aurora_server_id;

+---------------------------------+
| @@aurora_server_id              |
+---------------------------------+
| my-db-cluster-instance-2        |
+---------------------------------+
1 row in set (0.00 sec)

mysql> show processlist;

+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
| Id  | User     | Host               | db       | Command | Time | State        | Info
                                       |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
| 637 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest12 SET k=k+1 WHERE id=4802579  |
| 639 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest61 SET k=k+1 WHERE id=2503953  |
```

```
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
12 rows in set (0.00 sec)
```

Integrazione di Amazon Aurora SQL My con altri servizi AWS

Amazon Aurora My SQL si integra con altri AWS servizi in modo da poter estendere il cluster Aurora My SQL DB per utilizzare funzionalità aggiuntive nel cloud. AWS Il cluster Aurora My SQL DB può utilizzare AWS i servizi per eseguire le seguenti operazioni:

- Richiama una funzione in modo sincrono o asincrono utilizzando le AWS Lambda funzioni native o. `lambda_sync` `lambda_async` Per ulteriori informazioni, consulta Richiamo di una funzione Lambda da un cluster Amazon Aurora My DB SQL.
- Carica dati da testo o XML file archiviati in un bucket Amazon Simple Storage Service (Amazon S3) nel tuo cluster DB utilizzando il comando or. `LOAD DATA FROM S3` `LOAD XML FROM S3` Per ulteriori informazioni, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.
- Salvataggio dei dati nei file di testo archiviati in un bucket Amazon S3 dal cluster DB utilizzando il comando `SELECT INTO OUTFILE S3`. Per ulteriori informazioni, consulta Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3.
- Aggiunta o rimozione automatica di repliche Aurora con Application Auto Scaling. Per ulteriori informazioni, consulta Auto Scaling di Amazon Aurora con repliche Aurora.
- Esegui analisi del sentiment con Amazon Comprehend o un'ampia varietà di algoritmi SageMaker di apprendimento automatico con AI. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora.

Aurora assicura la possibilità di accedere ad altri AWS servizi utilizzando AWS Identity and Access Management (). IAM Concedi l'autorizzazione ad accedere ad altri AWS servizi creando un IAM ruolo con le autorizzazioni necessarie e quindi associando il ruolo al tuo cluster DB. Per dettagli e istruzioni su come consentire al cluster Aurora My SQL DB di accedere ad altri AWS servizi per tuo conto, consulta. Autorizzazione di Amazon Aurora MySQL ad accedere ad altri servizi AWS per tuo conto

Autorizzazione di Amazon Aurora MySQL ad accedere ad altri servizi AWS per tuo conto

Perché il tuo cluster database Aurora MySQL possa accedere ad altri servizi per tuo conto, crea e configura un ruolo AWS Identity and Access Management (IAM). Questo ruolo autorizza gli utenti del database nel cluster DB ad accedere ad altri servizi AWS. Per ulteriori informazioni, consulta Impostazione IAM dei ruoli per l'accesso AWS services.

È anche necessario configurare il cluster database Aurora per consentire le connessioni in uscita al servizio AWS di destinazione. Per ulteriori informazioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

In questo caso, gli utenti del database possono eseguire le operazioni utilizzando gli altri servizi AWS:

- Chiamata sincrona o asincrona di una funzione AWS Lambda utilizzando le funzioni native `lambda_sync` o `lambda_async`. In alternativa, chiamata asincrona di una funzione AWS Lambda utilizzando la procedura `mysql.lambda_async`. Per ulteriori informazioni, consulta Invocare una funzione Lambda con una funzione Aurora My native SQL.
- Caricamento dei dati da file di testo o XML memorizzati in un bucket Amazon S3 nel cluster DB utilizzando l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3`. Per ulteriori informazioni, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.
- Salvataggio dei dati dal cluster DB nei file di testo memorizzati in un bucket Amazon S3 utilizzando l'istruzione `SELECT INTO OUTFILE S3`. Per ulteriori informazioni, consulta Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3.
- Esportazione dei dati di log in Amazon CloudWatch Logs MySQL. Per ulteriori informazioni, consulta Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch .
- Aggiunta o rimozione automatica di repliche Aurora con Application Auto Scaling. Per ulteriori informazioni, consulta Auto Scaling di Amazon Aurora con repliche Aurora.

Impostazione IAM dei ruoli per l'accesso AWS services

Per consentire al cluster Aurora DB di accedere a un altro AWS servizio, procedi come segue:

1. Crea una IAM politica che conceda l'autorizzazione a AWS servizio. Per ulteriori informazioni, consulta i seguenti argomenti.
   - Creazione di una policy IAM per l'accesso alle risorse Amazon S3
   - Creazione di una policy IAM per l'accesso alle risorseAWS Lambda
   - Creazione di una policy IAM per l'accesso alle risorse CloudWatch Logs
   - Creazione di una policy IAM per l'accesso alle risorseAWS KMS
2. Crea un IAM ruolo e allega la politica che hai creato. Per ulteriori informazioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Associa quel IAM ruolo al tuo cluster Aurora DB. Per ulteriori informazioni, consulta Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

Creazione di una policy IAM per l'accesso alle risorse Amazon S3

Aurora può accedere alle risorse di Amazon S3 per caricare i dati o per salvare i dati da un cluster DB Aurora. Tuttavia, è necessario prima creare una policy IAM che assegni le autorizzazioni del bucket e dell'oggetto che rendono possibile l'accesso di Aurora ad Amazon S3.

La tabella seguente indica le funzionalità di Aurora che possono accedere a un bucket Amazon S3 per tuo conto e le autorizzazioni minime del bucket e dell'oggetto richieste da ciascuna funzionalità.

Caratteristica	Autorizzazioni del bucket	Autorizzazioni dell'oggetto
`LOAD DATA FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`
`LOAD XML FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`

`SELECT INTO OUTFILE S3`	`ListBucket`	`AbortMultipartUpload`
		`DeleteObject`
		`GetObject`
		`ListMultipartUploadParts`
		`PutObject`

La seguente policy aggiunge le autorizzazioni che Amazon S3 potrebbe richiedere da Aurora per accedere al bucket per tuo conto.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
```

```
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

> **Note**
>
> Assicurati di includere entrambe le voci per il valore `Resource`. Aurora ha bisogno delle autorizzazioni sia sul bucket stesso che su tutti gli oggetti all'interno del bucket.
> In base al caso d'uso, potrebbe non essere necessario aggiungere tutte le autorizzazioni presenti nella policy di esempio. Potrebbero inoltre essere richieste altre autorizzazioni. Ad esempio, se il bucket Amazon S3 è crittografato, occorre aggiungere autorizzazioni `kms:Decrypt`.

È possibile seguire i passaggi seguenti per creare una policy IAM che conceda le autorizzazioni minime necessarie affinché Aurora possa accedere a un bucket Amazon S3 per tuo conto. Per consentire ad Aurora l'accesso a tutti i bucket Amazon S3, puoi saltare questi passaggi e utilizzare la policy IAM predefinita `AmazonS3ReadOnlyAccess` o `AmazonS3FullAccess` invece di crearne una nuova.

Per creare una policy IAM per consentire l'accesso alle risorse Amazon S3

1. Aprire la console di gestione IAM.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Seleziona Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo) selezionare Choose a service (Scegli un servizio) e quindi S3.

5. Per Actions (Operazioni), scegliere Expand all (Espandi tutto), quindi scegliere le autorizzazioni del bucket e dell'oggetto necessarie per la policy IAM.

   Le autorizzazioni dell'oggetto si riferiscono alle operazioni sugli oggetti in Amazon S3 e devono essere concesse per gli oggetti presenti nel bucket e non per il bucket stesso. Per ulteriori informazioni sulle autorizzazioni per le operazioni degli oggetti in Amazon S3, consulta Autorizzazioni per operazioni relative agli oggetti.
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN) per bucket.
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), fornire i dettagli sulla risorsa e scegliere Add (Aggiungi).

   Specificare il bucket Amazon S3 a cui consentire l'accesso. Ad esempio, per consentire ad Aurora di accedere al bucket Amazon S3 denominato *`amzn-s3-demo-bucket`*, impostare il valore Amazon Resource Name (ARN) su `arn:aws:s3:::`*`amzn-s3-demo-bucket`*.
8. Se la risorsa object (oggetto) è elencata, scegliere Add ARN (Aggiungi ARN) per object (oggetto).
9. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), fornire i dettagli sulla risorsa.

   Per il bucket Amazon S3 specificare il bucket Amazon S3 a cui consentire l'accesso. Per l'oggetto, è anche possibile scegliere Any (Qualsiasi) per concedere le autorizzazioni per qualsiasi oggetto nel bucket.

   > **Note**
   >
   > È possibile impostare Amazon Resource Name (ARN) su un valore dell'ARN più specifico in modo da consentire ad Aurora di accedere solo a specifici file o cartelle presenti in un bucket Amazon S3. Per ulteriori informazioni su come definire una policy di accesso per Amazon S3, consulta Gestione delle autorizzazioni di accesso alle risorse Amazon S3.
10. (Facoltativo) Scegliere Add ARN (Aggiungi ARN) affinché il bucket aggiunga un altro bucket Amazon S3 alla policy e ripetere i passaggi precedenti per il bucket.

    > **Note**
    >
    > È possibile ripetere questa operazione per aggiungere le istruzioni di autorizzazione del bucket corrispondenti alla policy per ciascun bucket Amazon S3 a cui Aurora deve

accedere. Facoltativamente, è anche possibile concedere l'accesso a tutti i bucket e gli oggetti in Amazon S3.

11. Scegliere Review policy (Esamina policy).
12. Per Name (Nome), immettere un nome per la policy IAM, ad esempio `AllowAuroraToExampleBucket`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
13. Seleziona Create Policy (Crea policy).
14. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di una policy IAM per l'accesso alle risorseAWS Lambda

È possibile seguire i passaggi seguenti per creare una policy IAM che concede le autorizzazioni minime necessarie affinché Aurora possa richiamare una funzione AWS Lambda per tuo conto.

La seguente policy aggiunge le autorizzazioni richieste da Aurora per richiamare la funzione AWS Lambda per tuo conto.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAuroraToExampleFunction",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource":
 "arn:aws:lambda:<region>:<123456789012>:function:<example_function>"
        }
    ]
}
```

È possibile seguire i passaggi seguenti per creare una policy IAM che concede le autorizzazioni minime necessarie affinché Aurora possa richiamare una funzione AWS Lambda per tuo conto. Per consentire ad Aurora di richiamare tutte le funzioni AWS Lambda, puoi saltare questi passaggi e utilizzare la policy `AWSLambdaRole` predefinita invece di crearne una nuova.

Per creare una policy IAM per consentire la chiamata delle funzioni AWS Lambda

1. Aprire la console IAM.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo), selezionare Choose a service (Scegli un servizio) e scegliere Lambda.
5. Per Actions (Operazioni), scegliere Expand all (Espandi tutto), quindi scegliere le autorizzazioni AWS Lambda necessarie per la policy IAM.

   Verificare che `InvokeFunction` sia selezionato. È l'autorizzazione minima richiesta per consentire ad Amazon Aurora di richiamare una funzione AWS Lambda
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN) per function (funzione).
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), fornire i dettagli sulla risorsa.

   Specificare la funzione Lambda a cui consentire l'accesso. Ad esempio, per consentire ad Aurora l'accesso a una funzione Lambda denominata `example_function`, impostare il valore dell'ARN su `arn:aws:lambda:::function:example_function`.

   Per ulteriori informazioni su come definire una policy di accesso per AWS Lambda, consulta Autenticazione e controllo degli accessi per AWS Lambda.
8. Facoltativamente, scegliere Add additional permissions (Aggiungi autorizzazioni aggiuntive) per aggiungere un'altra funzione AWS Lambda alla policy e ripetere i passaggi precedenti per la funzione.

   > **Note**
   >
   > È possibile ripetere questa operazione per aggiungere le istruzioni di autorizzazione della funzione corrispondenti alla policy per ciascuna funzione AWS Lambda a cui Aurora deve accedere.
9. Scegliere Review policy (Esamina policy).
10. Impostare Name (Nome) su un nome per la policy IAM, ad esempio `AllowAuroraToExampleFunction`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.

11. Seleziona Create Policy (Crea policy).
12. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di una policy IAM per l'accesso alle risorse CloudWatch Logs

Aurora può accedere a CloudWatch Logs per esportare i dati dei log di controllo da un cluster DB Aurora. Tuttavia, è necessario innanzitutto creare una policy IAM che assegni le autorizzazioni del gruppo di log e del flusso di log che consentano ad Aurora di accedere a CloudWatch Logs.

La seguente policy aggiunge le autorizzazioni che Aurora richiede per accedere ad Amazon CloudWatch Logs per tuo conto e il numero minimo di autorizzazioni necessarie per creare i gruppi di log ed esportare i dati.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*"
        },
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogGroupsAndStreams",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutRetentionPolicy",
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*"
        }
    ]
}
```

È possibile modificare gli ARN nella policy per limitare l'accesso a una regione e a un account AWS specifici.

È possibile seguire i passaggi seguenti per creare una policy IAM che conceda le autorizzazioni minime necessarie affinché Aurora possa accedere a CloudWatch Logs per tuo conto. Per consentire ad Aurora l'accesso completo a CloudWatch Logs, puoi saltare questi passaggi e utilizzare la policy IAM predefinita `CloudWatchLogsFullAccess` invece di crearne una nuova. Per ulteriori informazioni, consulta Utilizzo delle policy basate su identità (policy IAM) per CloudWatch Logs nella Guida per l'utente di Amazon CloudWatch.

Per creare una policy IAM per consentire l'accesso alle risorse CloudWatch Logs

1. Aprire la console IAM.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo), selezionare Choose a service (Scegli un servizio) e scegliere CloudWatch Logs (Log di CloudWatch).
5. Per Actions (Operazioni), scegliere Expand all (Espandi tutto) (a destra), quindi scegliere le autorizzazioni Amazon CloudWatch Logs necessarie per la policy IAM.

   Verificare che le seguenti autorizzazioni siano selezionate:

   - `CreateLogGroup`
   - `CreateLogStream`
   - `DescribeLogStreams`
   - `GetLogEvents`
   - `PutLogEvents`
   - `PutRetentionPolicy`
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN) per log-group (gruppo di log).
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), immettere i seguenti valori:

   - Regione: una regione AWS o *
   - Account – Un numero di account o *
   - Log Group Name (Nome gruppo di log – `/aws/rds/*`
8. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), scegliere Add (Aggiungi).

9. Scegliere Add ARN (Aggiungi ARN) per log-stream (flusso di log).
10. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), immettere i seguenti valori:
    - Regione: una regione AWS o *
    - Account – Un numero di account o *
    - Log Group Name (Nome gruppo di log – `/aws/rds/*`
    - Log Stream Name (Nome flusso di log – *
11. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), scegliere Add (Aggiungi).
12. Scegliere Review policy (Esamina policy).
13. Impostare Name (Nome) su un nome per la policy IAM, ad esempio `AmazonRDSCloudWatchLogs`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
14. Seleziona Create Policy (Crea policy).
15. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di una policy IAM per l'accesso alle risorseAWS KMS

Aurora può accedere alle AWS KMS keys utilizzate per la crittografia dei backup dei database. Tuttavia, è necessario innanzitutto creare una policy IAM che assegni le autorizzazioni che consentano ad Aurora di accedere alle chiavi KMS.

La seguente policy aggiunge le autorizzazioni che Aurora richiede per accedere alle chiavi KMS per tuo conto.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:<region>:<123456789012>:key/<key-ID>"
        }
    ]
```

```
}
```

È possibile seguire i passaggi seguenti per creare una policy IAM che concede le autorizzazioni minime necessarie affinché Aurora possa accedere alle chiavi KMS per tuo conto.

Per creare una policy IAM per consentire l'accesso alle chiavi KMS

1. Aprire la console IAM.
2. Nel pannello di navigazione, selezionare Policies (Policy).
3. Scegli Create Policy (Crea policy).
4. Nella scheda Visual editor (Editor visivo), selezionare Choose a service (Scegli un servizio) e scegliere KMS.
5. Per Actions (Operazioni), scegliere Write (Scrivi) e selezionare Decrypt (Decrittografa).
6. Scegliere Resources (Risorse) e selezionare Add ARN (Aggiungi ARN).
7. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), immettere i seguenti valori:
   - Regione: immettere la regione AWS, ad esempio `us-west-2`.
   - Account – Specificare il numero dell'account utente.
   - Nome flusso di registro: digitare l'identificatore della chiave KMS.
8. Nella finestra di dialogo Add ARN(s) (Aggiungi ARN), scegliere Add (Aggiungi).
9. Scegliere Review policy (Esamina policy).
10. Impostare Name (Nome) su un nome per la policy IAM, ad esempio `AmazonRDSKMSKey`. Questo nome viene utilizzato quando si crea un ruolo IAM e lo si associa al cluster DB Aurora. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
11. Seleziona Create Policy (Crea policy).
12. Completa le fasi descritte in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.

Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS

Dopo aver creato una policy IAM per consentire ad Aurora di accedere alle risorse AWS, è necessario creare un ruolo IAM e collegare la policy IAM al nuovo ruolo IAM.

Per creare un ruolo IAM che consenta al cluster Amazon RDS di comunicare con altri servizi AWS per tuo conto, completa la procedura riportata di seguito.

Come creare un ruolo IAM per consentire ad Amazon RDS di accedere ai servizi AWS

1. Aprire la console IAM.
2. Nel pannello di navigazione, selezionare Ruoli.
3. Selezionare Create role (Crea ruolo).
4. In Servizio AWS, scegli RDS.
5. In Select your use case (Seleziona caso di utilizzo), selezionare RDS – Add Role to Database (RDS – Aggiungi ruolo al database).
6. Seleziona Next (Successivo).
7. Nella pagina Permissions policies (Policy di autorizzazione), immetti il nome della policy nel campo Search (Cerca).
8. Quando viene visualizzata nell'elenco, selezionare la policy definita in precedenza utilizzando le istruzioni presenti in una delle seguenti sezioni:
   - Creazione di una policy IAM per l'accesso alle risorse Amazon S3
   - Creazione di una policy IAM per l'accesso alle risorseAWS Lambda
   - Creazione di una policy IAM per l'accesso alle risorse CloudWatch Logs
   - Creazione di una policy IAM per l'accesso alle risorseAWS KMS
9. Seleziona Next (Successivo).
10. IN Role name (Nome ruolo), digitare un nome per il ruolo IAM, ad esempio `RDSLoadFromS3`. È anche possibile aggiungere un valore Description (Descrizione) facoltativo.
11. Selezionare Create Role (Crea ruolo).
12. Completa le fasi descritte in Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.

Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL

Per consentire agli utenti del database in un cluster database Amazon Aurora di accedere ad altri servizi AWS, è necessario associare il ruolo IAM creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster database. È anche possibile configurare AWS per creare un nuovo ruolo IAM associando direttamente il servizio.

> **Note**
>
> Non è possibile associare un ruolo IAM a un cluster DB Aurora Serverless v1. Per ulteriori informazioni, consulta Usare Amazon Aurora Serverless v1.
> Puoi associare un ruolo IAM a un cluster database Aurora Serverless v2.

Per associare un ruolo IAM a un cluster DB è necessario effettuare due operazioni:

1. Aggiungere il ruolo all'elenco dei ruoli associati del cluster di database tramite la console RDS, il comando add-role-to-db-cluster della AWS CLI o l'operazione API RDS AddRoleToDBCluster.

   Puoi aggiungere un massimo di cinque ruoli IAM per ogni cluster DB Aurora.

2. Impostare l'ARN del ruolo IAM associato nel parametro a livello di cluster per il servizio AWS interessato.

   La seguente tabella elenca i nomi dei parametri a livello di cluster per i ruoli IAM utilizzati per accedere ad altri servizi AWS.

Parametro a livello di cluster	Descrizione	
`aws_default_lambda_role`	Viene utilizzato quando si richiama una funzione Lambda dal cluster DB.	
`aws_default_logs_role`	Questo parametro non è più richiesto per esportare i dati del log dal cluster database ad Amazon CloudWatch Logs. Aurora MySQL ora utilizza un ruolo collegato ai servizi per le autorizzazioni necessarie. Per ulteriori informazioni sui ruoli collegati al servizio, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.	
`aws_default_s3_role`	Viene utilizzato quando si richiama l'istruzione `LOAD DATA FROM S3`, `LOAD XML FROM S3` o `SELECT INTO OUTFILE S3` dal cluster DB.	

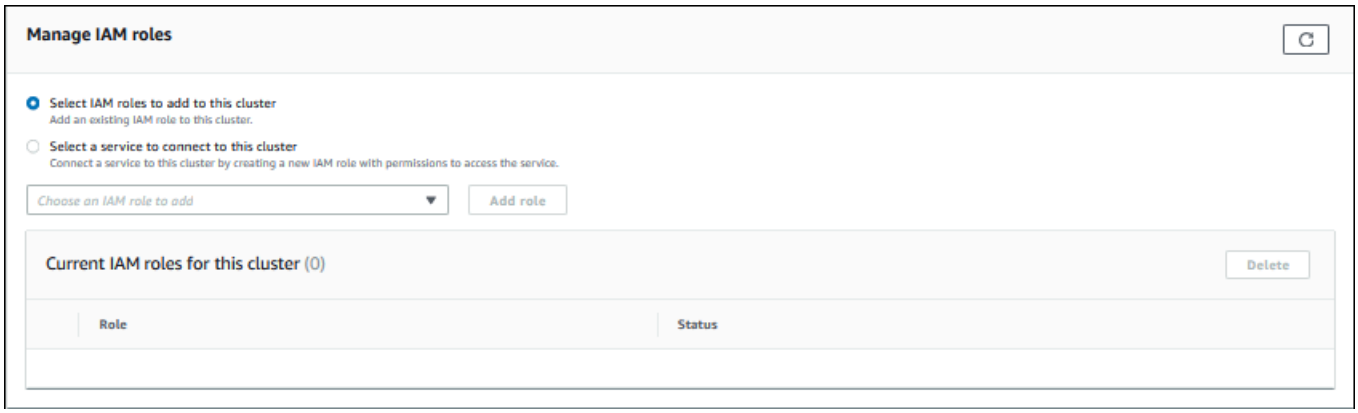
Parametro a livello di cluster	Descrizione	
	In Aurora MySQL versione 2, il ruolo IAM specificato in questo parametro viene utilizzato se un ruolo IAM non è specificato per `aurora_load_from_s3_role` o `aurora_select_into_s3_role` per l'istruzione appropriata. In Aurora MySQL versione 3, il ruolo IAM specificato per questo parametro è sempre utilizzato.	
`aurora_load_from_s3_role`	Viene utilizzato quando si richiama l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3` dal cluster DB. Se un ruolo IAM non è specificato per questo parametro viene usato il ruolo IAM specificato in `aws_default_s3_role` . In Aurora MySQL versione 3, questo parametro non è disponibile.	
`aurora_select_into_s3_role`	Viene utilizzato quando si richiama l'istruzione `SELECT INTO OUTFILE S3` dal cluster DB. Se un ruolo IAM non è specificato per questo parametro viene usato il ruolo IAM specificato in `aws_default_s3_role` . In Aurora MySQL versione 3, questo parametro non è disponibile.	

Per associare un ruolo IAM che consenta al cluster Amazon RDS di comunicare con altri servizi AWS per tuo conto, completa la procedura riportata di seguito.

Console

Per associare un ruolo IAM a un cluster DB Aurora utilizzando la console

1. Apri la console RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Databases (Database).
3. Scegliere il nome del cluster DB Aurora da associare a un ruolo IAM per mostrarne i dettagli.
4. Nella scheda Connettività e sicurezza, nella sezione Gestisci ruoli IAM, esegui una delle seguenti operazioni:
   - Seleziona ruoli IAM da aggiungere a questo cluster (impostazione predefinita)
   - Seleziona un servizio per connetterti a questo cluster



5. Per utilizzare un ruolo IAM esistente, selezionalo dal menu, quindi scegli Aggiungi ruolo.

   Se l'aggiunta del ruolo ha esito positivo, il suo stato viene visualizzato come `Pending`, quindi `Available`.

6. Per connettere direttamente un servizio:

   a. Scegli Seleziona un servizio per connetterti a questo cluster.

   b. Scegli il servizio dal menu, quindi seleziona Connetti un servizio.

   c. Per Connetti il cluster al ***`Service Name`***, immetti il nome della risorsa Amazon (ARN) da utilizzare per connettersi al servizio, quindi scegli Connetti un servizio.

   AWS crea un nuovo ruolo IAM per la connessione al servizio. Il suo stato viene visualizzato come `Pending`, quindi `Available`.

7. (Facoltativo) Per arrestare l'associazione di un ruolo IAM a un cluster database e rimuovere l'autorizzazione correlata, scegli il ruolo e quindi seleziona Elimina.

Per impostare il parametro a livello di cluster per il ruolo IAM associato

1. Nel riquadro di navigazione della console di RDS selezionare Parameter groups (Gruppi di parametri).
2. Se si sta già utilizzando un gruppo di parametri database personalizzato, è possibile selezionare quel gruppo per usarlo invece di creare un nuovo gruppo di parametri del cluster DB. Se si sta utilizzando il gruppo di parametri del cluster DB predefinito, creare un nuovo gruppo di parametri del cluster DB, come descritto nei passaggi seguenti:

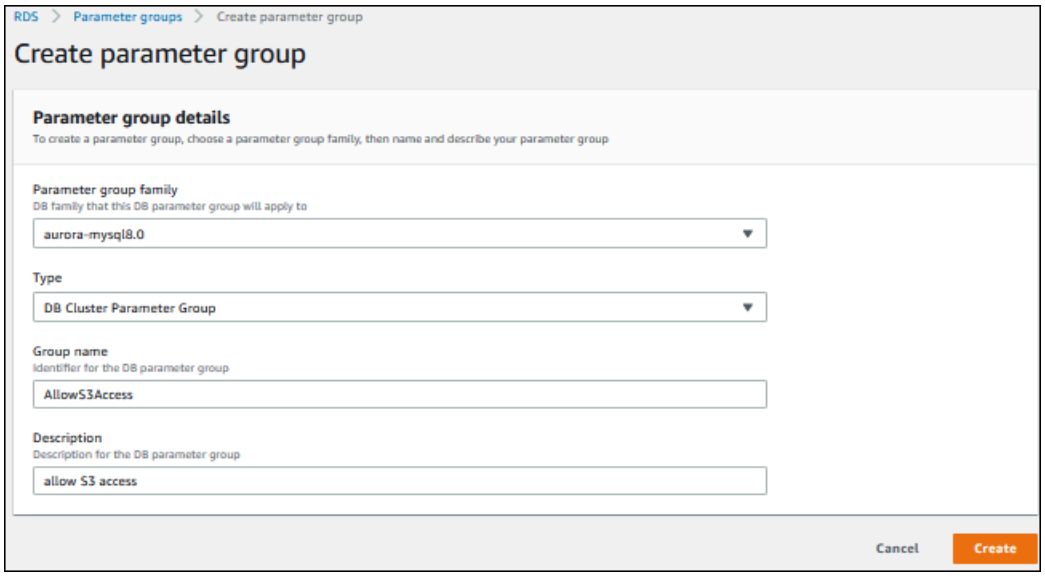
   a. Scegliere Create parameter group (Crea gruppo di parametri).

   b. Per Famiglia del gruppo di parametri, scegli `aurora-mysql8.0` per un cluster di database compatibile con Aurora MySQL 8.0 o scegli `aurora-mysql5.7` per un cluster di database compatibile con Aurora MySQL 5.7.

   c. Per Type (Tipo), scegliere DB Cluster Parameter Group (Gruppo di parametri del cluster DB).

   d. Per Group name (Nome gruppo), digitare il nome del nuovo gruppo di parametri del cluster DB.

   e. Per Description (Descrizione), digitare una descrizione per il nuovo gruppo di parametri del cluster DB.



   f. Scegliere Create (Crea).

3. Nella pagina Parameter groups (Gruppi di parametri), selezionare il gruppo di parametri del cluster DB e scegliere Edit (Modifica) per Parameter group actions (Operazioni del gruppo di parametri).
4. Imposta i parametri appropriati a livello di cluster sui valori ARN del ruolo IAM correlato.

   Ad esempio, è possibile impostare il parametro `aws_default_s3_role` su `arn:aws:iam::123456789012:role/AllowS3Access`.
5. Seleziona Save changes (Salva modifiche).
6. Per modificare il gruppo di parametri del cluster di database per il cluster di database, completare le seguenti fasi:

   a. Scegliere Databases (Database), quindi scegliere il cluster di database Aurora.
   b. Scegliere Modify (Modifica).
   c. Scorrere fino a Database options (Opzioni database) e impostare DB cluster parameter group (Gruppo di parametri del cluster di database) sul gruppo di parametri del cluster di database.
   d. Scegli Continue (Continua).
   e. Controllare le modifiche e scegliere Apply immediately (Applica immediatamente).
   f. Scegliere Modify cluster (Modifica cluster).
   g. Scegliere Databases (Database) e scegliere l'istanza primaria per il cluster di database.
   h. In Actions (Operazioni), scegliere Reboot (Riavvia).

      Al riavvio dell'istanza, il ruolo IAM viene associato al cluster DB.

      Per ulteriori informazioni sui gruppi di parametri del cluster, consulta Parametri di configurazione Aurora MySQL.

CLI

Per associare un ruolo IAM a un cluster DB utilizzando l AWS CLI

1. Chiamare il comando `add-role-to-db-cluster` dell'AWS CLI per aggiungere gli ARN per i ruoli IAM al cluster DB, come mostrato di seguito.

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraS3Role
```

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraLambdaRole
```

2. Se si sta utilizzando il gruppo di parametri del cluster DB predefinito, creare un nuovo gruppo di parametri del cluster DB. Se si sta già utilizzando un gruppo di parametri database personalizzato, è possibile usare quel gruppo invece di creare un nuovo gruppo di parametri del cluster DB.

   Per creare un nuovo gruppo di parametri del cluster DB, chiamare il comando `create-db-cluster-parameter-group` dell'AWS CLI, come mostrato di seguito.

   ```
   PROMPT> aws rds create-db-cluster-parameter-group  --db-cluster-parameter-group-
   name AllowAWSAccess \
        --db-parameter-group-family aurora5.7 --description "Allow access to Amazon S3
    and AWS Lambda"
   ```

   Per un cluster DB compatibile con Aurora MySQL 5.7, specificare `aurora-mysql5.7` per `--db-parameter-group-family`. Per un cluster database compatibile con Aurora MySQL 8.0, specificare `aurora-mysql8.0` per `--db-parameter-group-family`.

3. Impostare i parametri o i parametri appropriati a livello di cluster e i relativi valori ARN del ruolo IAM nel gruppo di parametri del cluster DB, come illustrato di seguito.

   ```
   PROMPT> aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name
    AllowAWSAccess \
       --parameters
    "ParameterName=aws_default_s3_role,ParameterValue=arn:aws:iam::123456789012:role/
   AllowAuroraS3Role,method=pending-reboot" \
       --parameters
    "ParameterName=aws_default_lambda_role,ParameterValue=arn:aws:iam::123456789012:role/
   AllowAuroraLambdaRole,method=pending-reboot"
   ```

4. Modificare il cluster DB per usare il nuovo gruppo di parametri del cluster DB e riavviare il cluster, come mostrato di seguito.

   ```
   PROMPT> aws rds modify-db-cluster --db-cluster-identifier my-cluster --db-cluster-
   parameter-group-name AllowAWSAccess
   PROMPT> aws rds reboot-db-instance --db-instance-identifier my-cluster-primary
   ```

   Al riavvio dell'istanza, i ruoli IAM vengono associati al cluster DB.

Per ulteriori informazioni sui gruppi di parametri del cluster, consulta Parametri di configurazione Aurora MySQL.

Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS

Per utilizzare determinati altri AWS servizi con Amazon Aurora, la configurazione di rete del cluster Aurora DB deve consentire le connessioni in uscita agli endpoint per tali servizi. Le seguenti operazioni richiedono questa configurazione di rete.

- Invocare funzioni. AWS Lambda Per informazioni su questa funzionalità, vedere Invocare una funzione Lambda con una funzione Aurora My native SQL.
- Accesso ai file da Amazon S3. Per informazioni su questa funzionalità, vedere Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3 e Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3.
- Accesso agli AWS KMS endpoint. AWS KMS l'accesso è necessario per utilizzare i flussi di attività del database con SQL Aurora My. Per informazioni su questa funzionalità, vedere Monitoraggio di Amazon Aurora tramite i flussi di attività del database.
- Accesso agli endpoint SageMaker AI. SageMaker L'accesso all'intelligenza artificiale è necessario per utilizzare l'apprendimento automatico dell' SageMaker intelligenza artificiale con Aurora My. SQL Per informazioni su questa funzionalità, vedere Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.

Quando non è possibile connettersi a un endpoint del servizio, Aurora restituisce il seguente messaggio di errore:

```
ERROR 1871 (HY000): S3 API returned error: Network Connection
```

```
ERROR 1873 (HY000): Lambda API returned error: Network Connection. Unable to connect to
 endpoint
```

```
ERROR 1815 (HY000): Internal error: Unable to initialize S3Stream
```

Per i flussi di attività del database che utilizzano Aurora SQL My, il flusso di attività smette di funzionare se il cluster DB non può accedere all' AWS KMS endpoint. Aurora ti informa di questo problema utilizzando Events. RDS

Se riscontri questi messaggi mentre usi i AWS servizi corrispondenti, controlla se il tuo cluster Aurora DB è pubblico o privato. Se il cluster DB Aurora è privato, devi configurarlo per abilitare le connessioni.

Affinché un cluster DB Aurora sia pubblico, deve essere contrassegnato come accessibile pubblicamente. Se esamini i dettagli del cluster DB in, se questo è il caso AWS Management Console, l'opzione Accessibile pubblicamente è Sì. Il cluster DB deve inoltre trovarsi in una sottorete VPC pubblica Amazon. Per ulteriori informazioni sulle istanze database accessibili pubblicamente, consulta Uso di un cluster database in un VPC. Per ulteriori informazioni sulle VPC sottoreti pubbliche di Amazon, consulta Your VPC and subnet.

Se il cluster Aurora DB non è accessibile pubblicamente e si trova in una sottorete VPC pubblica, è privato. È possibile che si disponga di un cluster DB privato e si desideri utilizzare una delle funzionalità che richiedono questa configurazione di rete. In tal caso, configura il cluster in modo che possa connettersi agli indirizzi Internet tramite Network Address Translation ()NAT. In alternativa ad Amazon S3, Amazon SageMaker AI e AWS Lambda, puoi invece configurarli per avere un VPC endpoint VPC per l'altro servizio associato alla tabella di routing del cluster DB, vedi. Uso di un cluster database in un VPC Per ulteriori informazioni sulla configurazione NAT nei tuoi gatewayVPC, consulta la sezione gateway. NAT Per ulteriori informazioni sulla configurazione degli endpoint, consulta VPC endpoints. VPC Puoi anche creare un endpoint del gateway S3 per accedere al bucket S3. Per ulteriori informazioni, consulta Endpoint gateway per Amazon S3.

Potrebbe inoltre essere necessario aprire le porte temporanee per gli elenchi di controllo degli accessi alla rete (ACLs) nelle regole in uscita per il gruppo di sicurezza. VPC Per ulteriori informazioni sulle porte temporanee per la reteACLs, consulta Porte temporanee nella Guida per l'utente di Amazon Virtual Private Cloud.

Argomenti correlati

- Integrazione di Aurora con altri servizi AWS
- Gestione di un cluster DB Amazon Aurora

Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3

Puoi usare l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3` per caricare i dati dai file memorizzati in un bucket Amazon S3. In Aurora MySQL, i file vengono prima archiviati sul disco

locale e quindi importati nel database. Al termine delle importazioni nel database, i file locali vengono eliminati.

> **Note**
>
> Il caricamento dei dati in una tabella dai file di testo non è supportato in Aurora Serverless v1. È supportata per Aurora Serverless v2.

Indice

- Accesso di Aurora a Amazon S3;
- Concessione dei privilegi per caricare dati in Amazon Aurora My SQL
- Specificare il percorso (URI) verso un bucket Amazon S3
- LOADDATAFROMS3
  - Sintassi
  - Parametri
  - Utilizzo di un manifest per specificare i file di dati da caricare
    - Verifica dei file caricati utilizzando la tabella aurora_s3_load_history
  - Esempi
- LOADXMLFROMS3
  - Sintassi
  - Parametri

Accesso di Aurora a Amazon S3;

Prima di poter caricare i dati da un bucket Amazon S3, devi prima concedere al cluster Aurora My SQL DB l'autorizzazione ad accedere ad Amazon S3.

Per consentire ad Aurora My di SQL accedere ad Amazon S3

1. Crea una policy AWS Identity and Access Management (IAM) che fornisca le autorizzazioni per i bucket e gli oggetti che consentono al cluster Aurora My SQL DB di accedere ad Amazon S3. Per istruzioni, consulta Creazione di una policy IAM per l'accesso alle risorse Amazon S3.

> **Note**
>
> In Aurora My SQL versione 3.05 e successive, puoi caricare oggetti crittografati utilizzando la versione gestita dal cliente. AWS KMS keys A tal fine, includi l'`kms:Decrypt`autorizzazione nella tua politica. IAM Per ulteriori informazioni, consulta Creazione di una policy IAM per l'accesso alle risorseAWS KMS.
> Non è necessaria questa autorizzazione per caricare oggetti crittografati utilizzando Chiavi gestite da AWS o chiavi gestite da Amazon S3 (SSE-S3).

2. Crea un IAM ruolo e collega la IAM policy che hai creato Creazione di una policy IAM per l'accesso alle risorse Amazon S3 al nuovo ruolo. IAM Per istruzioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Assicurati che il cluster DB stia utilizzando un gruppo di parametri del cluster DB personalizzato.

   Per ulteriori informazioni sulla creazione di un gruppo di parametri del cluster DB, consulta Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.
4. Per Aurora My SQL versione 2, imposta il parametro del cluster `aurora_load_from_s3_role` o `aws_default_s3_role` DB su Amazon Resource Name (ARN) del nuovo IAM ruolo. Se non è specificato un IAM ruolo per`aurora_load_from_s3_role`, Aurora utilizza il IAM ruolo specificato in. `aws_default_s3_role`

   Per Aurora My SQL versione 3, usa. `aws_default_s3_role`

   Se il cluster fa parte di un database globale Aurora, imposta questo parametro per ogni cluster Aurora nel database globale. Sebbene solo il cluster primario in un database globale Aurora può caricare i dati, un altro cluster potrebbe essere promosso dal meccanismo di failover e diventare il cluster primario.

   Per ulteriori informazioni sui parametri del cluster DB, vedi Parametri dell'istanza database e del cluster database di Amazon Aurora.
5. Per consentire agli utenti del database in un cluster Aurora My SQL DB di accedere ad Amazon S3, associa il ruolo che hai creato Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster DB. Per un database globale Aurora, associa il ruolo a ogni cluster Aurora nel database globale. Per informazioni sull'associazione di un IAM ruolo a un cluster DB, consulta. Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL

6. Configura il tuo cluster Aurora My SQL DB per consentire le connessioni in uscita ad Amazon S3. Per istruzioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

   Se il cluster DB non è accessibile pubblicamente e si trova in una sottorete VPC pubblica, è privato. Puoi creare un endpoint del gateway S3 per accedere al bucket S3. Per ulteriori informazioni, consulta Endpoint gateway per Amazon S3.

   Per un database globale Aurora, abilita le connessioni in uscita per ogni cluster Aurora nel database globale.

Concessione dei privilegi per caricare dati in Amazon Aurora My SQL

L'utente del database che invia l'istruzione `LOAD DATA FROM S3` o `LOAD XML FROM S3` deve avere un ruolo o privilegio specifico per rilasciare una delle due istruzioni. In Aurora My SQL version 3, concedi il `AWS_LOAD_S3_ACCESS` ruolo. In Aurora My SQL version 2, concedi il `LOAD FROM S3` privilegio. All'utente amministrativo per un cluster di database è concesso il ruolo o il privilegio appropriato per impostazione predefinita. Puoi concedere il privilegio a un altro utente usando le seguenti istruzioni.

Usa la seguente dichiarazione per Aurora My SQL versione 3:

```
GRANT AWS_LOAD_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Quando usi la tecnica del ruolo in Aurora My SQL versione 3, puoi anche attivare il ruolo utilizzando l'istruzione `SET ROLE role_name` or`SET ROLE ALL`. Se non conosci il sistema di ruoli My SQL 8.0, puoi saperne di più in. Privilegio basato sui ruoli Per maggiori dettagli, consulta Usare i ruoli nel My SQL Reference Manual.
> Questo vale solo per la sessione attiva corrente. Quando ti riconnetti, devi eseguire nuovamente l'`SET ROLE`istruzione per concedere i privilegi. Per ulteriori informazioni, vedere l'SETROLEistruzione nel My SQL Reference Manual.
> Puoi utilizzare il parametro `activate_all_roles_on_login` del cluster di database per attivare automaticamente tutti i ruoli quando un utente si connette a un'istanza database. Quando questo parametro è impostato, in genere non è necessario chiamare l'`SET ROLE`istruzione in modo esplicito per attivare un ruolo. Per ulteriori informazioni, vedere activate_all_roles_on_login nel My Reference Manual. SQL

Tuttavia, è necessario chiamare `SET ROLE ALL` esplicitamente all'inizio di una stored procedure per attivare il ruolo, quando la stored procedure viene chiamata da un altro utente.

Usa la seguente dichiarazione per Aurora My SQL versione 2:

```
GRANT LOAD FROM S3 ON *.* TO 'user'@'domain-or-ip-address'
```

Il `AWS_LOAD_S3_ACCESS` ruolo e il `LOAD FROM S3` privilegio sono specifici di Amazon Aurora e non sono disponibili per i database SQL My esterni RDS o per le istanze SQL My DB. Se hai impostato la replica tra un cluster Aurora DB come origine di replica e un database SQL My come client di replica, l'istruzione per il ruolo o `GRANT` il privilegio causa l'interruzione della replica con un errore. Puoi ignorare l'errore in modo sicuro per riprendere la replica. Per ignorare l'errore su un'istanza RDS for MySQL, utilizzate la procedura mysql_rds_skip_repl_error. Per ignorare l'errore su un SQL database My esterno, utilizzate la variabile di sistema slave_skip_errors (Aurora My SQL versione 2) o la variabile di sistema replica_skip_errors (Aurora My versione 3). SQL

> **Note**
>
> L'utente del database deve disporre dei privilegi per il database in cui carica i `INSERT` dati.

Specificare il percorso (URI) verso un bucket Amazon S3

La sintassi per specificare il percorso (URI) dei file archiviati in un bucket Amazon S3 è la seguente.

```
s3-region://amzn-s3-demo-bucket/file-name-or-prefix
```

Il percorso include i seguenti valori:

- `region`(opzionale): la AWS regione che contiene il bucket Amazon S3 da cui caricare. Questo valore è facoltativo. Se non specifichi un valore per `region`, Aurora carica il file da Amazon S3 nella stessa regione del cluster DB.
- `bucket-name` – Il nome del bucket Amazon S3 che contiene i dati da caricare. Sono supportati i prefissi degli oggetti che identificano un percorso di cartella virtuale.
- `file-name-or-prefix`— Il nome del file o del file di testo di Amazon S3 o un prefisso che identifica uno o più XML file di testo da caricare. XML È anche possibile specificare un file manifest che identifica uno o più file di testo da caricare. Per ulteriori informazioni sull'utilizzo di un file

manifest per caricare file di testo da Amazon S3, consulta Utilizzo di un manifest per specificare i file di dati da caricare.

Per copiare i file URI for in un bucket S3

1. Accedi AWS Management Console e apri la console Amazon S3 all'indirizzo. https://console.aws.amazon.com/s3/
2. Nel pannello di navigazione, scegli Bucket, quindi scegli il bucket da URI copiare.
3. Seleziona il prefisso o il file che desideri caricare da S3.
4. Scegli Copia S3. URI

LOADDATAFROMS3

È possibile utilizzare l'`LOAD DATA FROM S3`istruzione per caricare dati da qualsiasi formato di file di testo supportato dall'SQLLOADDATAINFILEistruzione My, ad esempio dati di testo delimitati da virgole. I file compressi non sono supportati.

> **Note**
>
> Assicurati che il tuo cluster Aurora My SQL DB consenta connessioni in uscita a S3. Per ulteriori informazioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

Sintassi

```
LOAD DATA [FROM] S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
```

```
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

> **Note**
>
> In Aurora My SQL versione 3.05 e successive, la parola chiave `FROM` è facoltativa.

Parametri

L'istruzione `LOAD DATA FROM S3` utilizza i seguenti parametri obbligatori e facoltativi. Puoi trovare maggiori dettagli su alcuni di questi parametri in LOADDATAStatement nella documentazione MySQL.

FILE | PREFIX | MANIFEST

Indica se caricare i dati da un singolo file, da tutti i file che corrispondono a un determinato prefisso o da tutti i file in un manifesto specificato. `FILE` è il valore predefinito.

S3- URI

Speciifica il caricamento URI di un file di testo o manifesto o il prefisso Amazon S3 da utilizzare. Specificare l'URIutilizzo della sintassi descritta in. Specificare il percorso (URI) verso un bucket Amazon S3

REPLACE | IGNORE

Determina quale azione intraprendere se una riga di input ha gli stessi valori chiave univoci di una riga esistente nella tabella del database.

- Specifica `REPLACE` se la riga di input deve sostituire la riga esistente nella tabella.
- Specifica `IGNORE` se la riga di input deve essere scartata.

INTO TABLE

Identifica il nome della tabella del database in cui caricare le righe di input.

PARTITION

Richiede che tutte le righe di input vengano inserite nelle partizioni identificate dall'elenco specificato di nomi di partizione separati da virgola. Se una riga di input non può essere inserita in una delle partizioni specificate, l'istruzione non riesce e viene restituito un errore.

CHARACTER SET

Identifica il set di caratteri dei dati nel file di input.

FIELDS | COLUMNS

Identifica il modo in cui i campi o le colonne nel file di input sono delimitati. I campi sono delimitati da tabulazioni per impostazione predefinita.

LINES

Identifica il modo in cui le righe nel file di input sono delimitati. Le linee sono delimitate da un carattere di nuova riga (`'\n'`) per impostazione predefinita.

IGNORE ***number*** LINES | ROWS

Specifica di ignorare un determinato numero di righe all'inizio del file di input. Ad esempio, è possibile utilizzare `IGNORE 1 LINES` per ignorare la riga di intestazione iniziale contenente i nomi di colonna o `IGNORE 2 ROWS` per ignorare le prime due righe di dati nel file di input. Se si utilizza anche `PREFIX`, `IGNORE` ignora un numero specifico di righe all'inizio del primo file di input.

col_name_or_user_var, ...

Specifica un elenco separato da virgola di uno o più nomi di colonne o variabili utente che identificano quali colonne caricare in base al nome. Il nome di una variabile utente utilizzata per questo scopo deve corrispondere al nome di un elemento del file di testo, preceduto da @. Puoi utilizzare le variabili utente per memorizzare i valori dei campi corrispondenti per un successivo riutilizzo.

Ad esempio, la seguente istruzione carica la prima colonna dal file di input nella prima colonna di `table1` e imposta il valore della colonna `table_column2` in `table1` sul valore di input della seconda colonna diviso per 100.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, @var1)
    SET table_column2 = @var1/100;
```

SET

Specifica l'elenco delle operazioni di assegnazione separate da virgole che imposta i valori delle colonne della tabella sui valori non inclusi nel file di input.

Ad esempio, la seguente istruzione imposta le prime due colonne di `table1` sui valori delle prime due colonne del file di input e imposta il valore della colonna `column3` in `table1` sul timestamp corrente.

```
LOAD DATA FROM S3  's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, column2)
    SET column3 = CURRENT_TIMESTAMP;
```

È possibile utilizzare le sottoquery nella parte destra delle assegnazioni `SET`. Per una sottoquery che restituisce un valore da assegnare a una colonna, puoi utilizzare solo una sottoquery scalare. Inoltre, non è possibile utilizzare una sottoquery per selezionare dalla tabella che viene caricata.

Non puoi usare la parola chiave `LOCAL` dell'istruzione `LOAD DATA FROM S3` se stai caricando i dati da un bucket Amazon S3.

Utilizzo di un manifest per specificare i file di dati da caricare

È possibile utilizzare l'`LOAD DATA FROM S3`istruzione con la `MANIFEST` parola chiave per specificare un file manifest in JSON formato che elenchi i file di testo da caricare in una tabella nel cluster DB.

JSONLo schema seguente descrive il formato e il contenuto di un file manifest.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "Aurora_LoadFromS3_Manifest",
    "properties": {
        "entries": {
            "additionalItems": false,
            "id": "/properties/entries",
            "items": {
                "additionalProperties": false,
                "id": "/properties/entries/items",
                "properties": {
                    "mandatory": {
                        "default": "false",
                        "id": "/properties/entries/items/properties/mandatory",
                        "type": "boolean"
```

```
                    },
                    "url": {
                        "id": "/properties/entries/items/properties/url",
                        "maxLength": 1024,
                        "minLength": 1,
                        "type": "string"
                    }
                },
                "required": [
                    "url"
                ],
                "type": "object"
            },
            "type": "array",
            "uniqueItems": true
        }
    },
    "required": [
        "entries"
    ],
    "type": "object"
}
```

Ciascuno `url` nel manifesto deve specificare un URL nome del bucket e il percorso completo dell'oggetto per il file, non solo un prefisso. Puoi utilizzare un manifest per caricare file da diversi bucket, regioni o file che non condividono lo stesso prefisso. Se non è specificata una regione inURL, viene utilizzata la regione del cluster Aurora DB di destinazione. L'esempio seguente mostra un file manifest che carica quattro file da diversi bucket.

```
{
  "entries": [
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":true
    },
    {
      "url":"s3-us-west-2://aurora-bucket-usw2/2013-10-05-customerdata",
      "mandatory":true
    },
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":false
    },
```

```
    {
      "url":"s3://aurora-bucket/2013-10-05-customerdata"
    }
  ]
}
```

Il flag facoltativo `mandatory` specifica se `LOAD DATA FROM S3` deve restituire un errore qualora il file non venga trovato. L'impostazione predefinita del flag `mandatory` è `false`. Indipendentemente dall'impostazione di `mandatory`, `LOAD DATA FROM S3` termina se non viene trovato alcun file.

I file manifest possono avere qualsiasi estensione. L'esempio seguente esegue l'istruzione `LOAD DATA FROM S3` con manifest nell'esempio precedente, che viene denominato **`customer.manifest`**.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/customer.manifest'
    INTO TABLE CUSTOMER
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL);
```

Al termine dell'istruzione, per ogni file caricato viene scritta una voce nella tabella `aurora_s3_load_history`.

Verifica dei file caricati utilizzando la tabella aurora_s3_load_history

Ogni istruzione `LOAD DATA FROM S3` con esito positivo aggiorna la tabella `aurora_s3_load_history` nello schema `mysql` con una voce per ogni file che è stato caricato.

Dopo aver eseguito l'istruzione `LOAD DATA FROM S3`, puoi verificare quali file sono stati caricati eseguendo una query sulla tabella `aurora_s3_load_history`. Per visualizzare i file caricati da un'iterazione dell'istruzione, utilizza la `WHERE` clausola per filtrare i record su Amazon URI S3 per il file manifesto utilizzato nell'istruzione. Se hai utilizzato lo stesso file manifest di prima, filtra i risultati utilizzando il campo `timestamp`.

```
select * from mysql.aurora_s3_load_history where load_prefix = 'S3_URI';
```

La tabella seguente descrive i campi della tabella `aurora_s3_load_history`.

Campo	Descrizione
`load_prefix`	Il URI that è stato specificato nell'istruzione load. Questo URI può corrispondere a uno dei seguenti elementi: • Un singolo file di dati per un'istruzione `LOAD DATA FROM S3 FILE` • Un prefisso Amazon S3 che mappa a più file di dati per un'istruzione `LOAD DATA FROM S3 PREFIX` • Un singolo file manifest contenente i nomi dei file da caricare per un'istruzione `LOAD DATA FROM S3 MANIFEST`
`file_name`	Il nome di un file che è stato caricato in Aurora da Amazon S3 utilizzando URI l'identificativo nel campo. `load_prefix`
`version_number`	Il numero di versione del file identificato dal campo `file_name` che è stato caricato se il bucket Amazon S3 ha un numero di versione.
`bytes_loaded`	Le dimensioni del file caricato in byte.
`load_timestamp`	Timestamp relativo al momento del completamento dell'istr uzione `LOAD DATA FROM S3`.

Esempi

La seguente istruzione carica i dati da un bucket Amazon S3 che si trova nella stessa regione del cluster DB Aurora. L'istruzione legge i dati delimitati da virgole nel file contenuto in `customerdata.txt` *`amzn-s3-demo-bucket`* Bucket Amazon S3, quindi carica i dati nella tabella. `store-schema.customer-table`

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/customerdata.csv'
    INTO TABLE store-schema.customer-table
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, ADDRESS, EMAIL, PHONE);
```

La seguente istruzione carica i dati da un bucket Amazon S3 che si trova in una regione diversa rispetto al cluster DB Aurora. L'istruzione legge i dati delimitati da virgole da tutti i file che corrispondono al prefisso dell'oggetto nel `employee-data` *amzn-s3-demo-bucket* Bucket Amazon S3 nella `us-west-2` regione, quindi carica i dati nella tabella. `employees`

```
LOAD DATA FROM S3 PREFIX 's3-us-west-2://amzn-s3-demo-bucket/employee_data'
    INTO TABLE employees
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL, SALARY);
```

L'istruzione seguente carica i dati dai file specificati in un file JSON manifest denominato q1_sales.json nella tabella. `sales`

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://amzn-s3-demo-bucket1/q1_sales.json'
    INTO TABLE sales
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (MONTH, STORE, GROSS, NET);
```

LOADXMLFROMS3

Puoi utilizzare l'`LOAD XML FROM S3`istruzione per caricare dati dai XML file archiviati su un bucket Amazon S3 in uno dei tre diversi formati: XML

- I nomi di colonna come attributi di un elemento `<row>`. Il valore dell'attributo identifica il contenuto del campo della tabella.

  ```
  <row column1="value1" column2="value2" .../>
  ```

- I nomi di colonna come elementi figlio di un elemento `<row>`. Il valore dell'elemento figlio identifica il contenuto del campo della tabella.

  ```
  <row>
    <column1>value1</column1>
    <column2>value2</column2>
  </row>
  ```

- I nomi di colonna nell'attributo `name` dell'elemento `<field>` in un elemento `<row>`. Il valore dell'elemento `<field>` identifica il contenuto del campo della tabella.

```
<row>
  <field name='column1'>value1</field>
  <field name='column2'>value2</field>
</row>
```

Sintassi

```
LOAD XML FROM S3 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<element-name>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Parametri

L'istruzione `LOAD XML FROM S3` utilizza i seguenti parametri obbligatori e facoltativi. Puoi trovare maggiori dettagli su alcuni di questi parametri in LOADXMLStatement nella mia SQL documentazione.

FILE | PREFIX

Indica se caricare i dati da un singolo file o da tutti i file che corrispondono a un determinato prefisso. `FILE` è il valore predefinito.

REPLACE | IGNORE

Determina quale azione intraprendere se una riga di input ha gli stessi valori chiave univoci di una riga esistente nella tabella del database.

- Specifica `REPLACE` se la riga di input deve sostituire la riga esistente nella tabella.
- Specifica `IGNORE` per ignorare la riga di input. `IGNORE` è il valore predefinito.

INTO TABLE

Identifica il nome della tabella del database in cui caricare le righe di input.

PARTITION

Richiede che tutte le righe di input vengano inserite nelle partizioni identificate dall'elenco specificato di nomi di partizione separati da virgola. Se una riga di input non può essere inserita in una delle partizioni specificate, l'istruzione non riesce e viene restituito un errore.

CHARACTER SET

Identifica il set di caratteri dei dati nel file di input.

ROWSIDENTIFIEDDI

Identifica il nome dell'elemento che identifica una riga nel file di input. Il valore predefinito è `<row>`.

IGNORE ***number*** LINES | ROWS

Specifica di ignorare un determinato numero di righe all'inizio del file di input. Ad esempio, è possibile utilizzare `IGNORE 1 LINES` per saltare la prima riga del file di testo o `IGNORE 2 ROWS` saltare le prime due righe di dati nell'input. XML

field_name_or_user_var, ...

Specificate un elenco separato da virgole di uno o più nomi di XML elementi o variabili utente che identificano gli elementi da caricare per nome. Il nome di una variabile utente utilizzata per questo scopo deve corrispondere al nome di un elemento del XML file, con il prefisso @. Puoi utilizzare le variabili utente per memorizzare i valori dei campi corrispondenti per un successivo riutilizzo.

Ad esempio, la seguente istruzione carica la prima colonna dal file di input nella prima colonna di `table1` e imposta il valore della colonna `table_column2` in `table1` sul valore di input della seconda colonna diviso per 100.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, @var1)
   SET table_column2 = @var1/100;
```

SET

Specifica l'elenco delle operazioni di assegnazione separate da virgole che imposta i valori delle colonne della tabella sui valori non inclusi nel file di input.

Ad esempio, la seguente istruzione imposta le prime due colonne di `table1` sui valori delle prime due colonne del file di input e imposta il valore della colonna `column3` in `table1` sul timestamp corrente.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, column2)
   SET column3 = CURRENT_TIMESTAMP;
```

È possibile utilizzare le sottoquery nella parte destra delle assegnazioni `SET`. Per una sottoquery che restituisce un valore da assegnare a una colonna, puoi utilizzare solo una sottoquery scalare. Inoltre, non è possibile utilizzare una sottoquery per eseguire selezioni nella tabella caricata.

Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3

Puoi utilizzare l'`SELECT INTO OUTFILE S3`istruzione per interrogare i dati da un cluster Amazon Aurora My SQL DB e salvarli in file di testo archiviati in un bucket Amazon S3. In Aurora MySQL, i file vengono prima archiviati sul disco locale e quindi esportati su S3. Al termine delle esportazioni, i file locali vengono eliminati.

Puoi crittografare il bucket Amazon S3 utilizzando una chiave gestita Amazon S3 (-S3) o SSE (-: AWS KMS key o SSE una chiave gestita dal cliente). KMS Chiave gestita da AWS

L'`LOAD DATA FROM S3`istruzione può utilizzare i file creati dall'`SELECT INTO OUTFILE S3`istruzione per caricare dati in un cluster Aurora DB. Per ulteriori informazioni, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.

> **Note**
>
> Questa funzionalità non è supportata per i cluster di database Aurora Serverless v1. È supportata per i cluster di database Aurora Serverless v2.
> Puoi anche salvare i dati del cluster DB e i dati degli snapshot del cluster DB su Amazon S3 utilizzando AWS Management Console AWS CLI, o Amazon. RDS API Per ulteriori informazioni, consulta Esportazione dei dati del cluster database in Amazon S3 e Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Indice

- Concedere ad Aurora My SQL l'accesso ad Amazon S3
- Concessione dei privilegi per il salvataggio dei dati in Aurora My SQL
- Specifica di un percorso in un bucket Amazon S3
- Creazione di un manifest per elencare i fili di dati
- SELECTINTOOUTFILES3
  - Sintassi
  - Parametri
  - Considerazioni
  - Esempi

Concedere ad Aurora My SQL l'accesso ad Amazon S3

Prima di poter salvare i dati in un bucket Amazon S3, devi prima concedere al cluster Aurora My SQL DB l'autorizzazione ad accedere ad Amazon S3.

Per consentire ad Aurora My di SQL accedere ad Amazon S3

1. Crea una policy AWS Identity and Access Management (IAM) che fornisca le autorizzazioni per i bucket e gli oggetti che consentono al cluster Aurora My SQL DB di accedere ad Amazon S3. Per istruzioni, consulta Creazione di una policy IAM per l'accesso alle risorse Amazon S3.

   > **Note**
   >
   > In Aurora My SQL versione 3.05 e successive, puoi crittografare gli oggetti utilizzando AWS KMS chiavi gestite dal cliente. A tal fine, includi l'`kms:GenerateDataKey`autorizzazione nella tua politica. IAM Per ulteriori informazioni, consulta Creazione di una policy IAM per l'accesso alle risorseAWS KMS.
   > Non è necessaria questa autorizzazione per crittografare gli oggetti utilizzando Chiavi gestite da AWS o le chiavi gestite da Amazon S3 SSE (-S3).

2. Crea un IAM ruolo e collega la IAM policy che hai creato Creazione di una policy IAM per l'accesso alle risorse Amazon S3 al nuovo ruolo. IAM Per istruzioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Per Aurora My SQL versione 2, imposta il parametro del cluster `aurora_select_into_s3_role` o `aws_default_s3_role` DB su Amazon

Resource Name (ARN) del nuovo IAM ruolo. Se non è specificato un IAM ruolo per `aurora_select_into_s3_role`, Aurora utilizza il IAM ruolo specificato in. `aws_default_s3_role`

Per Aurora My SQL versione 3, usa. `aws_default_s3_role`

Se il cluster fa parte di un database globale Aurora, imposta questo parametro per ogni cluster Aurora nel database globale.

Per ulteriori informazioni sui parametri del cluster DB, vedi Parametri dell'istanza database e del cluster database di Amazon Aurora.

4. Per consentire agli utenti del database in un cluster Aurora My SQL DB di accedere ad Amazon S3, associa il ruolo che hai creato Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster DB.

   Per un database globale Aurora, associa il ruolo a ogni cluster Aurora nel database globale.

   Per informazioni sull'associazione di un IAM ruolo a un cluster DB, consulta. Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL

5. Configura il tuo cluster Aurora My SQL DB per consentire le connessioni in uscita ad Amazon S3. Per istruzioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

   Per un database globale Aurora, abilita le connessioni in uscita per ogni cluster Aurora nel database globale.

Concessione dei privilegi per il salvataggio dei dati in Aurora My SQL

L'utente del database che emette l'istruzione `SELECT INTO OUTFILE S3` deve avere un ruolo o un privilegio specifico. In Aurora My SQL version 3, concedi il `AWS_SELECT_S3_ACCESS` ruolo. In Aurora My SQL version 2, concedi il `SELECT INTO S3` privilegio. All'utente amministrativo per un cluster di database è concesso il ruolo o il privilegio appropriato per impostazione predefinita. Puoi concedere il privilegio a un altro utente usando le seguenti istruzioni.

Usa la seguente dichiarazione per Aurora My SQL versione 3:

```
GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Quando si utilizza la tecnica del ruolo in Aurora My SQL versione 3, è possibile attivare il ruolo anche utilizzando l'istruzione `SET ROLE role_name` or`SET ROLE ALL`. Se non conosci il sistema di ruoli My SQL 8.0, puoi saperne di più in. Privilegio basato sui ruoli Per maggiori dettagli, consulta Usare i ruoli nel My SQL Reference Manual.
> Questo vale solo per la sessione attiva corrente. Quando ti riconnetti, devi eseguire nuovamente l'`SET ROLE`istruzione per concedere i privilegi. Per ulteriori informazioni, vedere l'SETROLEistruzione nel My SQL Reference Manual.
> Puoi utilizzare il parametro `activate_all_roles_on_login` del cluster di database per attivare automaticamente tutti i ruoli quando un utente si connette a un'istanza database. Quando questo parametro è impostato, in genere non è necessario chiamare l'`SET ROLE`istruzione in modo esplicito per attivare un ruolo. Per ulteriori informazioni, vedere activate_all_roles_on_login nel My Reference Manual. SQL
> Tuttavia, è necessario chiamare `SET ROLE ALL` esplicitamente all'inizio di una stored procedure per attivare il ruolo, quando la stored procedure viene chiamata da un altro utente.

Usa la seguente dichiarazione per Aurora My SQL versione 2:

```
GRANT SELECT INTO S3 ON *.* TO 'user'@'domain-or-ip-address'
```

Il `AWS_SELECT_S3_ACCESS` ruolo e il `SELECT INTO S3` privilegio sono specifici di Amazon Aurora SQL My e non sono disponibili per i miei database RDS o per le SQL istanze SQL My DB. Se è stata impostata la replica tra un cluster Aurora SQL My DB come origine di replica e un database SQL My come client di replica, l'istruzione per il ruolo o `GRANT` il privilegio causa l'interruzione della replica con un errore. Puoi ignorare l'errore in modo sicuro per riprendere la replica. Per ignorare l'errore su un'istanza RDS for My SQL DB, usa la procedura mysql_rds_skip_repl_error. Per ignorare l'errore su un SQL database My esterno, utilizzate la variabile di sistema slave_skip_errors (Aurora My SQL versione 2) o la variabile di sistema replica_skip_errors (Aurora My versione 3). SQL

Specifica di un percorso in un bucket Amazon S3

La sintassi per specificare un percorso per archiviare i file manifest e di dati in un bucket Amazon S3 è simile a quella utilizzata nell'istruzione `LOAD DATA FROM S3 PREFIX`, come mostrato di seguito.

```
s3-region://bucket-name/file-prefix
```

Il percorso include i seguenti valori:

- `region`(opzionale): la AWS regione che contiene il bucket Amazon S3 in cui salvare i dati. Questo valore è facoltativo. Se non specifichi un valore per `region`, Aurora salva i file in Amazon S3 nella stessa regione del cluster DB.
- `bucket-name` – Il nome del bucket Amazon S3 in cui salvare i dati. Sono supportati i prefissi degli oggetti che identificano un percorso di cartella virtuale.
- `file-prefix` – Il prefisso dell'oggetto Amazon S3 che identifica i file da salvare in Amazon S3.

I file di dati creati dall'`SELECT INTO OUTFILE S3`istruzione utilizzano il seguente percorso, in cui *`00000`* rappresenta un numero intero a 5 cifre con base zero.

```
s3-region://bucket-name/file-prefix.part_00000
```

Ad esempio, supponiamo che un'istruzione `SELECT INTO OUTFILE S3` specifichi `s3-us-west-2://bucket/prefix` come percorso in cui archiviare i file di dati e crea tre file di dati. Il bucket Amazon S3 specificato contiene i seguenti file di dati.

- s3-us-west-2://bucket/prefix.part_00000
- s3-us-west-2://bucket/prefix.part_00001
- s3-us-west-2://bucket/prefix.part_00002

Creazione di un manifest per elencare i fili di dati

È possibile utilizzare l'`SELECT INTO OUTFILE S3`istruzione con l'`MANIFEST ON`opzione per creare un file manifesto in JSON formato che elenchi i file di testo creati dall'istruzione. L'`LOAD DATA FROM S3`istruzione può utilizzare il file manifest per caricare nuovamente i file di dati in un cluster Aurora My SQL DB. Per ulteriori informazioni sull'utilizzo di un manifest per caricare file di dati da Amazon S3 in un cluster Aurora My SQL DB, consulta. Utilizzo di un manifest per specificare i file di dati da caricare

I file di dati inclusi nel manifest creati dall'istruzione `SELECT INTO OUTFILE S3` sono elencati nell'ordine in cui vengono creati dall'istruzione. Ad esempio, supponiamo che un'istruzione `SELECT INTO OUTFILE S3` specifichi `s3-us-west-2://bucket/prefix` come percorso in cui archiviare i file di dati e crea tre file di dati e un file manifest. Il bucket Amazon S3 specificato contiene un file manifest denominato `s3-us-west-2://bucket/prefix.manifest` che contiene le seguenti informazioni.

```
{
  "entries": [
    {
      "url":"s3-us-west-2://bucket/prefix.part_00000"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00001"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00002"
    }
  ]
}
```

SELECTINTOOUTFILES3

Puoi usare l'istruzione `SELECT INTO OUTFILE S3` per eseguire una query sui dati da un cluster DB e salvarlo direttamente nei file di testo delimitato archiviati in un bucket Amazon S3.

I file compressi non sono supportati. I file crittografati sono supportati a partire dalla SQL versione 2.09.0 di Aurora My.

Sintassi

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
        [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
         [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
    [export_options]
```

```
    [MANIFEST {ON | OFF}]
    [OVERWRITE {ON | OFF}]
    [ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['cmk_id']}]

export_options:
    [FORMAT {CSV|TEXT} [HEADER]]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
]
```

Parametri

L'istruzione `SELECT INTO OUTFILE S3` utilizza i seguenti parametri obbligatori e facoltativi specifici per Aurora.

s3-uri

Speciifica il URI prefisso Amazon S3 da utilizzare. Utilizza la sintassi descritta in Specifica di un percorso in un bucket Amazon S3.

FORMAT {CSV|TEXT} [HEADER]

Facoltativamente, salva i dati in formato. CSV

L'`TEXT`opzione è l'impostazione predefinita e produce il formato My SQL export esistente.

L'opzione `CSV` genera valori di dati separati da virgole. Il CSV formato segue le specifiche in RFC-4180. Se si specifica la parola chiave facoltativa `HEADER`, il file di output contiene una riga di intestazione. Le etichette nella riga di intestazione corrispondono ai nomi di colonna dell'istruzione `SELECT`. È possibile utilizzare i CSV file per addestrare modelli di dati da utilizzare con i servizi AWS ML. Per ulteriori informazioni sull'utilizzo dei dati Aurora esportati AWS con i servizi ML, vedere. Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)

MANIFEST{ON |} OFF

Indica se un file manifesto viene creato in Amazon S3. Il file manifest è un file JavaScript Object Notation (JSON) che può essere utilizzato per caricare dati in un cluster Aurora DB con `LOAD`

`DATA FROM S3 MANIFEST` l'istruzione. Per ulteriori informazioni su `LOAD DATA FROM S3 MANIFEST`, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.

Se `MANIFEST ON` è specificato nella query, il file manifest viene creato in Amazon S3 dopo che tutti i file di dati sono stati creati e caricati. Il file manifest viene creato utilizzando il seguente percorso:

```
s3-region://bucket-name/file-prefix.manifest
```

Per ulteriori informazioni sul formato del contenuto del file manifest, consulta Creazione di un manifest per elencare i fili di dati.

OVERWRITE{SU |} OFF

Indica se i file esistenti nel bucket Amazon S3 specificato vengono sovrascritti. Se `OVERWRITE ON` viene specificato, i file esistenti che corrispondono al prefisso del file URI specificato in `s3-uri` vengono sovrascritti. In caso contrario, si verifica un errore.

ENCRYPTION{ON | | SSE _S3 OFF | _ ['SSEKMS*cmk_id*']}

Indica se utilizzare la crittografia lato server con le chiavi gestite di Amazon S3 SSE (-S3) AWS KMS keys o SSE (KMS-, Chiavi gestite da AWS incluse le chiavi gestite dal cliente). Le `SSE_KMS` impostazioni `SSE_S3` e sono disponibili in Aurora My SQL versione 3.05 e successive.

Puoi anche utilizzare la variabile di sessione `aurora_select_into_s3_encryption_default` anziché la clausola `ENCRYPTION`, come illustrato nell'esempio seguente. Utilizzate la SQL clausola o la variabile di sessione, ma non entrambe.

```
set session set session aurora_select_into_s3_encryption_default={ON | OFF | SSE_S3
 | SSE_KMS};
```

Le `SSE_KMS` impostazioni `SSE_S3` e sono disponibili in Aurora My SQL versione 3.05 e successive.

Impostando `aurora_select_into_s3_encryption_default` sul seguente valore:

- `OFF`: viene seguita la policy di crittografia predefinita del bucket S3. Il valore predefinito di `aurora_select_into_s3_encryption_default` è `OFF`.

- 0Noppure `SSE_S3` — L'oggetto S3 è crittografato utilizzando chiavi gestite di Amazon S3 SSE (-S3).
- `SSE_KMS`— L'oggetto S3 viene crittografato utilizzando un. AWS KMS key

  In questo caso, includi anche la variabile di sessione `aurora_s3_default_cmk_id`, ad esempio:

  ```
  set session aurora_select_into_s3_encryption_default={SSE_KMS};
  set session aurora_s3_default_cmk_id={NULL | 'cmk_id'};
  ```

  - Quando `aurora_s3_default_cmk_id` è `NULL`, l'oggetto S3 viene crittografato utilizzando una Chiave gestita da AWS.
  - Quando `aurora_s3_default_cmk_id` è una stringa `cmk_id` non vuota, l'oggetto S3 viene crittografato utilizzando una chiave gestita dal cliente.

    Il valore di `cmk_id` non può essere una stringa vuota.

Quando utilizzi il comando `SELECT INTO OUTFILE S3`, Aurora determina la crittografia come segue:

- Se la `ENCRYPTION` clausola è presente nel SQL comando, Aurora si basa solo sul valore `ENCRYPTION` di e non utilizza una variabile di sessione.
- Se la clausola `ENCRYPTION` non è presente, Aurora si basa sul valore della variabile di sessione.

Per ulteriori informazioni, consulta Uso della crittografia lato server con chiavi gestite di Amazon S3 SSE (-S3) e Utilizzo della crittografia lato server con AWS KMS chiavi (SSE-KMS) nella Guida per l'utente di Amazon Simple Storage Service.

Puoi trovare maggiori dettagli su altri parametri nelle istruzioni e LOAD DATA nelle istruzioni, nella SELECTdocumentazione personale. SQL

Considerazioni

Il numero di file scritti nel bucket Amazon S3 dipende dalla quantità di dati selezionata dall'`SELECT INTO OUTFILE S3`istruzione e dalla soglia di dimensione del file per Aurora My. SQL La soglia della dimensione del file predefinita è 6 gigabyte (GB). Se i dati selezionati dall'istruzione sono inferiori alla soglia della dimensione del file, viene creato un singolo file; in caso contrario, vengono creati più file. Altre considerazioni per i file creati da questa dichiarazione includono quanto segue:

- Aurora My SQL garantisce che le righe dei file di dati non siano suddivise tra i confini dei file. Nel caso di più file, la dimensione di ogni file di dati tranne l'ultimo è in genere vicino alla soglia della dimensione del file. Tuttavia, ogni tanto poiché restano sotto la soglia della dimensione del file, una riga viene suddivisa su due file di dati. In questo caso, Aurora My SQL crea un file di dati che mantiene intatta la riga, ma potrebbe superare la soglia di dimensione del file.
- Poiché ogni `SELECT` istruzione in Aurora My SQL viene eseguita come transazione atomica, un'`SELECT INTO OUTFILE S3`istruzione che seleziona un set di dati di grandi dimensioni potrebbe essere eseguita per qualche tempo. Se l'istruzione non riesce per qualsiasi motivo, potrebbe essere necessario ricominciare ed emettere nuovamente l'istruzione. Se l'istruzione non riesce, tuttavia, i file che sono già stati caricati in Amazon S3 rimangono nel bucket Amazon S3 specifico. Puoi utilizzare un'altra istruzione per caricare i dati rimanenti anziché ricominciare da capo.
- Se la quantità di dati da selezionare è elevata (oltre 25 GB), ti consigliamo di utilizzare più istruzioni `SELECT INTO OUTFILE S3` per salvare i dati in Amazon S3. Ogni istruzione dovrebbe selezionare una parte diversa dei dati da salvare e anche specificare un diverso `file_prefix` nel parametro `s3-uri` da utilizzare quando si salvano i file di dati. Il partizionamento dei dati da selezionare con più istruzioni facilita il recupero da un errore in un'istruzione. Se si verifica un errore per un'istruzione, solo una parte dei dati deve essere riselezionata e caricata in Amazon S3. L'utilizzo di più istruzioni aiuta anche a evitare una singola transazione di lunga durata, che può migliorare le prestazioni.
- Se più istruzioni `SELECT INTO OUTFILE S3` che utilizzano lo stesso `file_prefix` nel parametro `s3-uri` vengono eseguite in parallelo per selezionare i dati in Amazon S3, il comportamento non è definito.
- I metadati, come lo schema delle tabelle o i metadati dei file, non vengono caricati da Aurora My su Amazon SQL S3.
- In alcuni casi, potresti rieseguire una query `SELECT INTO OUTFILE S3`, ad esempio per ripristinare a causa di un errore. In questi casi, devi rimuovere tutti i file di dati esistenti nel bucket Amazon S3 con lo stesso prefisso di file specificato in `s3-uri` o includere `OVERWRITE ON` nella query `SELECT INTO OUTFILE S3`.

L'`SELECT INTO OUTFILE S3`istruzione restituisce un tipico numero di SQL errore My e una risposta in caso di esito positivo o negativo. Se non hai accesso al numero di SQL errore e alla risposta di My error, il modo più semplice per determinare quando è stato completato è specificarlo `MANIFEST ON` nell'istruzione. Il file manifest è l'ultimo file scritto dall'istruzione. In altre parole, se hai un file manifest, l'istruzione ha completato.

Al momento non è possibile monitorare direttamente l'avanzamento dell'istruzione `SELECT INTO OUTFILE S3` durante l'esecuzione. Tuttavia, supponiamo che tu stia scrivendo una grande quantità di dati da Aurora SQL My ad Amazon S3 utilizzando questa istruzione e che tu conosca la dimensione dei dati selezionati dall'istruzione. In questo caso, puoi stimare l'avanzamento monitorando la creazione di file di dati in Amazon S3.

Per fare ciò, è possibile utilizzare il fatto che un file di dati viene creato nel bucket Amazon S3 specificato ogni circa 6 GB di dati selezionati dall'istruzione. Dividi la dimensione dei dati selezionati da 6 GB per ottenere il numero stimato di file di dati da creare. È quindi possibile stimare l'avanzamento dell'istruzione monitorando il numero di file caricati in Amazon S3 durante l'esecuzione.

Esempi

L'istruzione seguente seleziona tutti i dati nella `employees` tabella e li salva in un bucket Amazon S3 che si trova in una regione diversa dal cluster Aurora My DB. SQL L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (`,`) e ogni riga è terminata da un carattere di nuova riga (`\n`). L'istruzione restituisce un errore se i file che corrispondono al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

L'istruzione seguente seleziona tutti i dati nella `employees` tabella e li salva in un bucket Amazon S3 che si trova nella stessa regione del cluster Aurora My DB. SQL L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (`,`) e ogni riga è terminata da un carattere di nuova riga (`\n`) e un file manifest. L'istruzione restituisce un errore se i file che corrispondono al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON;
```

La seguente istruzione seleziona tutti i dati nella tabella `employees` e li salva in un bucket Amazon S3 che si trova in una regione diversa dal cluster DB Aurora. L'istruzione crea file di dati in cui ogni

campo è terminato da una virgola (,) e ogni riga è terminata da un carattere di nuova riga (\n). L'istruzione sostituisce ogni file che corrisponde al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    OVERWRITE ON;
```

L'istruzione seguente seleziona tutti i dati nella `employees` tabella e li salva in un bucket Amazon S3 che si trova nella stessa regione del cluster Aurora My DB. SQL L'istruzione crea file di dati in cui ogni campo è terminato da una virgola (,) e ogni riga è terminata da un carattere di nuova riga (\n) e un file manifest. L'istruzione sostituisce ogni file che corrisponde al prefisso del file `sample_employee_data` esistono nel bucket Amazon S3 specificato.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON
    OVERWRITE ON;
```

Richiamo di una funzione Lambda da un cluster Amazon Aurora My DB SQL

È possibile richiamare un AWS Lambda funzione da un cluster Amazon Aurora My SQL -Compatible Edition DB con la funzione nativa o. `lambda_sync` `lambda_async` Prima di richiamare una funzione Lambda da Aurora MySQL, il cluster Aurora DB deve avere accesso a Lambda. Per informazioni dettagliate sulla concessione dell'accesso ad Aurora SQL My, consulta. Accesso di Aurora a Lambda; Per informazioni sulla `lambda_sync` e su funzioni archiviate `lambda_async`, consultare Invocare una funzione Lambda con una funzione Aurora My native SQL.

Puoi anche chiamare un AWS Lambda eseguire una funzione utilizzando una procedura memorizzata. Tuttavia, l'utilizzo di una procedura archiviate è obsoleto. Si consiglia vivamente di utilizzare una funzione SQL nativa di Aurora My se si utilizza una delle seguenti versioni di Aurora My: SQL

- Aurora My SQL versione 2, per cluster compatibili con My SQL 5.7.

- Aurora My SQL versione 3.01 e successive, per cluster compatibili con My SQL 8.0. La stored procedure non è disponibile in Aurora My SQL versione 3.

Per informazioni su come concedere ad Aurora l'accesso a Lambda e invocare una funzione Lambda, consulta i seguenti argomenti.

Argomenti

- Accesso di Aurora a Lambda;
- Invocare una funzione Lambda con una funzione Aurora My native SQL
- Richiamo di una funzione Lambda con una procedura memorizzata Aurora SQL My (obsoleta)

Accesso di Aurora a Lambda;

Prima di poter richiamare le funzioni Lambda da un cluster Aurora SQL My DB, assicurati di concedere al cluster l'autorizzazione ad accedere a Lambda.

Per consentire ad Aurora My di SQL accedere a Lambda

1. Crea un AWS Identity and Access Management (IAM) politica che fornisce le autorizzazioni che consentono al cluster Aurora SQL My DB di richiamare le funzioni Lambda. Per istruzioni, consulta Creazione di una policy IAM per l'accesso alle risorseAWS Lambda.
2. Crea un IAM ruolo e allega la IAM policy che hai creato Creazione di una policy IAM per l'accesso alle risorseAWS Lambda al nuovo ruolo. IAM Per istruzioni, consulta Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS.
3. Imposta il parametro del cluster `aws_default_lambda_role` DB sull'Amazon Resource Name (ARN) del nuovo IAM ruolo.

   Se il cluster fa parte di un database globale Aurora, applica la stessa impostazione per ogni cluster Aurora nel database globale.

   Per ulteriori informazioni sui parametri del cluster DB, vedi Parametri dell'istanza database e del cluster database di Amazon Aurora.
4. Per consentire agli utenti del database in un cluster Aurora My SQL DB di richiamare le funzioni Lambda, associa il ruolo che hai creato Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS al cluster DB. Per informazioni sull'associazione di un IAM ruolo a un cluster DB, consulta. Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL

Se il cluster fa parte di un database globale Aurora, associa il ruolo a ogni cluster Aurora nel database globale.

5. Configura il tuo cluster Aurora My SQL DB per consentire le connessioni in uscita a Lambda. Per istruzioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

   Se il cluster fa parte di un database globale Aurora, abilita le connessioni in uscita per ogni cluster Aurora nel database globale.

Invocare una funzione Lambda con una funzione Aurora My native SQL

> **Note**
>
> È possibile richiamare le funzioni native `lambda_sync` e `lambda_async` quando si utilizza Aurora My SQL versione 2 o Aurora My SQL versione 3.01 e successive. Per ulteriori informazioni sulle SQL versioni di Aurora My, consulta. Aggiornamenti del motore di database per Amazon Aurora My SQL

È possibile richiamare una AWS Lambda funzione da un cluster Aurora SQL My DB chiamando le funzioni native `lambda_sync` e. `lambda_async` Questo approccio può essere utile quando si desidera integrare il database in esecuzione su Aurora My SQL con altri AWS servizi. Ad esempio, potresti voler inviare una notifica utilizzando Amazon Simple Notification Service (AmazonSNS) ogni volta che una riga viene inserita in una tabella specifica del tuo database.

Indice

- Utilizzo delle funzioni native per chiamare una funzione Lambda
  - Concessione del ruolo in Aurora My version 3 SQL
  - Concessione del privilegio in Aurora My versione 2 SQL
  - Sintassi per la funzione lambda_sync
  - Parametri per la funzione lambda_sync
  - Esempio per la funzione lambda_sync
  - Sintassi per la funzione lambda_async
  - Parametri per la funzione lambda_async
  - Esempio per la funzione lambda_async

- Invocazione di una funzione Lambda all'interno di un trigger

Utilizzo delle funzioni native per chiamare una funzione Lambda

Le funzioni `lambda_sync` e `lambda_async` sono funzioni native predefinite che chiamano una funzione Lambda in modo sincrono o asincrono. Quando è necessario conoscere il risultato dell'esecuzione della funzione Lambda prima di passare a un'altra operazione, utilizza la funzione sincrona `lambda_sync`. Quando non è necessario conoscere il risultato della funzione Lambda prima di passare a un'altra operazione, utilizza la funzione asincrona `lambda_async`.

Concessione del ruolo in Aurora My version 3 SQL

In Aurora My SQL versione 3, all'utente che richiama una funzione nativa deve essere concesso il ruolo. `AWS_LAMBDA_ACCESS` Per concedere questo privilegio a un utente, connettersi all'istanza database come utente amministratore ed eseguire la seguente istruzione.

```
GRANT AWS_LAMBDA_ACCESS TO user@domain-or-ip-address
```

Si può revocare questo ruolo eseguendo la seguente istruzione.

```
REVOKE AWS_LAMBDA_ACCESS FROM user@domain-or-ip-address
```

> **Tip**
>
> Quando si utilizza la tecnica del ruolo in Aurora My SQL versione 3, è possibile attivare il ruolo anche utilizzando l'istruzione `SET ROLE role_name` or`SET ROLE ALL`. Se non conosci il sistema di ruoli My SQL 8.0, puoi saperne di più in. Privilegio basato sui ruoli Per maggiori dettagli, consulta Usare i ruoli nel My SQL Reference Manual.
> Questo vale solo per la sessione attiva corrente. Quando ti riconnetti, devi eseguire nuovamente l'`SET ROLE`istruzione per concedere i privilegi. Per ulteriori informazioni, vedere l'SETROLEistruzione nel My SQL Reference Manual.
> Puoi utilizzare il parametro `activate_all_roles_on_login` del cluster di database per attivare automaticamente tutti i ruoli quando un utente si connette a un'istanza database. Quando questo parametro è impostato, in genere non è necessario chiamare l'`SET ROLE`istruzione in modo esplicito per attivare un ruolo. Per ulteriori informazioni, vedere activate_all_roles_on_login nel My Reference Manual. SQL

Tuttavia, è necessario chiamare `SET ROLE ALL` esplicitamente all'inizio di una stored procedure per attivare il ruolo, quando la stored procedure viene chiamata da un altro utente.

Se ricevi un errore come il seguente quando provi a richiamare una funzione Lambda, esegui un'istruzione `SET ROLE`.

```
SQL Error [1227] [42000]: Access denied; you need (at least one of) the Invoke Lambda
 privilege(s) for this operation
```

Assicurati di concedere il ruolo all'utente corretto, come mostrato nelle voci della `mysql.users` tabella. Potrebbero esserci più utenti con lo stesso nome, ma su host diversi. A seconda dell'applicazione o dell'host che richiama la `lambda_sync` funzione, My SQL seleziona l'utente con la corrispondenza migliore in base alle voci della `host` colonna.

Concessione del privilegio in Aurora My versione 2 SQL

In Aurora My SQL versione 2, all'utente che richiama una funzione nativa deve essere concesso il privilegio. `INVOKE LAMBDA` Per concedere questo privilegio a un utente, connettersi all'istanza database come utente amministratore ed eseguire la seguente istruzione.

```
GRANT INVOKE LAMBDA ON *.* TO user@domain-or-ip-address
```

Puoi revocare questo privilegio eseguendo la seguente istruzione.

```
REVOKE INVOKE LAMBDA ON *.* FROM user@domain-or-ip-address
```

Sintassi per la funzione lambda_sync

Invochi la funzione `lambda_sync` in modo sincrono con il tipo di chiamata `RequestResponse`. La funzione restituisce il risultato dell'invocazione Lambda in un payload. JSON La funzione ha la seguente sintassi.

```
lambda_sync (
  lambda_function_ARN,
  JSON_payload
)
```

Parametri per la funzione lambda_sync

La funzione `lambda_sync` include i seguenti parametri.

lambda_function_ ARN

L'Amazon Resource Name (ARN) della funzione Lambda da richiamare.

JSON_payload

Il payload per la funzione Lambda richiamata, in formato. JSON

> **Note**
>
> Aurora My SQL versione 3 supporta le funzioni di JSON analisi di My 8.0. SQL Tuttavia, Aurora My SQL versione 2 non include queste funzioni. JSONl'analisi non è richiesta quando una funzione Lambda restituisce un valore atomico, come un numero o una stringa.

Esempio per la funzione lambda_sync

La seguente query basata su `lambda_sync` richiama la funzione Lambda in modo `BasicTestLambda` sincrono utilizzando la funzione. ARN Il payload per la funzione è `{"operation": "ping"}`.

```
SELECT lambda_sync(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Sintassi per la funzione lambda_async

Invochi la funzione `lambda_async` in modo asincrono con il tipo di chiamata `Event`. La funzione restituisce il risultato dell'invocazione Lambda in un payload. JSON La funzione ha la seguente sintassi.

```
lambda_async (
  lambda_function_ARN,
  JSON_payload
)
```

Parametri per la funzione lambda_async

La funzione `lambda_async` include i seguenti parametri.

lambda_function_ ARN

L'Amazon Resource Name (ARN) della funzione Lambda da richiamare.

JSON_payload

Il payload per la funzione Lambda richiamata, in formato. JSON

> **Note**
>
> Aurora My SQL versione 3 supporta le funzioni di JSON analisi di My 8.0. SQL Tuttavia, Aurora My SQL versione 2 non include queste funzioni. JSONl'analisi non è richiesta quando una funzione Lambda restituisce un valore atomico, come un numero o una stringa.

Esempio per la funzione lambda_async

La seguente query basata su `lambda_async` richiama la funzione Lambda in modo `BasicTestLambda` asincrono utilizzando la funzione. ARN Il payload per la funzione è `{"operation": "ping"}`.

```
SELECT lambda_async(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Invocazione di una funzione Lambda all'interno di un trigger

Puoi utilizzare i trigger per chiamare Lambda sulle istruzioni di modifica dei dati. L'esempio seguente illustra come utilizzare la funzione `lambda_async` nativa e memorizzare il risultato in una variabile.

```
mysql>SET @result=0;
mysql>DELIMITER //
mysql>CREATE TRIGGER myFirstTrigger
      AFTER INSERT
          ON Test_trigger FOR EACH ROW
      BEGIN
      SELECT lambda_async(
          'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
```

```
          '{"operation": "ping"}')
          INTO @result;
      END; //
mysql>DELIMITER ;
```

> **Note**
>
> I trigger non vengono eseguiti una volta per SQL istruzione, ma una volta per riga modificata, una riga alla volta. Quando viene eseguito un trigger, il processo è sincrono. L'istruzione di modifica dei dati viene restituita solo al completamento del trigger.
> Fai attenzione quando richiami una AWS Lambda funzione dai trigger su tabelle che presentano un traffico di scrittura elevato. `INSERTUPDATE`, e i `DELETE` trigger vengono attivati per riga. Un carico di lavoro che richiede molta scrittura su una tabella con `INSERT` o `DELETE` trigger genera un gran numero di chiamate alla funzione. `UPDATE` AWS Lambda

Richiamo di una funzione Lambda con una procedura memorizzata Aurora SQL My (obsoleta)

È possibile richiamare un AWS Lambda funzione da un cluster Aurora My SQL DB chiamando la `mysql.lambda_async` procedura. Questo approccio può essere utile quando si desidera integrare il database in esecuzione su Aurora My SQL con altri AWS servizi. Ad esempio, potresti voler inviare una notifica utilizzando Amazon Simple Notification Service (AmazonSNS) ogni volta che una riga viene inserita in una tabella specifica del tuo database.

Indice

- Aurora: Considerazioni sulla mia versione SQL
- Utilizzo della procedura mysql.lambda_async per chiamare una funzione Lambda (obsoleto)
  - Sintassi
  - Parametri
  - Esempi

Aurora: Considerazioni sulla mia versione SQL

A partire dalla SQL versione 2 di Aurora My, è possibile utilizzare il metodo della funzione nativa anziché queste stored procedure per richiamare una funzione Lambda. Per ulteriori informazioni sulle funzioni native, consulta Utilizzo delle funzioni native per chiamare una funzione Lambda.

In Aurora My SQL versione 2, la stored procedure non `mysql.lambda_async` è più supportata. Ti consigliamo di utilizzare le funzioni Lambda native.

In Aurora My SQL versione 3, la stored procedure non è disponibile.

Utilizzo della procedura mysql.lambda_async per chiamare una funzione Lambda (obsoleto)

La procedura `mysql.lambda_async` è una stored procedure predefinita che chiama una funzione Lambda in modo asincrono. Per utilizzare questa procedura, l'utente del database deve avere il privilegio `EXECUTE` sulla stored procedure `mysql.lambda_async`.

Sintassi

La procedura `mysql.lambda_async` ha la seguente sintassi.

```
CALL mysql.lambda_async (
  lambda_function_ARN,
  lambda_function_input
)
```

Parametri

La procedura `mysql.lambda_async` include i seguenti parametri.

funzione_lambda_ ARN

L'Amazon Resource Name (ARN) della funzione Lambda da richiamare.

lambda_function_input

La stringa di input, in JSON formato, per la funzione Lambda richiamata.

Esempi

Come best practice, ti consigliamo di inserire le chiamate alla procedura `mysql.lambda_async` in una procedura archiviata che può essere richiamata da origini diverse come trigger o codice client. Questo approccio può evitare problemi di resistenza non corrispondente e rendere più semplice per richiamare le funzioni Lambda.

> **Note**
>
> Fai attenzione quando invoci un AWS Lambda funzione generata dai trigger su tabelle che presentano un traffico di scrittura elevato. `INSERTUPDATE`, e i `DELETE` trigger vengono

attivati per riga. Un carico di lavoro di scrittura intenso su una tabella con `INSERT` o o `DELETE` trigger genera un gran numero di chiamate al `UPDATE` AWS Lambda funzione. Sebbene le chiamate alla procedura `mysql.lambda_async` siano asincrone, i trigger sono sincroni. Un'istruzione che genera un gran numero di attivazioni di attivazione non attende la chiamata al AWS Lambda la funzione viene completata, ma attende il completamento dei trigger prima di restituire il controllo al client.

Example Esempio: invocare un AWS Lambda funzione per inviare e-mail

Nell'esempio seguente viene creata una stored procedure che è possibile chiamare nel codice del database per inviare un'e-mail utilizzando una funzione Lambda.

AWS Lambda Funzione

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },

        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Stored procedure

```
DROP PROCEDURE IF EXISTS SES_send_email;
DELIMITER ;;
  CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                  IN email_to VARCHAR(255),
                                  IN subject VARCHAR(255),
                                  IN body TEXT) LANGUAGE SQL
  BEGIN
    CALL mysql.lambda_async(
         'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
         CONCAT('{"email_to" : "', email_to,
             '", "email_from" : "', email_from,
             '", "email_subject" : "', subject,
             '", "email_body" : "', body, '"}')
     );
  END
  ;;
DELIMITER ;
```

Richiama la Stored Procedure per richiamare il AWS Lambda Funzione

```
mysql> call SES_send_email('example_from@amazon.com', 'example_to@amazon.com', 'Email
 subject', 'Email content');
```

Example Esempio: invocare un AWS Lambda funzione per pubblicare un evento da un trigger

L'esempio seguente crea una stored procedure che pubblica un evento utilizzando AmazonSNS. Il codice chiama la procedura da un trigger quando una riga viene aggiunta a una tabella.

AWS Lambda Funzione

```
import boto3

sns = boto3.client('sns')

def SNS_publish_message(event, context):

    return sns.publish(
        TopicArn='arn:aws:sns:us-west-2:123456789012:Sample_Topic',
        Message=event['message'],
        Subject=event['subject'],
        MessageStructure='string'
    )
```

Stored procedure

```
DROP PROCEDURE IF EXISTS SNS_Publish_Message;
DELIMITER ;;
CREATE PROCEDURE SNS_Publish_Message (IN subject VARCHAR(255),
                                      IN message TEXT) LANGUAGE SQL
BEGIN
  CALL mysql.lambda_async('arn:aws:lambda:us-
west-2:123456789012:function:SNS_publish_message',
     CONCAT('{ "subject" : "', subject,
            '", "message" : "', message, '" }')
     );
END
;;
DELIMITER ;
```

Tabella

```
CREATE TABLE 'Customer_Feedback' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'customer_name' varchar(255) NOT NULL,
  'customer_feedback' varchar(1024) NOT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Trigger

```
DELIMITER ;;
CREATE TRIGGER TR_Customer_Feedback_AI
  AFTER INSERT ON Customer_Feedback
  FOR EACH ROW
BEGIN
  SELECT CONCAT('New customer feedback from ', NEW.customer_name),
 NEW.customer_feedback INTO @subject, @feedback;
  CALL SNS_Publish_Message(@subject, @feedback);
END
;;
DELIMITER ;
```

Inserisci una riga nella tabella per attivare la notifica

```
mysql> insert into Customer_Feedback (customer_name, customer_feedback) VALUES ('Sample
 Customer', 'Good job guys!');
```

Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch

Puoi configurare il cluster Aurora MySQL DB per pubblicare dati di log generali, lenti, di controllo e di errore in un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log e utilizzarli CloudWatch per creare allarmi e visualizzare metriche. È possibile utilizzare CloudWatch Logs per archiviare i record di registro in un archivio altamente durevole.

Per pubblicare i log in CloudWatch Logs, i rispettivi log devono essere abilitati. I registri dei dati di errore sono abilitati per impostazione predefinita, ma è necessario abilitare esplicitamente gli altri tipi di log. Per informazioni sull'abilitazione dei log in MySQL, consulta la sezione relativa alla selezione delle destinazioni di output del log delle query generiche e lente nella documentazione di MySQL. Per ulteriori informazioni sull'abilitazione dei log di audit di Aurora MySQL, consulta Abilitazione dell'audit avanzato.

> **Note**
>
> - Se l'esportazione dei dati del registro è disabilitata, Aurora non elimina i gruppi di log esistenti o i flussi di log. Se l'esportazione dei dati di registro è disabilitata, i dati di registro esistenti rimangono disponibili nei CloudWatch registri, a seconda della conservazione dei log, e l'utente deve comunque sostenere costi per i dati dei log di controllo memorizzati. È possibile eliminare i flussi di log e i gruppi di log utilizzando la console CloudWatch Logs, o l' AWS CLI API Logs. CloudWatch
> - Un modo alternativo per pubblicare i log di controllo su CloudWatch Logs consiste nell'abilitare Advanced Auditing, quindi creare un gruppo di parametri del cluster DB personalizzato e impostare il parametro su. `server_audit_logs_upload` `1` L'impostazione predefinita per il parametro del cluster `server_audit_logs_upload` DB è. `0` Per informazioni sull'attivazione del controllo avanzato, vedereUtilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.
>
>   Se si utilizza questo metodo alternativo, è necessario disporre di un ruolo IAM per accedere ai CloudWatch registri e impostare il parametro a `aws_default_logs_role` livello di cluster sull'ARN per questo ruolo. Per ulteriori informazioni sulla creazione del

ruolo, consulta Impostazione IAM dei ruoli per l'accesso AWS services. Tuttavia, se disponi del ruolo `AWSServiceRoleForRDS` collegato al servizio, fornisce l'accesso ai CloudWatch log e sostituisce qualsiasi ruolo definito personalizzato. Per informazioni sui ruoli collegati ai servizi per Amazon RDS, consultare Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

- Se non desideri esportare i log di controllo in Logs, assicurati che tutti i metodi di esportazione CloudWatch dei log di controllo siano disabilitati. Questi metodi sono l' AWS Management Console, l' AWS CLI, l'API RDS e il parametro `server_audit_logs_upload`.
- La procedura è leggermente diversa per Aurora Serverless v1 Cluster DB rispetto ai cluster DB con provisioning o Aurora Serverless v2 Istanze DB. Aurora Serverless v1 i cluster caricano automaticamente tutti i log abilitati tramite i parametri di configurazione.

  Pertanto, è possibile attivare o disattivare il caricamento dei registri per Aurora Serverless v1 Cluster DB attivando e disattivando diversi tipi di log nel gruppo di parametri del cluster DB. Non è possibile modificare le impostazioni del cluster stesso tramite l'API AWS Management Console AWS CLI, o RDS. Per informazioni sull'attivazione e la disattivazione dei log MySQL per Aurora Serverless v1 cluster, vedi. Gruppi di parametri per Aurora Serverless v1

Console

È possibile pubblicare i log di Aurora MySQL per i cluster predisposti su Logs with the console. CloudWatch

Per pubblicare i log Aurora MySQL dalla console

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il cluster DB Aurora MySQL per cui vuoi pubblicare i dati dei log.
4. Scegliere Modify (Modifica).
5. Nella sezione Esportazioni dei log, scegli i log che desideri iniziare a pubblicare su Logs. CloudWatch
6. Scegliere Continue (Continua) e quindi selezionare Modify DB Cluster (Modifica cluster DB) nella pagina di riepilogo.

AWS CLI

È possibile pubblicare i log di Aurora MySQL per i cluster predisposti con la CLI. AWS A tale scopo, esegui il comando con le seguenti opzioni: modify-db-cluster AWS CLI

- `--db-cluster-identifier`— – L'identificatore del cluster DB.
- `--cloudwatch-logs-export-configuration`—L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Puoi pubblicare i log Aurora MySQL anche eseguendo uno dei seguenti comandi della AWS CLI :

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-istantanea
- restore-db-cluster-to-point-in-time

Esegui uno di questi AWS CLI comandi con le seguenti opzioni:

- `--db-cluster-identifier`— – L'identificatore del cluster DB.
- `--engine`— – Il motore di database.
- `--enable-cloudwatch-logs-exports`—L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Potrebbero essere necessarie altre opzioni a seconda del AWS CLI comando eseguito.

Example

Il comando seguente modifica un cluster Aurora MySQL DB esistente per pubblicare i file di registro nei registri. CloudWatch

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Example

Il comando seguente crea un cluster Aurora MySQL DB per pubblicare i file di registro nei registri. CloudWatch

In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine aurora \
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

In Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine aurora ^
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

API RDS

Puoi pubblicare i log di Aurora MySQL per i cluster con provisioning con l'API RDS. A tale scopo, si esegue l'DBClusteroperazione Modifica con le seguenti opzioni:

- `DBClusterIdentifier`— – L'identificatore del cluster DB.
- `CloudwatchLogsExportConfiguration`— L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Puoi anche pubblicare i log Aurora MySQL con l'API RDS eseguendo uno delle seguenti operazioni dell'API RDS:

- CreaDBCluster
- Ripristina da S3 DBCluster
- RipristinaDBClusterFromSnapshot
- RipristinaDBClusterToPointInTime

Utilizza l'operazione API RDS con i seguenti parametri:

- `DBClusterIdentifier`— – L'identificatore del cluster DB.
- `Engine`— – Il motore di database.
- `EnableCloudwatchLogsExports`—L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Potrebbero essere necessari altri parametri a seconda del AWS CLI comando eseguito.

Monitoraggio degli eventi di registro in Amazon CloudWatch

Dopo aver abilitato gli eventi di log di Aurora MySQL, puoi monitorare gli eventi in Amazon Logs. CloudWatch Un nuovo gruppo di log viene creato automaticamente per il cluster DB Aurora con il seguente prefisso, in cui *`cluster-name`* rappresenta il nome del cluster DB e *`log_type`* rappresenta il tipo di log.

```
/aws/rds/cluster/cluster-name/log_type
```

Ad esempio, se configuri la funzione di esportazione per includere il log delle query lente per un cluster DB denominato `mydbcluster`, i dati delle query lente vengono archiviati nel gruppo di log `/aws/rds/cluster/mydbcluster/slowquery`.

Gli eventi di tutte le istanze in un cluster vengono inviati a un gruppo di log utilizzando flussi di log diversi. Il comportamento dipende da quale delle seguenti condizioni è vera:

- Esiste un gruppo di registro con il nome specificato.

  Aurora utilizza il gruppo di registro esistente per esportare i dati di registro per il cluster. Per creare i gruppi di registro con periodi di conservazione di registro predefiniti, filtri di parametri e accesso cliente, puoi utilizzare la configurazione automatizzata, ad esempio AWS CloudFormation.

- Non esiste un gruppo di registri con il nome specificato.

  Quando viene rilevata una voce di registro corrispondente nel file di registro dell'istanza, Aurora MySQL crea automaticamente un nuovo gruppo di log in Logs. CloudWatch Il gruppo di log utilizza il periodo di conservazione predefinito dei log di Never Expire (Nessuna scadenza).

  Per modificare il periodo di conservazione dei log, utilizza la console CloudWatch Logs, o l' AWS CLI API Logs. CloudWatch Per ulteriori informazioni sulla modifica dei periodi di conservazione dei log in CloudWatch Logs, consulta Modificare la conservazione dei dati di log nei log. CloudWatch

Per cercare informazioni all'interno degli eventi di registro per un cluster DB, usa la console CloudWatch Logs AWS CLI, o l' CloudWatch API Logs. Per ulteriori informazioni sulla ricerca e l'applicazione di filtri per i dati di log, consulta Ricerca e filtraggio dei dati di log.

Modalità di laboratorio per Amazon Aurora MySQL

> **Important**
>
> La modalità Lab è stata introdotta in Aurora MySQL versione 2 per abilitare l'ottimizzazione Fast DDL, che ha migliorato l'efficienza di alcune operazioni DDL. Nella versione 3 di Aurora MySQL, la modalità lab è stata rimossa e Fast DDL è stato sostituito dalla funzionalità MySQL 8.0 denominata Instant DDL.

La modalità di laboratorio per Aurora consente di abilitare le caratteristiche di Aurora disponibili nella versione del database Aurora corrente, ma che non vengono abilitate per impostazione predefinita. Sebbene l'utilizzo della modalità di laboratorio per Aurora non sia consigliato nei cluster di database di produzione, questa modalità Aurora può essere utilizzata per abilitare le caratteristiche per i cluster di database negli ambienti di test e sviluppo. Per ulteriori informazioni sulle caratteristiche di Aurora disponibili quando è abilitata la modalità di laboratorio di Aurora, consulta Caratteristiche della modalità di laboratorio per Aurora.

`aurora_lab_mode` è un parametro a livello di istanza impostato nel gruppo di parametri predefiniti. Il parametro è impostato su `0` (disattivato) nel gruppo di parametri predefiniti. Per abilitare la modalità di laboratorio per Aurora, crea un gruppo di parametri personalizzato, imposta il parametro `aurora_lab_mode` su `1` (abilitato) nel gruppo di parametri predefinito e modifica una o più istanze di database Aurora in modo che venga utilizzato il gruppo di parametri personalizzato. Quindi collegati all'endpoint dell'istanza pertinente per provare le funzionalità della modalità di laboratorio. Per informazioni sulla modifica di un gruppo di parametri database, consulta Modifica dei parametri in un gruppo di parametri DB in . Per informazioni sui gruppi di parametri e su Amazon Aurora, consulta Parametri di configurazione Aurora MySQL.

Caratteristiche della modalità di laboratorio per Aurora

La seguente funzionalità Aurora è attualmente disponibile quando la modalità Aurora lab è abilitata. Per poter utilizzare queste caratteristiche, occorre innanzitutto attivare la modalità di laboratorio per Aurora.

Fast DDL

Questa funzione consente di eseguire un'operazione `ALTER TABLE tbl_name ADD COLUMN col_name column_definition` quasi istantaneamente. L'operazione si conclude senza che

vi sia necessità di copiare la tabella e senza alcuna conseguenza materiale sulle altre istruzioni DML. Poiché non utilizza spazio di archiviazione temporaneo per una copia della tabella, rende pratiche le istruzioni DDL anche per tabelle di grandi dimensioni su classi di istanze di piccole dimensioni.

Al momento, l'operazione Fast DDL è supportata solo per l'aggiunta di una colonna nullable, senza un valore predefinito, alla fine della tabella. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consulta Alterazione delle tabelle in Amazon Aurora mediante Fast DDL.

Le migliori pratiche con Amazon Aurora My SQL

Questo argomento include informazioni sulle migliori pratiche e opzioni per l'utilizzo o la migrazione dei dati verso un cluster Amazon Aurora SQL My DB. Le informazioni contenute in questo argomento riassumono e ribadiscono alcune linee guida e procedure disponibili in Gestione di un cluster DB Amazon Aurora.

Indice

- Determinazione dell'istanza database a cui si è connessi
- Best practice per le prestazioni e il dimensionamento di Aurora MySQL
  - Utilizzo delle classi di istanza T per lo sviluppo e i test
  - Ottimizzazione delle query di join indicizzate Aurora MySQL con prefetch asincrono delle chiavi
    - Abilitazione del prefetch asincrono delle chiavi
    - Ottimizzazione delle query per il prefetch asincrono delle chiavi
  - Ottimizzazione di grandi query di join Aurora MySQL con hash join
    - Abilitazione di hash join
    - Ottimizzazione delle query per gli hash join
  - Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL
  - Ottimizzazione delle operazioni di timestamp
  - Errori di overflow dell'ID dell'indice virtuale
- Le migliori pratiche per Aurora La mia alta disponibilità SQL
  - Utilizzo di Amazon Aurora per il disaster recovery con i tuoi database My SQL
  - Migrazione da My SQL ad Amazon Aurora SQL My con tempi di inattività ridotti
  - Evitare prestazioni lente, riavvio automatico e failover per le istanze Aurora My DB SQL
- Consigli per le mie SQL funzionalità in Aurora My SQL
  - Utilizzo della replica multithread in Aurora My SQL
  - Invocando AWS Lambda funzioni che utilizzano le funzioni My SQL native
  - Evitare transazioni XA con Amazon Aurora My SQL
  - Mantenere le chiavi esterne attivate durante gli estratti conto DML
  - Configurazione della frequenza di svuotamento del buffer dei registri
  - Riduzione al minimo e risoluzione dei problemi relativi ai deadlock di Aurora My SQL
    - Riduzione al minimo dei deadlock di InnoDB

    - Monitoraggio dei deadlock di InnoDB
- Valutazione dell'utilizzo delle istanze DB per Aurora My SQL con i parametri di Amazon CloudWatch

Determinazione dell'istanza database a cui si è connessi

Per determinare a quale istanza DB in un cluster Aurora My SQL DB è connessa una connessione, controlla la variabile `innodb_read_only` globale, come mostrato nell'esempio seguente.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

La variabile `innodb_read_only` è impostata su `0N` se si è connessi a un'istanza database di lettura. Questa impostazione è `0FF` se si è connessi a un'istanza database di scrittura, ad esempio un'istanza primaria in un cluster con provisioning.

Questo approccio può essere utile se desideri aggiungere logica al codice dell'applicazione per bilanciare il carico di lavoro o garantire che un'operazione di scrittura utilizzi la connessione corretta.

Best practice per le prestazioni e il dimensionamento di Aurora MySQL

È possibile applicare le best practice seguenti per migliorare le prestazioni e la scalabilità dei cluster Aurora MySQL.

Argomenti

- Utilizzo delle classi di istanza T per lo sviluppo e i test
- Ottimizzazione delle query di join indicizzate Aurora MySQL con prefetch asincrono delle chiavi
- Ottimizzazione di grandi query di join Aurora MySQL con hash join
- Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL
- Ottimizzazione delle operazioni di timestamp
- Errori di overflow dell'ID dell'indice virtuale

Utilizzo delle classi di istanza T per lo sviluppo e i test

Le istanze Amazon Aurora MySQL che utilizzano le classi di istanza database `db.t2`, `db.t3` o `db.t4g` sono ideali per applicazioni che non supportano un carico di lavoro elevato per una

quantità di tempo significativa. Le istanze T sono progettate per offrire prestazioni di base moderate e garantire prestazioni notevolmente maggiori se il carico di lavoro lo richiede. Sono concepite per carichi di lavoro che non utilizzano completamente la CPU spesso o in maniera regolare, ma che occasionalmente necessitano di un incremento delle prestazioni. Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanze T, consulta Istanze espandibili.

Se il cluster Aurora è più grande di 40 TB, non utilizzare le classi di istanza T. Quando il database dispone di un volume elevato di dati, il sovraccarico di memoria per la gestione degli oggetti dello schema può superare la capacità di un'istanza T.

Non abilitare lo schema di prestazioni MySQL sulle istanze T di Amazon Aurora MySQL. In caso contrario, è possibile che la memoria disponibile per l'istanza T potrebbe esaurirsi.

> **Tip**
>
> Se il database a volte è inattivo ma altre volte ha un carico di lavoro notevole, è possibile utilizzare Aurora Serverless v2 in alternativa alle istanze T. Con Aurora Serverless v2, definisci un intervallo di capacità e Aurora ridimensiona automaticamente il database verso l'alto o verso il basso a seconda del carico di lavoro corrente. Per informazioni dettagliate sull'utilizzo, consulta Utilizzo Aurora Serverless v2. Per le versioni del motore di database che è possibile utilizzare con Aurora Serverless v2, consulta Requisiti e limitazioni per Aurora Serverless v2.

Quando si utilizza un'istanza T come istanza di database in un cluster di database Aurora MySQL, si consiglia quanto segue:

- Utilizza la stessa classe di istanza database per tutte le istanze nel cluster di database. Ad esempio, se si utilizza `db.t2.medium` per l'istanza di scrittura, allora si consiglia di usare `db.t2.medium` anche per le istanze di lettura.
- Non regolare le impostazioni di configurazione relative alla memoria, ad esempio `innodb_buffer_pool_size`. Aurora utilizza un insieme altamente sintonizzato di valori di default per i buffer di memoria sulle istanze T. Questi valoridi di default speciali sono necessari per consentire l'esecuzione di Aurora su istanze vincolate dalla memoria. Se si modificano le impostazioni relative alla memoria su un'istanza T, è molto più probabile che si verifichino out-of-memory condizioni, anche se la modifica è intesa ad aumentare le dimensioni del buffer.

- Eseguire il monitoraggio del saldo del credito CPU (`CPUCreditBalance`) per garantire che il relativo livello sia sostenibile. In altre parole, i crediti CPU vengono accumulati alla stessa velocità con cui vengono utilizzati.

  Una volta esauriti i crediti CPU per un'istanza, si assisterà a un calo immediato nella CPU disponibile e a un aumento della latenza di lettura e scrittura per l'istanza. Questa situazione provoca un'importante riduzione delle prestazioni complessive dell'istanza.

  Se il livello del saldo del credito CPU non è sostenibile, consigliamo di modificare l'istanza database in modo da utilizzare una delle classi di istanza database R supportate (dimensionamento del calcolo).

  Per ulteriori informazioni sui parametri di monitoraggio, consulta Visualizzazione delle metriche nella console Amazon RDS.
- Monitora il ritardo di replica (`AuroraReplicaLag`) tra l'istanza di scrittura e le istanze di lettura.

  Se un'istanza di lettura esaurisce i crediti della CPU prima dell'istanza si scrittura, il ritardo risultante può causare il riavvio frequente dell'istanza di lettura. Si tratta di un risultato comune nei casi in cui un'applicazione dispone di un carico elevato di operazioni di lettura distribuito tra le istanze di lettura, allo stesso tempo che l'istanza di scrittura presenta un carico minimo di operazioni di scrittura.

  Se il ritardo della replica aumenta notevolmente, sarà necessario verificare che il saldo del credito CPU per le istanze di lettura nel cluster DB non sia esaurito.

  Se il livello del saldo del credito CPU non è sostenibile, consigliamo di modificare l'istanza database in modo da utilizzare una delle classi di istanza database R supportate (dimensionamento del calcolo).
- Mantenere il numero di inserimenti per transazione al di sotto di 1 milione per i cluster di database per cui è abilitata la registrazione binaria.

  Se il gruppo di parametri del cluster DB per il cluster DB ha il `binlog_format` parametro impostato su un valore diverso da`OFF`, il cluster di DB potrebbe presentare out-of-memory delle condizioni se il cluster DB riceve transazioni che contengono più di 1 milione di righe da inserire. È possibile monitorare il parametro della memoria da liberare (`FreeableMemory`) per determinare se la memoria disponibile per il cluster di database sta per esaurire. Controllando quindi il parametro delle operazioni di scrittura (`VolumeWriteIOPS`) sarà possibile verificare se l'istanza di scrittura riceve un carico elevato di operazioni di scrittura. In tal caso, consigliamo di

aggiornare l'applicazione per limitare gli inserimenti in una transazione a un numero inferiore a 1 milione. In alternativa, è possibile modificare l'istanza in modo da utilizzare una delle classi di istanza database R supportate (dimensionamento del calcolo).

Ottimizzazione delle query di join indicizzate Aurora MySQL con prefetch asincrono delle chiavi

Aurora MySQL può utilizzare il prefetch asincrono delle chiavi per migliorare le prestazioni delle query che eseguono il join delle tabelle negli indici. Questa funzionalità migliora le prestazioni anticipando le righe necessarie per eseguire query in cui una query JOIN richieda l'uso dell'algoritmo di join Batched Key Access (BKA) e le funzionalità di ottimizzazione Multi-Range Read (MRR). Per ulteriori informazioni su BKA e MRR, consulta Block Nested-Loop and Batched Key Access Joins e Multi-Range Read Optimization nella documentazione di MySQL.

Per trarre vantaggio dalla funzionalità di prefetch asincrono delle chiavi, è necessario che una query utilizzi sia gli algoritmi BKA sia le funzionalità MRR. Tale query viene in genere eseguita quando la clausola JOIN di una query utilizza un indice secondario, ma richiede anche alcune colonne dell'indice principale. Puoi ad esempio utilizzare il prefetch asincrono delle chiavi quando una clausola JOIN rappresenta un equijoin sui valori di indice tra una tabella esterna di piccole dimensioni e una interna di dimensioni maggiori, in cui l'indice è altamente selettivo nella tabella più grande. Il prefetch asincrono delle chiavi interagisce con BKA e MRR per eseguire una ricerca nell'indice da secondario a principale durante la valutazione di una clausola JOIN. Il prefetch asincrono delle chiavi identifica le righe necessarie per eseguire la query durante la valutazione della clausola JOIN. Utilizza quindi un thread in background per caricare in modo asincrono le pagine contenenti le righe in memoria prima di eseguire la query.

Il prefetch asincrono delle chiavi è disponibile per Aurora MySQL versione 2.10 e successive e per la versione 3. Per ulteriori informazioni sulle versioni di Aurora MySQL, consulta Aggiornamenti del motore di database per Amazon Aurora My SQL.

Abilitazione del prefetch asincrono delle chiavi

È possibile abilitare la funzionalità di prefetch asincrono delle chiavi impostando `aurora_use_key_prefetch`, una variabile del server MySQL, su on. Per impostazione predefinita, questo valore è impostato su on. È tuttavia possibile abilitare il prefetch asincrono delle chiavi solo dopo aver abilitato anche l'algoritmo di join BKA e aver disabilitato la funzionalità MRR basata sui costi. A tale scopo, è necessario impostare i seguenti valori per `optimizer_switch`, una variabile di server MySQL:

- Imposta `batched_key_access` su on. Questo valore controlla l'utilizzo dell'algoritmo di join BKA. Per impostazione predefinita, questo valore è impostato su `off`.
- Imposta `mrr_cost_based` su `off`. Questo valore controlla l'utilizzo della funzionalità MRR basata sui costi. Per impostazione predefinita, questo valore è impostato su on.

È attualmente possibile impostare questi valori solo a livello di sessione. Il seguente esempio illustra come impostare questi valori in modo da abilitare la funzionalità di prefetch asincrono delle chiavi per la sessione corrente eseguendo le istruzioni SET.

```
mysql> set @@session.aurora_use_key_prefetch=on;
mysql> set @@session.optimizer_switch='batched_key_access=on,mrr_cost_based=off';
```

In modo analogo, è possibile utilizzare le istruzioni SET per disabilitare la funzionalità di prefetch asincrono delle chiavi e l'algoritmo di join BKA e riabilitare la funzionalità MRR basata sui costi per la sessione corrente, come mostrato nell'esempio seguente.

```
mysql> set @@session.aurora_use_key_prefetch=off;
mysql> set @@session.optimizer_switch='batched_key_access=off,mrr_cost_based=on';
```

Per ulteriori informazioni sulle opzioni di ottimizzazione batched_key_access e mrr_cost_based, consulta Switchable Optimizations nella documentazione di MySQL.

Ottimizzazione delle query per il prefetch asincrono delle chiavi

È possibile verificare se una query può trarre vantaggio dalla funzionalità di prefetch asincrono delle chiavi. A tale scopo, utilizza l'istruzione `EXPLAIN` per profilare la query prima di eseguirla. L'istruzione `EXPLAIN` fornisce informazioni sul piano di esecuzione da utilizzare per una query specificata.

Nell'output dell'istruzione `EXPLAIN` la colonna `Extra` descrive le informazioni aggiuntive incluse nel piano di esecuzione. Se la funzionalità di prefetch asincrono delle chiavi si applica a una tabella utilizzata nella query, questa colonna include uno dei seguenti valori:

- `Using Key Prefetching`
- `Using join buffer (Batched Key Access with Key Prefetching)`

Il seguente esempio illustra l'utilizzo di `EXPLAIN` per visualizzare il piano di esecuzione per una query che può sfruttare il prefetch asincrono delle chiavi.

```
mysql> explain select sql_no_cache
    ->     ps_partkey,
    ->     sum(ps_supplycost * ps_availqty) as value
    -> from
    ->     partsupp,
    ->     supplier,
    ->     nation
    -> where
    ->     ps_suppkey = s_suppkey
    ->     and s_nationkey = n_nationkey
    ->     and n_name = 'ETHIOPIA'
    -> group by
    ->     ps_partkey having
    ->         sum(ps_supplycost * ps_availqty) > (
    ->             select
    ->                 sum(ps_supplycost * ps_availqty) * 0.0000003333
    ->             from
    ->                 partsupp,
    ->                 supplier,
    ->                 nation
    ->             where
    ->                 ps_suppkey = s_suppkey
    ->                 and s_nationkey = n_nationkey
    ->                 and n_name = 'ETHIOPIA'
    ->         )
    -> order by
    ->     value desc;
+----+-------------+----------+------+----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
| id | select_type | table    | type | possible_keys        | key           | key_len
 | ref                              | rows | filtered | Extra
                            |
+----+-------------+----------+------+----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
|  1 | PRIMARY     | nation   | ALL  | PRIMARY              | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where; Using temporary;
 Using filesort                |
```

```
|  1 | PRIMARY     | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  1 | PRIMARY     | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
|  2 | SUBQUERY    | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                             |   25 |   100.00 | Using where
                              |
|  2 | SUBQUERY    | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                              |
|  2 | SUBQUERY    | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
+----+-------------+----------+------+-----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
6 rows in set, 1 warning (0.00 sec)
```

Per ulteriori informazioni sul formato di output `EXPLAIN`, consulta l'argomento relativo al formato di output EXPLAIN esteso nella documentazione di MySQL.

Ottimizzazione di grandi query di join Aurora MySQL con hash join

Se devi eseguire il join di una grande quantità di dati tramite una query equijoin, l'utilizzo di un hash join può migliorare le prestazioni della query. Puoi abilitare gli hash join per Aurora MySQL.

Una colonna hash join può essere una qualsiasi espressione complessa. Di seguito sono indicati i modi per eseguire il confronto tra i tipi di dati in una colonna hash join:

- È possibile confrontare qualsiasi elemento nella categoria dei tipi di dati numerici precisi, ad esempio `int`, `bigint`, `numeric` e `bit`.
- È possibile confrontare qualsiasi elemento nella categoria dei tipi di dati numerici di approssimazione, ad esempio `float` e `double`.
- È possibile confrontare elementi nei tipi di stringa, se i tipi di stringa dispongono dello stesso set di caratteri e delle stesse regole di confronto.
- È possibile confrontare elementi con tipi di dati data e timestamp, se i tipi corrispondono.

> **Note**
>
> Non è possibile confrontare i tipi di dati in categorie differenti.

Di seguito sono indicate le limitazioni che si applicano agli hash join per Aurora MySQL:

- Gli outer join sinistri-destri non sono supportati per Aurora MySQL versione 2, ma sono supportati per la versione 3.
- I semi-join come le query secondarie non sono supportati, a meno che non vengano prima materializzate le query secondarie.
- Aggiornamenti o eliminazioni di più tabelle non sono supportati.

  > **Note**
  >
  > Aggiornamenti o eliminazioni di tabelle singole sono supportati.

- Le colonne con tipi di dati BLOB e spaziali non possono essere colonne join in un hash join.

Abilitazione di hash join

Per abilitare gli hash join:

- Aurora MySQL versione 2: imposta il parametro di database o il parametro del cluster di database `aurora_disable_hash_join` su `0`. La disattivazione di `aurora_disable_hash_join` imposta il valore di `optimizer_switch` su `hash_join=on`.
- Aurora MySQL versione 3: imposta il parametro del server MySQL `optimizer_switch` su `block_nested_loop=on`.

Gli hash join sono attivati per impostazione predefinita in Aurora MySQL Versione 3 e Aurora MySQL versione 2. Nell'esempio seguente viene illustrato come abilitare gli hash join per Aurora MySQL versione 3. È possibile rilasciare l'istruzione `select @@optimizer_switch` per prima cosa per vedere quali altre impostazioni sono presenti nella stringa dei parametri di `SET`. Aggiornamento di un'impostazione nel parametro di `optimizer_switch` non cancella o modifica le altre impostazioni.

```
mysql> SET optimizer_switch='block_nested_loop=on';
```

> **Note**
>
> Per Aurora MySQL Versione 3, il supporto hash join è disponibile in tutte le versioni secondarie ed è attivato per impostazione predefinita.
> Per Aurora MySQL versione 2, il supporto hash join è disponibile in tutte le versioni secondarie. In Aurora MySQL versione 2, la funzionalità di join hash è sempre controllata dal valore `aurora_disable_hash_join`.

Con questa impostazione, l'ottimizzatore sceglie di utilizzare un hash join in base al costo, alle caratteristiche della query e alla disponibilità delle risorse. Se la stima dei costi non è corretta, puoi imporre all'ottimizzatore di scegliere un hash join, impostando `hash_join_cost_based`, una variabile del server MySQL, su `off`. L'esempio seguente illustra come imporre all'ottimizzatore di scegliere un hash join.

```
mysql> SET optimizer_switch='hash_join_cost_based=off';
```

> **Note**
>
> Questa impostazione sostituisce le decisioni dell'ottimizzatore basato sui costi. Sebbene l'impostazione possa essere utile per il test e lo sviluppo, si consiglia di non utilizzarla in produzione.

Ottimizzazione delle query per gli hash join

Per sapere se una query può utilizzare un hash join, è possibile utilizzare in primo luogo l'istruzione `EXPLAIN` per profilare la query. L'istruzione `EXPLAIN` fornisce informazioni sul piano di esecuzione da utilizzare per una query specificata.

Nell'output dell'istruzione `EXPLAIN` la colonna `Extra` descrive le informazioni aggiuntive incluse nel piano di esecuzione. Se un hash join si applica alle tabelle utilizzate nella query, questa colonna include valori simili a quelli indicato di seguito:

- `Using where; Using join buffer (Hash Join Outer table table1_name)`
- `Using where; Using join buffer (Hash Join Inner table table2_name)`

Il seguente esempio illustra l'utilizzo di EXPLAIN per visualizzare il piano di esecuzione per una query con hash join.

```
mysql> explain SELECT sql_no_cache * FROM hj_small, hj_big, hj_big2
    ->     WHERE hj_small.col1 = hj_big.col1 and hj_big.col1=hj_big2.col1 ORDER BY 1;
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows |
 Extra                                                          |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
|  1 | SIMPLE      | hj_small | ALL  | NULL          | NULL | NULL    | NULL |    6 |
 Using temporary; Using filesort                                |
|  1 | SIMPLE      | hj_big   | ALL  | NULL          | NULL | NULL    | NULL |   10 |
 Using where; Using join buffer (Hash Join Outer table hj_big)  |
|  1 | SIMPLE      | hj_big2  | ALL  | NULL          | NULL | NULL    | NULL |   15 |
 Using where; Using join buffer (Hash Join Inner table hj_big2) |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
3 rows in set (0.04 sec)
```

Nell'output `Hash Join Inner table` indica la tabella utilizzata per creare la tabella hash e `Hash Join Outer table` indica la tabella utilizzata per il probe della tabella hash.

Per ulteriori informazioni sul formato di output `EXPLAIN` esteso, consulta Formato di output EXPLAIN esteso nella documentazione di MySQL.

In Aurora MySQL 2.08 e versioni successive, è possibile utilizzare gli hint SQL per determinare se una query utilizza o meno hash join e quali tabelle utilizzare per i lati di build e probe del join. Per informazioni dettagliate, consultare Aurora I miei suggerimenti SQL.

Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL

Puoi utilizzare Amazon Aurora con l'istanza database MySQL per usufruire delle funzionalità di dimensionamento della lettura di Amazon Aurora ed espandere il reale carico di lavoro della tua istanza database MySQL. Per utilizzare Aurora per ridimensionare la lettura dell'istanza database MySQL, crea un cluster di database Aurora MySQL e rendilo una replica di lettura per l'istanza database MySQL. Quindi connettiti al cluster Aurora MySQL per elaborare le query di lettura. Il database di origine può essere un'istanza database RDS for MySQL o un database MySQL in

esecuzione esternamente a Amazon RDS. Per ulteriori informazioni, consulta Ridimensionamento delle letture per il tuo SQL database My con Amazon Aurora.

Ottimizzazione delle operazioni di timestamp

Quando il valore della variabile di sistema `time_zone` è impostato su `SYSTEM`, ogni chiamata di funzione MySQL che richiede un calcolo del fuso orario effettua una chiamata alla libreria di sistema. Quando esegui istruzioni SQL che restituiscono o modificano tali valori `TIMESTAMP` in caso di elevata concorrenza, è possibile che si verifichi un aumento della latenza, dei conflitti di blocco e dell'utilizzo della CPU. Per ulteriori informazioni, consulta time_zone nella documentazione di MySQL.

Per evitare questo comportamento, consigliamo di modificare il valore del parametro `time_zone` del cluster database impostandolo su `UTC`. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.

Sebbene il parametro `time_zone` sia dinamico (non richiede il riavvio del server di database), il nuovo valore viene utilizzato solo per le nuove connessioni. Per assicurarti che tutte le connessioni siano aggiornate in modo che venga utilizzato il nuovo valore `time_zone`, consigliamo di riciclare le connessioni alle applicazioni dopo aver aggiornato il parametro del cluster database.

Errori di overflow dell'ID dell'indice virtuale

Aurora MySQL limita i valori per l'indice virtuale IDs a 8 bit per prevenire un problema causato dal formato di annullamento in MySQL. Se un indice supera il limite di ID dell'indice virtuale, il cluster potrebbe non essere disponibile. Quando un indice si avvicina al limite di ID dell'indice virtuale o quando si tenta di creare un indice superiore al limite di ID dell'indice virtuale, RDS potrebbe generare un codice di errore `63955` o un codice di avviso. `63955` Per risolvere un errore relativo al limite degli ID dell'indice virtuale, ti consigliamo di ricreare il database con un dump logico e ripristinarlo.

Per ulteriori informazioni sul dump e il ripristino logici per Amazon Aurora MySQL, consulta Migrare database di grandi dimensioni su Amazon Aurora MySQL usando and. MyDumper MyLoader Per ulteriori informazioni sull'accesso ai log degli errori in Amazon Aurora, consulta. Monitoraggio dei file di log di Aurora

Le migliori pratiche per Aurora La mia alta disponibilità SQL

È possibile applicare le seguenti best practice per migliorare la disponibilità dei cluster Aurora MySQL.

Argomenti

- Utilizzo di Amazon Aurora per il disaster recovery con i tuoi database My SQL
- Migrazione da My SQL ad Amazon Aurora SQL My con tempi di inattività ridotti
- Evitare prestazioni lente, riavvio automatico e failover per le istanze Aurora My DB SQL

Utilizzo di Amazon Aurora per il disaster recovery con i tuoi database My SQL

Puoi usare Amazon Aurora con la tua istanza My SQL DB per creare un backup offsite per il disaster recovery. Per utilizzare Aurora per il disaster recovery della tua istanza My SQL DB, crea un cluster Amazon Aurora DB e rendilo una replica di lettura della tua istanza My DB. SQL Questo vale per un'istanza RDS for My SQL DB o un SQL database My in esecuzione all'esterno di AmazonRDS.

> **Important**
>
> Quando configuri la replica tra un'istanza My SQL DB e un cluster Amazon Aurora SQL My DB, devi monitorare la replica per assicurarti che rimanga integra e ripararla se necessario.

Per istruzioni su come creare un cluster Amazon Aurora My SQL DB e renderlo una replica di lettura della tua istanza My SQL DB, segui la procedura riportata in. Utilizzo di Amazon Aurora per dimensionare le letture per il database MySQL

Per ulteriori informazioni sui modelli di disaster recovery, consulta Come scegliere la migliore opzione di disaster recovery per il tuo Amazon Aurora My SQL cluster.

Migrazione da My SQL ad Amazon Aurora SQL My con tempi di inattività ridotti

Quando importi dati da un SQL database My che supporta un'applicazione live in un cluster Amazon Aurora SQL My DB, potresti voler ridurre il tempo di interruzione del servizio durante la migrazione. A tale scopo, puoi utilizzare la procedura documentata in Importazione di dati in un'istanza DB My o SQL MariaDB con tempi di inattività ridotti nella Amazon Relational Database Service User Guide. che può essere particolarmente utile se utilizzi un database di dimensioni particolarmente elevate. È possibile utilizzare la procedura per ridurre i costi di importazione riducendo al minimo la quantità di dati trasmessi attraverso la rete a AWS.

La procedura elenca i passaggi per trasferire una copia dei dati del database su un'EC2istanza Amazon e importare i dati in una nuova istanza RDS for My SQL DB. Poiché Amazon Aurora è

compatibile con MySQL, puoi invece utilizzare un cluster Amazon Aurora DB per l'istanza RDS Amazon SQL My DB di destinazione.

Evitare prestazioni lente, riavvio automatico e failover per le istanze Aurora My DB SQL

Se si eseguono uno o più carichi di lavoro pesanti che eccedono le risorse allocate dell'istanza database, è possibile che le risorse su cui si esegue l'applicazione e il database Aurora si esauriscano. Per ottenere parametri sulla tua istanza di database come l'utilizzo, CPU l'utilizzo della memoria e il numero di connessioni al database utilizzate, puoi fare riferimento ai parametri forniti da Amazon, Performance CloudWatch Insights e Enhanced Monitoring. Per ulteriori informazioni sul monitoraggio dell'istanza database, consultare Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Se il carico di lavoro esaurisce le risorse in uso, è possibile che l'istanza database rallenti, venga riavviata o esegua un failover su un'altra istanza database. Per evitare che questo si verifichi, occorre monitorare l'utilizzo delle risorse, esaminare il carico di lavoro in esecuzione sull'istanza database ed eseguire le ottimizzazioni dove necessario. Se le ottimizzazioni non migliorano le metriche dell'istanza e non mitigano l'esaurimento delle risorse, valutare la possibilità di aumentare l'istanza database prima che raggiunga i suoi limiti. Per ulteriori informazioni sulle classi di istanza database disponibili e le relative specifiche, consultare Classi di istanze DB Amazon Aurora.

Consigli per le mie SQL funzionalità in Aurora My SQL

Le seguenti funzionalità sono disponibili nella compatibilità con Aurora My SQL for MySQL. Tuttavia, hanno problemi di prestazioni, scalabilità, stabilità o compatibilità nell'ambiente Aurora. Pertanto, suggeriamo di seguire specifiche linee guida quando si utilizzano queste caratteristiche. Ad esempio, non è consigliabile utilizzare determinate funzionalità per le distribuzioni Aurora di produzione.

Argomenti

- Utilizzo della replica multithread in Aurora My SQL
- Invocando AWS Lambda funzioni che utilizzano le funzioni My SQL native
- Evitare transazioni XA con Amazon Aurora My SQL
- Mantenere le chiavi esterne attivate durante gli estratti conto DML
- Configurazione della frequenza di svuotamento del buffer dei registri
- Riduzione al minimo e risoluzione dei problemi relativi ai deadlock di Aurora My SQL

Utilizzo della replica multithread in Aurora My SQL

Con la replica di log binari multithread, un SQL thread legge gli eventi dal relay log e li mette in coda per l'applicazione dei thread di lavoro. SQL I thread di lavoro sono gestiti da SQL un thread coordinatore. Gli eventi di log binari vengono applicati in parallelo quando possibile.

La replica multithread è supportata in Aurora My versione 3 e in Aurora My SQL versione 2.12.1 e successive. SQL

Per SQL le versioni di Aurora My precedenti alla 3.04, Aurora utilizza la replica a thread singolo per impostazione predefinita quando un cluster Aurora My SQL DB viene utilizzato come replica di lettura per la replica dei log binari.

Le versioni precedenti di Aurora My SQL version 2 hanno ereditato diversi problemi relativi alla replica multithread da My Community Edition. SQL Per queste versioni, si consiglia di non utilizzare la replica multithread in produzione.

Se si utilizza la replica multithread, si consiglia di testarla accuratamente.

Per ulteriori informazioni sull'uso della replica in Amazon Aurora, consulta Replica con Amazon Aurora. Per ulteriori informazioni sulla replica multithread in Aurora My, vedere. SQL Replica di log binari multithread

Invocando AWS Lambda funzioni che utilizzano le funzioni My SQL native

Consigliamo di utilizzare le SQL funzioni My native `lambda_sync` e di `lambda_async` richiamare le funzioni Lambda.

In caso di utilizzo della procedura `mysql.lambda_async` obsoleta, ti consigliamo di eseguire il wrapping delle chiamate alla procedura `mysql.lambda_async` in una stored procedure. Questa stored procedure può essere chiamata da varie origini, ad esempio trigger o codice client. Questo approccio può evitare problemi di resistenza non corrispondente e rendere più semplice per i programmatori del database richiamare le funzioni Lambda.

Per ulteriori informazioni sul richiamo delle funzioni Lambda da Amazon Aurora, consulta Richiamo di una funzione Lambda da un cluster Amazon Aurora My DB SQL.

Evitare transazioni XA con Amazon Aurora My SQL

Ti consigliamo di non utilizzare le transazioni eXtended Architecture (XA) con Aurora SQL My, poiché possono causare lunghi tempi di ripristino se l'XA si trovasse nello stato. `PREPARED` Se devi utilizzare transazioni XA con Aurora SQL My, segui queste best practice:

- Non lasciare una transazione XA aperta nello stato `PREPARED`.
- Mantieni le dimensioni delle transazioni XA più ridotte possibile.

Per ulteriori informazioni sull'utilizzo delle transazioni XA con MySQL, consulta Transazioni XA nella documentazione My. SQL

Mantenere le chiavi esterne attivate durante gli estratti conto DML

Ti consigliamo vivamente di non eseguire alcuna istruzione Data Definition Language (DDL) quando la `foreign_key_checks` variabile è impostata su `0` (off).

Per inserire o aggiornare righe che richiedono una violazione temporanea delle chiavi esterne, procedere nel seguente modo:

1. Imposta `foreign_key_checks` su `0`.
2. Apporta modifiche al linguaggio di manipolazione dei dati (DML).
3. Assicurarsi che le modifiche apportate non violino eventuali vincoli delle chiavi esterne.
4. Impostare `foreign_key_checks` su `1` (attivato).

È inoltre necessario attenersi alle ulteriori best practice seguenti per i vincoli delle chiavi esterne:

- Assicurarsi che le applicazioni client non impostino la variabile `foreign_key_checks` su `0` come parte della variabile `init_connect`.
- Se un ripristino da un backup logico come `mysqldump` non riesce o è incompleto, assicurarsi che `foreign_key_checks` sia impostato su `1` prima di avviare eventuali altre operazioni nella stessa sessione. Un backup logico imposta `foreign_key_checks` su `0` all'avvio.

Configurazione della frequenza di svuotamento del buffer dei registri

In My SQL Community Edition, per rendere durevoli le transazioni, il buffer di log di InnoDB deve essere scaricato su una memoria durevole. Per configurare la frequenza di svuotamento del buffer dei registri su disco si utilizza il parametro `innodb_flush_log_at_trx_commit`.

Quando si imposta il parametro `innodb_flush_log_at_trx_commit` sul valore predefinito di 1, il buffer dei registri viene svuotato ad ogni commit della transazione. Questa impostazione aiuta a mantenere il database conforme. ACID Ti consigliamo di mantenere il valore predefinito di 1.

La modifica `innodb_flush_log_at_trx_commit` a un valore non predefinito può aiutare a ridurre la latenza del linguaggio di manipolazione dei dati (DML), ma compromette la durabilità dei record di registro. Questa mancanza di durabilità rende il database non conforme. ACID È consigliabile che i database siano ACID conformi per evitare il rischio di perdita di dati in caso di riavvio del server. Per ulteriori informazioni su questo parametro, consulta innodb_flush_log_at_trx_commit nella mia documentazione. SQL

In Aurora MySQL, l'elaborazione dei redo log viene scaricata sul livello di archiviazione, quindi non si verifica alcun trasferimento dei file di registro sull'istanza DB. Quando viene emesso un comando di scrittura, i registri redo vengono inviati direttamente dall'istanza database di scrittura al volume del cluster Aurora. Le sole scritture che attraversano la rete sono i record di registro redo. Nessuna pagina viene mai scritta dal livello database.

Per impostazione predefinita, ogni thread che esegue il commit di una transazione attende la conferma dal volume del cluster Aurora. Questa conferma indica che tale record e tutti i record di registro redo precedenti sono stati scritti e hanno raggiunto il quorum. La persistenza dei record di registro e il raggiungimento del quorum rendono la transazione durevole, tramite commit automatico o commit esplicito. Per ulteriori informazioni sull'architettura di archiviazione Aurora, consultare Amazon Aurora storage demystified.

Aurora My SQL non scarica i log nei file di dati come fa My SQL Community Edition. Tuttavia, è possibile utilizzare il parametro `innodb_flush_log_at_trx_commit` per allentare i vincoli di durabilità durante la scrittura dei record di registro redo nel volume del cluster Aurora.

Per Aurora My SQL versione 2:

- `innodb_flush_log_at_trx_commit=` 0 o 2 — Il database non attende la conferma che i record di redo log siano stati scritti nel volume del cluster Aurora.
- `innodb_flush_log_at_trx_commit=` 1 — Il database attende la conferma che i record di redo log siano stati scritti nel volume del cluster Aurora.

Per Aurora My SQL versione 3:

- `innodb_flush_log_at_trx_commit=` 0 — Il database non attende la conferma che i record di redo log siano stati scritti nel volume del cluster Aurora.
- `innodb_flush_log_at_trx_commit=` 1 o 2 — Il database attende la conferma che i record di redo log siano stati scritti nel volume del cluster Aurora.

Pertanto, per ottenere lo stesso comportamento non predefinito in Aurora SQL My versione 3 che otterresti con il valore impostato su 0 o 2 in Aurora SQL My versione 2, imposta il parametro su 0.

Sebbene queste impostazioni possano ridurre DML la latenza del client, possono anche causare la perdita di dati in caso di failover o riavvio. Pertanto, ti consigliamo di mantenere il parametro `innodb_flush_log_at_trx_commit` impostato sul valore predefinito di 1.

Sebbene si possa verificare una perdita di dati sia in My SQL Community Edition che in Aurora MySQL, il comportamento è diverso in ogni database a causa delle diverse architetture. Queste differenze architetturali possono portare a vari gradi di perdita di dati. Per assicurarvi che il database sia ACID conforme, impostate sempre su 1. `innodb_flush_log_at_trx_commit`

> **Note**
>
> In Aurora My SQL versione 3, prima di poter passare `innodb_flush_log_at_trx_commit` a un valore diverso da 1, è necessario modificare il valore `innodb_trx_commit_allow_data_loss` di 1. In questo modo, riconosci il rischio di perdita dei dati.

Riduzione al minimo e risoluzione dei problemi relativi ai deadlock di Aurora My SQL

Gli utenti che eseguono carichi di lavoro che restituiscono regolarmente violazioni dei vincoli su indici secondari univoci o chiavi esterne quando modificano contemporaneamente i record sulla stessa pagina di dati, possono riscontrare un aumento dei deadlock e dei timeout di attesa dei blocchi. Questi deadlock e timeout sono dovuti a una correzione di bug in My Community Edition. SQL

Questa correzione è inclusa nelle versioni 5.7.26 e successive di My SQL Community Edition ed è stata trasferita nelle versioni 2.10.3 e successive di SQL Aurora My. La correzione è necessaria per imporre la serializzabilità, implementando un blocco aggiuntivo per questi tipi di operazioni del linguaggio di manipolazione dei dati (DML), sulle modifiche apportate ai record in una tabella InnoDB. Questo problema è stato scoperto nell'ambito di un'indagine sui problemi di deadlock introdotti da una precedente correzione di bug di My Community Edition. SQL

La correzione ha modificato la gestione interna per il rollback parziale dell'aggiornamento di una tupla (riga) nel motore di archiviazione InnoDB. Le operazioni che generano violazioni dei vincoli su chiavi esterne o indici secondari univoci causano un rollback parziale. Sono incluse, a titolo esemplificativo e non esaustivo, le istruzioni `INSERT...ON DUPLICATE KEY UPDATE`, `REPLACE INTO`, e `INSERT IGNORE` simultanee (upsert).

In questo contesto, il rollback parziale non si riferisce alle transazioni a livello di applicazione, ma piuttosto a un rollback interno di InnoDB delle modifiche di un indice cluster, quando si verifica una violazione dei vincoli. Ad esempio, durante un'operazione upsert viene individuato un valore chiave duplicato.

In una normale operazione di inserimento, InnoDB crea atomicamente le voci degli indici secondari e in cluster per ciascun indice. Se InnoDB rileva un valore duplicato in un indice secondario univoco durante un'operazione di upsert, la voce inserita nell'indice in cluster deve essere ripristinata (rollback parziale) e l'aggiornamento deve quindi essere applicato alla riga duplicata esistente. Durante questa fase di rollback parziale interno, InnoDB deve bloccare ogni record considerato come parte dell'operazione. La correzione garantisce la serializzabilità delle transazioni introducendo un blocco aggiuntivo dopo il rollback parziale.

Riduzione al minimo dei deadlock di InnoDB

Puoi utilizzare i seguenti approcci per ridurre la frequenza dei deadlock nell'istanza database. Altri esempi sono disponibili nella mia SQL documentazione.

1. Per ridurre le possibilità di deadlock, esegui il commit delle transazioni immediatamente dopo aver apportato una serie di modifiche correlate. Puoi farlo suddividendo le transazioni di grandi dimensioni (aggiornamenti di più righe tra i commit) in transazioni più piccole. Se stai inserendo righe in batch, prova a ridurre le dimensioni degli inserimenti batch, specialmente quando usi le operazioni upsert menzionate in precedenza.

   Per ridurre il numero di possibili rollback parziali, puoi provare alcuni dei seguenti approcci:

   a. Sostituisci le operazioni di inserimento in batch con l'inserimento di una riga alla volta. In tal modo puoi ridurre il periodo di tempo in cui i blocchi vengono mantenuti dalle transazioni che possono avere conflitti.

   b. Invece di utilizzare`REPLACE INTO`, riscrivi l'SQListruzione come transazione a più dichiarazioni come la seguente:

   ```
   BEGIN;
   DELETE conflicting rows;
   INSERT new rows;
   COMMIT;
   ```

   c. Invece di utilizzare`INSERT...ON DUPLICATE KEY UPDATE`, riscrivi l'SQListruzione come transazione a più dichiarazioni come la seguente:

```
BEGIN;
SELECT rows that conflict on secondary indexes;
UPDATE conflicting rows;
INSERT new rows;
COMMIT;
```

2. Evita le transazioni di lunga durata, attive o inattive, che potrebbero rimanere bloccate. Ciò include sessioni interattive di My SQL client che potrebbero rimanere aperte per un periodo prolungato con una transazione non impegnata. Quando si ottimizzano le dimensioni delle transazioni o dei batch, l'impatto può variare in base a una serie di fattori come la concorrenza, il numero di duplicati e la struttura delle tabelle. Qualsiasi modifica deve essere implementata e testata in base al carico di lavoro.
3. In alcune situazioni, i deadlock possono verificarsi quando due transazioni tentano di accedere agli stessi set di dati, in una o più tabelle, in ordini diversi. Per evitare queste situazioni, è possibile modificare le transazioni per accedere ai dati nello stesso ordine, serializzando così l'accesso. Ad esempio, crea una coda di transazioni da completare. Questo approccio può aiutare a evitare deadlock quando si verificano più transazioni contemporaneamente.
4. L'aggiunta di indici scelti con attenzione alle tabelle può migliorare la selettività e ridurre la necessità di accedere alle righe, con conseguente riduzione dei blocchi.
5. Se riscontri un blocco di intervallo, puoi impostare il livello di isolamento della transazione su `READ COMMITTED` per la sessione o la transazione per evitarlo. Per ulteriori informazioni sui livelli di isolamento di InnoDB e sui loro comportamenti, consulta Livelli di isolamento delle transazioni nella mia documentazione. SQL

> **Note**
>
> Sebbene sia possibile prendere precauzioni per ridurre la possibilità che si verifichino, i deadlock sono un comportamento previsto del database e possono comunque verificarsi. Le applicazioni devono disporre della logica necessaria per gestire i deadlock quando si verificano. Ad esempio, implementa la logica dei tentativi e del backup nell'applicazione. È consigliabile risolvere la causa principale del problema, ma se si verifica un deadlock, l'applicazione ha la possibilità di attendere e ritentare.

Monitoraggio dei deadlock di InnoDB

I deadlock possono verificarsi in My SQL quando le transazioni dell'applicazione tentano di applicare blocchi a livello di tabella e di riga in un modo che si traduce in un'attesa circolare. Un deadlock occasionale di InnoDB non è necessariamente un problema, perché il motore di archiviazione InnoDB rileva immediatamente la condizione e ripristina automaticamente le transazioni. Se riscontri spesso dei deadlock, ti consigliamo di rivedere e modificare l'applicazione per contenere i problemi di prestazioni ed evitare i deadlock. Quando il rilevamento dei deadlock è attivato (impostazione predefinita), InnoDB rileva automaticamente i deadlock delle transazioni e ripristina una o più transazioni per interrompere il deadlock. InnoDB cerca di selezionare piccole transazioni da ripristinare, in cui la dimensione della transazione è determinata dal numero di righe inserite, aggiornate o eliminate.

- Istruzione `SHOW ENGINE`: l'istruzione `SHOW ENGINE INNODB STATUS \G` contiene i dettagli del deadlock più recente riscontrato nel database dall'ultimo riavvio.
- Il mio registro SQL degli errori: se si verificano frequenti deadlock in cui l'output dell'`SHOW ENGINE`istruzione è inadeguato, è possibile attivare il parametro del cluster DB innodb_print_all_deadlocks.

  Quando questo parametro è attivato, le informazioni su tutti i deadlock nelle transazioni utente di InnoDB vengono registrate nel registro degli errori di Aurora My. SQL
- Parametri Amazon: ti consigliamo inoltre di monitorare in modo proattivo i deadlock utilizzando la CloudWatch metrica. CloudWatch `Deadlocks` Per ulteriori informazioni, consulta Parametri a livello di istanza per Amazon Aurora.
- Amazon CloudWatch Logs: con CloudWatch Logs puoi visualizzare metriche, analizzare i dati di log e creare allarmi in tempo reale. Per ulteriori informazioni, consulta Monitoraggio degli errori in Amazon Aurora My e SQL Amazon RDS for My con SQL Amazon CloudWatch e invio di notifiche tramite Amazon. SNS

  Utilizzando CloudWatch Logs with on`innodb_print_all_deadlocks`, puoi configurare allarmi per avvisarti quando il numero di deadlock supera una determinata soglia. Per definire una soglia, ti consigliamo di osservare le tendenze e utilizzare un valore basato sul normale carico di lavoro.
- Approfondimenti sulle prestazioni: quando utilizzi Approfondimenti sulle prestazioni, puoi monitorare le metriche `innodb_deadlocks` e `innodb_lock_wait_timeout`. Per ulteriori informazioni su tali parametri, consulta Contatori non nativi per Aurora MySQL.

Valutazione dell'utilizzo delle istanze DB per Aurora My SQL con i parametri di Amazon CloudWatch

Puoi utilizzare i CloudWatch parametri per monitorare il throughput dell'istanza DB e determinare se la classe di istanza DB fornisce risorse sufficienti per le tue applicazioni. Per informazioni sui limiti delle classi di istanze DB, consulta. Specifiche hardware per le classi di istanza database per Aurora Trova le specifiche per la tua classe di istanza DB per trovare le prestazioni di rete.

Se l'utilizzo dell'istanza database è prossimo al limite della classe di istanza, le prestazioni potrebbero iniziare a rallentare. Le CloudWatch metriche possono confermare questa situazione, quindi puoi pianificare la scalabilità manuale verso una classe di istanze più ampia.

Combina i seguenti valori CloudWatch delle metriche per scoprire se ti stai avvicinando al limite della classe di istanza:

- NetworkThroughput— La quantità di throughput di rete ricevuta e trasmessa dai client per ogni istanza nel cluster Aurora DB. La velocità di trasmissione effettiva non include il traffico di rete tra le istanze nel cluster database e il volume del cluster.
- StorageNetworkThroughput— La quantità di throughput di rete ricevuta e inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.

Aggiungi NetworkThroughputa per trovare il StorageNetworkThroughputthroughput di rete ricevuto e inviato al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB. Il limite della classe dell'istanza deve essere maggiore della somma di questi due parametri combinati.

È possibile utilizzare i seguenti parametri per esaminare ulteriori dettagli del traffico di rete proveniente dalle applicazioni client durante l'invio e la ricezione:

- NetworkReceiveThroughput— La quantità di throughput di rete ricevuta dai client da ciascuna istanza DB nel cluster Aurora SQL My DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- NetworkTransmitThroughput— La quantità di throughput di rete inviata ai client da ciascuna istanza nel cluster Aurora DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- StorageNetworkReceiveThroughput— La quantità di throughput di rete ricevuta dal sottosistema di archiviazione Aurora da ciascuna istanza del cluster DB.

- StorageNetworkTransmitThroughput— La quantità di throughput di rete inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster DB.

Sommate tutte queste metriche per valutare il confronto tra l'utilizzo della rete e il limite della classe di istanze DB. Il limite della classe di istanza deve essere maggiore della somma di questi parametri combinati.

I limiti di rete e CPU l'utilizzo per lo storage sono direttamente correlati. Quando la velocità di trasmissione della rete aumenta, aumenta anche l'CPUutilizzo. Il monitoraggio CPU e l'utilizzo della rete forniscono informazioni su come e perché le risorse vengono esaurite.

Per ridurre al minimo l'utilizzo della rete, puoi prendere in considerazione quanto segue:

- Utilizzo di una classe di istanza DB più grande.
- La suddivisione delle richieste di scrittura in batch per ridurre le transazioni in generale.
- L'instradamento del carico di lavoro di sola lettura a un'istanza di sola lettura.
- L'eliminazione degli indici non utilizzati.

Risoluzione dei problemi relativi alle prestazioni del database Amazon Aurora My SQL

Questo argomento si concentra su alcuni problemi comuni di prestazioni di Aurora My SQL DB e su come risolvere o raccogliere informazioni per risolverli rapidamente. Dividiamo le prestazioni del database in due categorie:

- Prestazioni del server: l'intero server del database è più lento.
- Prestazioni delle query: l'esecuzione di una o più query richiede più tempo.

AWS opzioni di monitoraggio

Si consiglia di utilizzare le seguenti opzioni di AWS monitoraggio per facilitare la risoluzione dei problemi:

- Amazon CloudWatch : Amazon CloudWatch monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale. Puoi utilizzarlo CloudWatch per raccogliere e tenere traccia delle metriche, che sono variabili che puoi misurare per le tue risorse e applicazioni. Per ulteriori informazioni, consulta What is Amazon CloudWatch? .

  È possibile visualizzare tutte le metriche di sistema e le informazioni di processo per le istanze DB su. AWS Management Console Puoi configurare il tuo cluster Aurora My SQL DB per pubblicare dati di log generali, lenti, di controllo e di errore in un gruppo di log in Amazon CloudWatch Logs. Ciò consente di visualizzare le tendenze, gestire i log se un host è interessato e creare una base di riferimento per prestazioni «normali» per identificare facilmente anomalie o modifiche. Per ulteriori informazioni, consulta Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch .
- Monitoraggio avanzato: per abilitare CloudWatch parametri Amazon aggiuntivi per un database Aurora SQL My, attiva Enhanced Monitoring. Quando crei o modifichi un cluster Aurora DB, seleziona Abilita monitoraggio avanzato. Ciò consente ad Aurora di pubblicare le metriche delle prestazioni su. CloudWatch Alcune delle metriche chiave disponibili includono l'utilizzo, le connessioni al database, CPU l'utilizzo dello storage e la latenza delle query. Queste possono aiutare a identificare i punti deboli in termini di prestazioni.

  La quantità di informazioni trasferite per un'istanza DB è direttamente proporzionale alla granularità definita per Enhanced Monitoring. Un intervallo di monitoraggio più piccolo comporta report più frequenti sui parametri del sistema operativo e aumenta i costi di monitoraggio. Per gestire i costi,

imposta granularità diverse per le diverse istanze del tuo. Account AWS La granularità predefinita alla creazione di un'istanza è di 60 secondi. Per ulteriori informazioni, consulta Costo di Enhanced Monitoring (monitoraggio avanzato).

- Performance Insights: puoi visualizzare tutte le metriche delle chiamate al database. Ciò include i blocchi del database, le attese e il numero di righe elaborate, tutti elementi che è possibile utilizzare per la risoluzione dei problemi. Quando crei o modifichi un cluster Aurora DB, seleziona Attiva Performance Insights. Per impostazione predefinita, Performance Insights ha un periodo di conservazione dei dati di 7 giorni, ma può essere personalizzato per analizzare le tendenze delle prestazioni a lungo termine. Per una conservazione superiore a 7 giorni, è necessario passare al livello a pagamento. Per ulteriori informazioni, consulta i prezzi di Performance Insights. È possibile impostare separatamente il periodo di conservazione dei dati per ogni istanza Aurora DB. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Le cause più comuni dei problemi di prestazioni del SQL database Aurora My

È possibile utilizzare i seguenti passaggi per risolvere i problemi di prestazioni nel database Aurora My. SQL Elenchiamo questi passaggi nell'ordine logico di indagine, ma non sono pensati per essere lineari. Una scoperta potrebbe passare da una fase all'altra, il che consente una serie di percorsi investigativi.

1. Carico di lavoro: comprendi il carico di lavoro del tuo database.
2. Registrazione: esamina tutti i log del database.
3. Connessioni al database: assicuratevi che le connessioni tra le applicazioni e il database siano affidabili.
4. Prestazioni delle query: esamina i piani di esecuzione delle query per vedere se sono cambiati. Le modifiche al codice possono causare la modifica dei piani.

Risoluzione dei problemi relativi al carico di lavoro per i database Aurora My SQL

Il carico di lavoro del database può essere visualizzato come lettura e scrittura. Una volta compreso il «normale» carico di lavoro del database, è possibile ottimizzare le query e il server del database per soddisfare la domanda man mano che questa cambia. Esistono diversi motivi per cui le prestazioni possono cambiare, quindi il primo passo è capire cosa è cambiato.

- È stato effettuato un aggiornamento della versione principale o secondaria?

  Un aggiornamento della versione principale include modifiche al codice del motore, in particolare nell'ottimizzatore, che possono modificare il piano di esecuzione delle query. Quando si aggiornano le versioni del database, in particolare le versioni principali, è molto importante analizzare il carico di lavoro del database e ottimizzarlo di conseguenza. L'ottimizzazione può comportare l'ottimizzazione e la riscrittura delle query o l'aggiunta e l'aggiornamento delle impostazioni dei parametri, a seconda dei risultati dei test. Capire cosa sta causando l'impatto ti consentirà di iniziare a concentrarti su quell'area specifica.

  Per ulteriori informazioni, consulta Novità in My SQL 8.0 e Variabili e opzioni di stato aggiunte, obsolete o rimosse in My SQL 8.0 nella documentazione La mia documentazione e. SQL Confronto tra Aurora My SQL versione 2 e Aurora My versione 3 SQL
- C'è stato un aumento dei dati in fase di elaborazione (numero di righe)?
- Ci sono più interrogazioni in esecuzione contemporaneamente?
- Sono state apportate modifiche allo schema o al database?
- Sono stati rilevati difetti o correzioni del codice?

Indice

- Metriche relative all'host dell'istanza
  - Utilizzo di CPU
  - Utilizzo della memoria
  - Throughput di rete
- Parametri del database
- Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL
  - Esempio 1: utilizzo continuo ed elevato della memoria
  - Esempio 2: picchi di memoria transitori

- Esempio 3: la memoria liberabile diminuisce continuamente e non viene recuperata
- Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL

Metriche relative all'host dell'istanza

Monitora le metriche relative all'host dell'istanzaCPU, ad esempio la memoria e l'attività di rete, per capire se c'è stata una modifica del carico di lavoro. Esistono due concetti principali per comprendere le modifiche del carico di lavoro:

- Utilizzo: l'utilizzo di un dispositivo, ad esempio CPU un disco. Può essere basato sul tempo o sulla capacità.
  - Basato sul tempo: la quantità di tempo in cui una risorsa è occupata in un determinato periodo di osservazione.
  - Basato sulla capacità: la quantità di velocità effettiva che un sistema o un componente è in grado di fornire, espressa in percentuale della sua capacità.
- Saturazione: il grado in cui una risorsa richiede più lavoro di quanto ne possa elaborare. Quando l'utilizzo basato sulla capacità raggiunge il 100%, il lavoro aggiuntivo non può essere elaborato e deve essere messo in coda.

Utilizzo di CPU

È possibile utilizzare i seguenti strumenti per identificare l'utilizzo e la saturazione: CPU

- CloudWatch fornisce la `CPUUtilization` metrica. Se raggiunge il 100%, l'istanza è satura. Tuttavia, le CloudWatch metriche vengono calcolate in media su 1 minuto e mancano di granularità.

  Per ulteriori informazioni sulle metriche, consulta. CloudWatch Parametri a livello di istanza per Amazon Aurora
- Enhanced Monitoring fornisce le metriche restituite dal comando del sistema `top` operativo. Mostra le medie di carico e i seguenti CPU stati, con una granularità di 1 secondo:
  - `Idle (%)`= Tempo di inattività
  - `IRQ (%)`= Interruzioni software
  - `Nice (%)`= Bel periodo per i processi con una buona priorità.
  - `Steal (%)`= Tempo dedicato al servizio di altri inquilini (legato alla virtualizzazione)

- `System (%)`= Ora del sistema
- `User (%)`= Ora dell'utente
- `Wait (%)`= Attesa I/O

Per ulteriori informazioni sulle metriche di Enhanced Monitoring, vedere. Parametri del sistema operativo per Aurora

Utilizzo della memoria

Se il sistema è sotto pressione in termini di memoria e il consumo di risorse sta raggiungendo la saturazione, si dovrebbe osservare un elevato grado di scansione, paginazione, scambio ed errori delle pagine. out-of-memory

È possibile utilizzare i seguenti strumenti per identificare l'utilizzo e la saturazione della memoria:

CloudWatch fornisce la `FreeableMemory` metrica che mostra quanta memoria può essere recuperata svuotando alcune cache del sistema operativo e la memoria attualmente disponibile.

Per ulteriori informazioni sulle metriche, consulta. CloudWatch Parametri a livello di istanza per Amazon Aurora

Enhanced Monitoring fornisce le seguenti metriche che possono aiutarti a identificare i problemi di utilizzo della memoria:

- `Buffers (KB)`— La quantità di memoria utilizzata per il buffering delle richieste di I/O prima della scrittura sul dispositivo di storage, in kilobyte.
- `Cached (KB)`— La quantità di memoria utilizzata per la memorizzazione nella cache degli I/O basati sul file system.
- `Free (KB)`— La quantità di memoria non assegnata, espressa in kilobyte.
- `Swap`— Memorizzata nella cache, gratuita e totale.

Ad esempio, se vedi che l'istanza DB utilizza `Swap` memoria, la quantità totale di memoria per il carico di lavoro è maggiore di quella attualmente disponibile sull'istanza. Ti consigliamo di aumentare le dimensioni dell'istanza DB o di ottimizzare il carico di lavoro per utilizzare meno memoria.

Per ulteriori informazioni sulle metriche di Enhanced Monitoring, consulta. Parametri del sistema operativo per Aurora

Per informazioni più dettagliate sull'utilizzo dello schema e `sys` dello schema delle prestazioni per determinare quali connessioni e componenti utilizzano la memoria, vedereRisoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL.

Throughput di rete

CloudWatch fornisce le seguenti metriche per la velocità di trasmissione totale della rete, tutte calcolate in media su 1 minuto:

- `NetworkReceiveThroughput`— La quantità di throughput di rete ricevuta dai client da ciascuna istanza nel cluster Aurora DB.
- `NetworkTransmitThroughput`— La quantità di throughput di rete inviata ai client da ciascuna istanza nel cluster Aurora DB.
- `NetworkThroughput`— La quantità di throughput di rete ricevuta e trasmessa ai client da ciascuna istanza del cluster Aurora DB.
- `StorageNetworkReceiveThroughput`— La quantità di throughput di rete ricevuta dal sottosistema di archiviazione Aurora da ciascuna istanza del cluster DB.
- `StorageNetworkTransmitThroughput`— La quantità di throughput di rete inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.
- `StorageNetworkThroughput`— La quantità di throughput di rete ricevuta e inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.

Per ulteriori informazioni sulle metriche, consulta. CloudWatch Parametri a livello di istanza per Amazon Aurora

Enhanced Monitoring fornisce i grafici `network` ricevuti (RX) e trasmessi (TX), con una granularità fino a 1 secondo.

Per ulteriori informazioni sulle metriche di Enhanced Monitoring, consulta. Parametri del sistema operativo per Aurora

Parametri del database

Esamina le seguenti CloudWatch metriche per le modifiche del carico di lavoro:

- `BlockedTransactions`— Il numero medio di transazioni bloccate nel database al secondo.
- `BufferCacheHitRatio`— La percentuale di richieste servite dalla buffer cache.
- `CommitThroughput`— Il numero medio di operazioni di commit al secondo.

- `DatabaseConnections`— Il numero di connessioni di rete del client all'istanza del database.
- `Deadlocks`— Il numero medio di deadlock nel database al secondo.
- `DMLThroughput`— Il numero medio di inserimenti, aggiornamenti ed eliminazioni al secondo.
- `ResultSetCacheHitRatio`— La percentuale di richieste servite dalla cache delle query.
- `RollbackSegmentHistoryListLength`— I registri di annullamento che registrano le transazioni impegnate con record contrassegnati da eliminare.
- `RowLockTime`— Il tempo totale impiegato per l'acquisizione di blocchi di riga per le tabelle InnoDB.
- `SelectThroughput`— Il numero medio di query selezionate al secondo.

Per ulteriori informazioni sulle CloudWatch metriche, vedere. Parametri a livello di istanza per Amazon Aurora

Considerate le seguenti domande quando esaminate il carico di lavoro:

1. Sono state apportate modifiche recenti nella classe dell'istanza DB, ad esempio la riduzione della dimensione dell'istanza da 8xlarge a 4xlarge o il passaggio da db.r5 a db.r6?
2. È possibile creare un clone e riprodurre il problema o si verifica solo su quell'istanza?
3. C'è un esaurimento delle risorse del server, un elevato CPU o un esaurimento della memoria? In caso affermativo, ciò potrebbe significare che è necessario hardware aggiuntivo.
4. Una o più domande richiedono più tempo?
5. Le modifiche sono causate da un aggiornamento, in particolare da un aggiornamento della versione principale? In caso affermativo, confronta le metriche precedenti e successive all'aggiornamento.
6. Sono state apportate modifiche al numero di istanze DB di Reader?
7. Hai abilitato la registrazione generale, di controllo o binaria? Per ulteriori informazioni, consulta Registrazione per i database Aurora MySQL.
8. Hai abilitato, disabilitato o modificato l'uso della replica dei log binari (binlog)?
9. Esistono transazioni di lunga durata con un gran numero di blocchi di righe? Esamina la lunghezza dell'elenco della cronologia di InnoDB (HLL) per le indicazioni sulle transazioni di lunga durata.

   Per ulteriori informazioni, consulta La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo il post sul blog Perché la mia SELECT query viene eseguita lentamente sul mio cluster Amazon Aurora My SQL DB? .

a. Se un errore di grandi dimensioni HLL è causato da una transazione di scrittura, significa che `UNDO` i log si accumulano (non vengono puliti regolarmente). In una transazione di scrittura di grandi dimensioni, questo accumulo può crescere rapidamente. In MySQL, `UNDO` viene memorizzato nel SYSTEMtablespace. Il `SYSTEM` tablespace non è restringibile. Il `UNDO` log potrebbe far crescere il `SYSTEM` tablespace fino a diversi GB o addirittura TB. Dopo l'eliminazione, rilascia lo spazio allocato eseguendo un backup logico (dump) dei dati, quindi importa il dump in una nuova istanza DB.

b. Se un errore di grandi dimensioni HLL è causato da una transazione di lettura (query a esecuzione prolungata), può significare che la query sta utilizzando una grande quantità di spazio temporaneo. Rilascia lo spazio temporaneo riavviando. Esamina le metriche di Performance Insights DB per eventuali modifiche nella `Temp` sezione, ad esempio`created_tmp_tables`. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

10. È possibile suddividere le transazioni di lunga durata in transazioni più piccole che modificano un minor numero di righe?

11. Ci sono cambiamenti nelle transazioni bloccate o aumenti delle situazioni di stallo? Esamina le metriche di Performance Insights DB per eventuali modifiche alle variabili di stato nella `Locks` sezione, ad esempio `innodb_row_lock_time innodb_row_lock_waits`, e `innodb_dead_locks`. Utilizza intervalli di 1 minuto o 5 minuti.

12. I tempi di attesa sono aumentati? Esamina gli eventi di attesa e i tipi di attesa di Performance Insights utilizzando intervalli di 1 minuto o 5 minuti. Analizza i principali eventi di attesa e verifica se sono correlati alle modifiche del carico di lavoro o ai conflitti del database. Ad esempio, `buf_pool mutex` indica la contesa del pool di buffer. Per ulteriori informazioni, consulta Regolazione di Aurora MySQL con eventi di attesa.

Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL

Sebbene CloudWatch Enhanced Monitoring e Performance Insights forniscano una buona panoramica dell'utilizzo della memoria a livello di sistema operativo, ad esempio la quantità di memoria utilizzata dal processo del database, non consentono di analizzare quali connessioni o componenti all'interno del motore potrebbero causare questo utilizzo di memoria.

Per risolvere questo problema, è possibile utilizzare lo schema e `sys` lo schema delle prestazioni. In Aurora MySQL versione 3, la strumentazione di memoria è abilitata per impostazione predefinita quando lo schema delle prestazioni è abilitato. In Aurora MySQL versione 2, per impostazione predefinita è abilitata solo la strumentazione di memoria per l'utilizzo della memoria Performance

Schema. Per informazioni sulle tabelle disponibili nello schema delle prestazioni per tenere traccia dell'utilizzo della memoria e abilitare la strumentazione di memoria dello schema di prestazioni, vedere Tabelle di riepilogo della memoria nella documentazione di MySQL. Per ulteriori informazioni sull'utilizzo del Performance Schema con Performance Insights, vedere Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

Sebbene nello schema delle prestazioni siano disponibili informazioni dettagliate per tenere traccia dell'utilizzo corrente della memoria, lo schema sys di MySQL offre viste sulle tabelle dello schema delle prestazioni che è possibile utilizzare per individuare rapidamente dove viene utilizzata la memoria.

Nello `sys` schema, sono disponibili le seguenti viste per tenere traccia dell'utilizzo della memoria per connessione, componente e query.

Vista	Descrizione
memory_by_host_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore da parte dell'host. Ciò può essere utile per identificare quali server applicativi o host client stanno consumando memoria.
memory_by_thread_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore in base all'ID del thread. L'ID del thread in MySQL può essere una connessione client o un thread in background. È possibile mappare il thread IDs alla IDs connessione MySQL utilizzando la vista sys.processlist o la tabella performance_schema.threads.
memory_by_user_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore da parte dell'utente. Ciò può essere utile per identificare quali account utente o client stanno consumando memoria.
memory_global_by_current_bytes	Fornisce informazioni sull'utilizzo della memoria del motore per componente del motore. Ciò può essere utile per identificare l'utilizzo della memoria a livello globale in base ai buffer

Vista	Descrizione
	o ai componenti del motore. Ad esempio, potresti vedere l'`memory/innodb/buf_buf_pool` evento per il buffer pool InnoDB o l'`memory/sql/Prepared_statement::main_mem_root` evento per le istruzioni preparate.
memory_global_total	Fornisce una panoramica dell'utilizzo totale della memoria tracciata nel motore di database.

In Aurora MySQL versione 3.05 e successive, puoi anche tenere traccia dell'utilizzo massimo della memoria tramite statement digest nelle tabelle di riepilogo delle istruzioni Performance Schema. Le tabelle di riepilogo delle istruzioni contengono riassunti di istruzioni normalizzati e statistiche aggregate sulla loro esecuzione. La `MAX_TOTAL_MEMORY` colonna può aiutarti a identificare la quantità massima di memoria utilizzata da Query Digest dall'ultima reimpostazione delle statistiche o dal riavvio dell'istanza del database. Ciò può essere utile per identificare query specifiche che potrebbero consumare molta memoria.

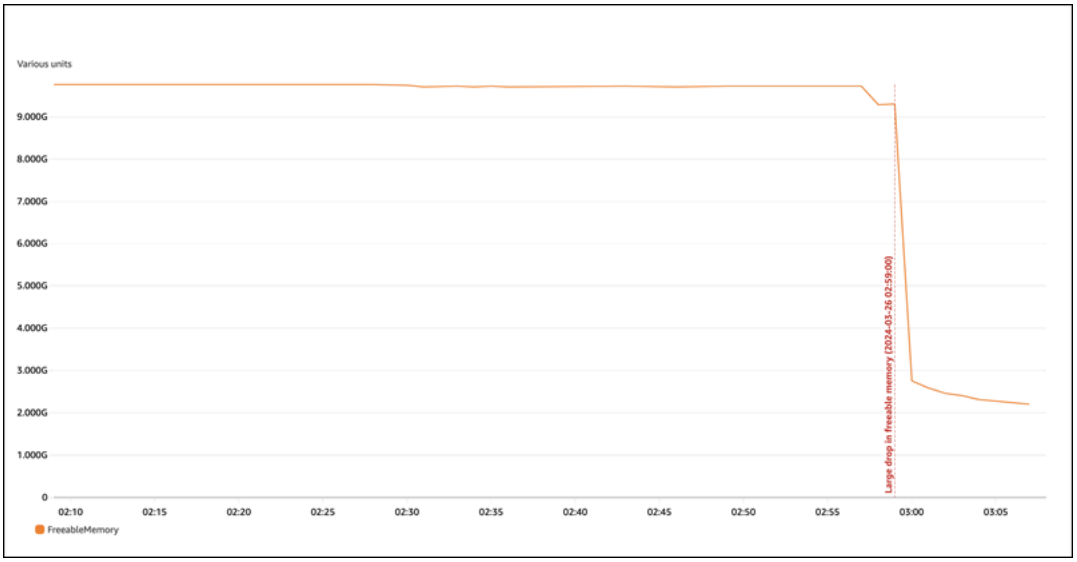
> **Note**
>
> Lo schema e lo `sys` schema delle prestazioni mostrano l'utilizzo corrente della memoria sul server e i livelli massimi di memoria consumata per ogni componente del motore e della connessione. Poiché lo schema delle prestazioni viene mantenuto in memoria, le informazioni vengono reimpostate al riavvio dell'istanza DB. Per mantenere una cronologia nel tempo, si consiglia di configurare il recupero e l'archiviazione di questi dati al di fuori dello schema delle prestazioni.

Argomenti

- Esempio 1: utilizzo continuo ed elevato della memoria
- Esempio 2: picchi di memoria transitori
- Esempio 3: la memoria liberabile diminuisce continuamente e non viene recuperata

Esempio 1: utilizzo continuo ed elevato della memoria

Guardando `FreeableMemory` a livello globale CloudWatch, possiamo vedere che l'utilizzo della memoria è aumentato notevolmente alle 02:59 UTC del 26/03/2020.



Questo non ci dice il quadro completo. Per determinare quale componente utilizza più memoria, puoi accedere al database e guardare`sys.memory_global_by_current_bytes`. Questa tabella contiene un elenco di eventi di memoria che MySQL tiene traccia, insieme a informazioni sull'allocazione della memoria per evento. Ogni evento di tracciamento della memoria inizia con`memory/%`, seguito da altre informazioni sul componente/funzionalità del motore a cui è associato l'evento.

Ad esempio, `memory/performance_schema/%` è per gli eventi di memoria relativi allo schema delle prestazioni, `memory/innodb/%` è per InnoDB e così via. Per ulteriori informazioni sulle convenzioni di denominazione degli eventi, vedere Convenzioni di [denominazione degli strumenti Performance Schema nella documentazione MySQL](#).

Dalla seguente interrogazione, possiamo trovare il probabile colpevole in base `acurrent_alloc`, ma possiamo anche vedere molti eventi. `memory/performance_schema/%`

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+-----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
```

```
+-----------------------------------------------------------------------------
+---------------+---------------+------------------+------------+------------
+----------------+
| memory/sql/Prepared_statement::main_mem_root                                |
 512817 | 4.91 GiB      | 10.04 KiB         |     512823 | 4.91 GiB   | 10.04 KiB
 |
| memory/performance_schema/prepared_statements_instances                     |
   252 | 488.25 MiB    | 1.94 MiB          |        252 | 488.25 MiB | 1.94 MiB       |
| memory/innodb/hash0hash                                                     |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error          |
  1028 | 52.27 MiB     | 52.06 KiB         |       1028 | 52.27 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     4 | 47.25 MiB     | 11.81 MiB         |          4 | 47.25 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest               |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/performance_schema/memory_summary_by_thread_by_event_name            |
     4 | 31.64 MiB     | 7.91 MiB          |          4 | 31.64 MiB  | 7.91 MiB       |
| memory/innodb/memory                                                        |
 15227 | 27.44 MiB     | 1.85 KiB          |      20619 | 33.33 MiB  | 1.66 KiB       |
| memory/sql/String::value                                                    |
 74411 | 21.85 MiB     |  307 bytes        |      76867 | 25.54 MiB  |  348 bytes     |
| memory/sql/TABLE                                                            |
  8381 | 21.03 MiB     | 2.57 KiB          |       8381 | 21.03 MiB  | 2.57 KiB       |
+-----------------------------------------------------------------------------
+---------------+---------------+------------------+------------+------------
+----------------+
10 rows in set (0.02 sec)
```

Abbiamo detto in precedenza che lo schema delle prestazioni è archiviato in memoria, il che significa che viene anche registrato nella strumentazione di memoria. `performance_schema`

> **Note**
>
> Se riscontri che il Performance Schema utilizza molta memoria e desideri limitarne l'utilizzo, puoi regolare i parametri del database in base alle tue esigenze. Per ulteriori informazioni, vedere Il modello di allocazione della memoria Performance Schema nella documentazione di MySQL.

Per motivi di leggibilità, è possibile eseguire nuovamente la stessa query ma escludere gli eventi di Performance Schema. L'output mostra quanto segue:

- Il principale consumatore di memoria è`memory/sql/Prepared_statement::main_mem_root`.
- La `current_alloc` colonna ci dice che MySQL ha 4,91 GiB attualmente assegnati a questo evento.
- Ci `high_alloc column` dice che 4,91 GiB è il limite massimo dall'ultima reimpostazione `current_alloc` delle statistiche o dal riavvio del server. Ciò significa che `memory/sql/Prepared_statement::main_mem_root` è al suo valore massimo.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name NOT LIKE
 'memory/performance_schema/%' LIMIT 10;

+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root  |        512817 | 4.91 GiB      | 10.04
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
| memory/innodb/hash0hash                       |             4 | 79.07 MiB     | 19.77
 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/memory                          |         17096 | 31.68 MiB     | 1.90
 KiB          |      22498 | 37.60 MiB  | 1.71 KiB       |
| memory/sql/String::value                      |        122277 | 27.94 MiB     |  239
 bytes        |     124699 | 29.47 MiB  |  247 bytes     |
| memory/sql/TABLE                              |          9927 | 24.67 MiB     | 2.55
 KiB          |       9929 | 24.68 MiB  | 2.55 KiB       |
| memory/innodb/lock0lock                       |          8888 | 19.71 MiB     | 2.27
 KiB          |       8888 | 19.71 MiB  | 2.27 KiB       |
| memory/sql/Prepared_statement::infrastructure |        257623 | 16.24 MiB     |   66
 bytes        |     257631 | 16.24 MiB  |   66 bytes     |
| memory/mysys/KEY_CACHE                        |             3 | 16.00 MiB     | 5.33
 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/innodb/sync0arr                        |             3 | 7.03 MiB      | 2.34
 MiB          |          3 | 7.03 MiB   | 2.34 MiB       |
| memory/sql/THD::main_mem_root                 |           815 | 6.56 MiB      | 8.24
 KiB          |        849 | 7.19 MiB   | 8.67 KiB       |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
10 rows in set (0.06 sec)
```

Dal nome dell'evento, possiamo dire che questa memoria viene utilizzata per le dichiarazioni preparate. Se vuoi vedere quali connessioni utilizzano questa memoria, puoi controllare memory_by_thread_by_current_bytes.

Nell'esempio seguente, a ogni connessione sono allocati circa 7 MiB, con un limite massimo di circa 6,29 MiB (). `current_max_alloc` Ciò ha senso, perché l'esempio utilizza 80 tabelle e 800 `sysbench` connessioni con istruzioni preparate. Se si desidera ridurre l'utilizzo della memoria in questo scenario, è possibile ottimizzare l'utilizzo da parte dell'applicazione delle istruzioni preparate per ridurre il consumo di memoria.

```
mysql> SELECT * FROM sys.memory_by_thread_by_current_bytes;

+-----------+-------------------------------------------+--------------------
+------------------+------------------+------------------+-----------------+
| thread_id | user                                      | current_count_used |
 current_allocated | current_avg_alloc | current_max_alloc | total_allocated |
+-----------+-------------------------------------------+--------------------
+------------------+------------------+------------------+-----------------+
|        46 | rdsadmin@localhost                        |                405 | 8.47 MiB
          | 21.42 KiB         | 8.00 MiB          | 155.86 MiB      |
|        61 | reinvent@10.0.4.4                         |               1749 | 6.72 MiB
          | 3.93 KiB          | 6.29 MiB          | 14.24 MiB       |
|       101 | reinvent@10.0.4.4                         |               1845 | 6.71 MiB
          | 3.72 KiB          | 6.29 MiB          | 14.50 MiB       |
|        55 | reinvent@10.0.4.4                         |               1674 | 6.68 MiB
          | 4.09 KiB          | 6.29 MiB          | 14.13 MiB       |
|        57 | reinvent@10.0.4.4                         |               1416 | 6.66 MiB
          | 4.82 KiB          | 6.29 MiB          | 13.52 MiB       |
|       112 | reinvent@10.0.4.4                         |               1759 | 6.66 MiB
          | 3.88 KiB          | 6.29 MiB          | 14.17 MiB       |
|        66 | reinvent@10.0.4.4                         |               1428 | 6.64 MiB
          | 4.76 KiB          | 6.29 MiB          | 13.47 MiB       |
|        75 | reinvent@10.0.4.4                         |               1389 | 6.62 MiB
          | 4.88 KiB          | 6.29 MiB          | 13.40 MiB       |
|       116 | reinvent@10.0.4.4                         |               1333 | 6.61 MiB
          | 5.08 KiB          | 6.29 MiB          | 13.21 MiB       |
|        90 | reinvent@10.0.4.4                         |               1448 | 6.59 MiB
          | 4.66 KiB          | 6.29 MiB          | 13.58 MiB       |
|        98 | reinvent@10.0.4.4                         |               1440 | 6.57 MiB
          | 4.67 KiB          | 6.29 MiB          | 13.52 MiB       |
|        94 | reinvent@10.0.4.4                         |               1433 | 6.57 MiB
          | 4.69 KiB          | 6.29 MiB          | 13.49 MiB       |
```

```
|        62 | reinvent@10.0.4.4                          |              1323 | 6.55 MiB
         | 5.07 KiB          | 6.29 MiB          | 13.48 MiB       |
|        87 | reinvent@10.0.4.4                          |              1323 | 6.55 MiB
         | 5.07 KiB          | 6.29 MiB          | 13.25 MiB       |
|        99 | reinvent@10.0.4.4                          |              1346 | 6.54 MiB
         | 4.98 KiB          | 6.29 MiB          | 13.24 MiB       |
|       105 | reinvent@10.0.4.4                          |              1347 | 6.54 MiB
         | 4.97 KiB          | 6.29 MiB          | 13.34 MiB       |
|        73 | reinvent@10.0.4.4                          |              1335 | 6.54 MiB
         | 5.02 KiB          | 6.29 MiB          | 13.23 MiB       |
|        54 | reinvent@10.0.4.4                          |              1510 | 6.53 MiB
         | 4.43 KiB          | 6.29 MiB          | 13.49 MiB       |
.                                                                                .
.                                                                                .
.                                                                                .
|       812 | reinvent@10.0.4.4                          |              1259 | 6.38 MiB
         | 5.19 KiB          | 6.29 MiB          | 13.05 MiB       |
|       214 | reinvent@10.0.4.4                          |              1279 | 6.38 MiB
         | 5.10 KiB          | 6.29 MiB          | 12.90 MiB       |
|       325 | reinvent@10.0.4.4                          |              1254 | 6.38 MiB
         | 5.21 KiB          | 6.29 MiB          | 12.99 MiB       |
|       705 | reinvent@10.0.4.4                          |              1273 | 6.37 MiB
         | 5.13 KiB          | 6.29 MiB          | 13.03 MiB       |
|       530 | reinvent@10.0.4.4                          |              1268 | 6.37 MiB
         | 5.15 KiB          | 6.29 MiB          | 12.92 MiB       |
|       307 | reinvent@10.0.4.4                          |              1263 | 6.37 MiB
         | 5.17 KiB          | 6.29 MiB          | 12.87 MiB       |
|       738 | reinvent@10.0.4.4                          |              1260 | 6.37 MiB
         | 5.18 KiB          | 6.29 MiB          | 13.00 MiB       |
|       819 | reinvent@10.0.4.4                          |              1252 | 6.37 MiB
         | 5.21 KiB          | 6.29 MiB          | 13.01 MiB       |
|        31 | innodb/srv_purge_thread                    |             17810 | 3.14 MiB
         |  184 bytes        | 2.40 MiB          | 205.69 MiB      |
|        38 | rdsadmin@localhost                         |               599 | 1.76 MiB
         | 3.01 KiB          | 1.00 MiB          | 25.58 MiB       |
|         1 | sql/main                                   |              3756 | 1.32 MiB
         |  367 bytes        | 355.78 KiB        | 6.19 MiB        |
|       854 | rdsadmin@localhost                         |                46 | 1.08 MiB
         | 23.98 KiB         | 1.00 MiB          | 5.10 MiB        |
|        30 | innodb/clone_gtid_thread                   |              1596 | 573.14
 KiB        |  367 bytes        | 254.91 KiB        | 970.69 KiB      |
```

```
|        40 | rdsadmin@localhost                          |                235 | 245.19
 KiB        | 1.04 KiB          | 128.88 KiB        | 808.64 KiB      |
|       853 | rdsadmin@localhost                          |                 96 | 94.63
 KiB         | 1009 bytes        | 29.73 KiB         | 422.45 KiB      |
|        36 | rdsadmin@localhost                          |                 33 | 36.29
 KiB         | 1.10 KiB          | 16.08 KiB         | 74.15 MiB       |
|        33 | sql/event_scheduler                         |                  3 | 16.27
 KiB         | 5.42 KiB          | 16.04 KiB         | 16.27 KiB       |
|        35 | sql/compress_gtid_table                     |                  8 | 14.20
 KiB         | 1.77 KiB          | 8.05 KiB          | 18.62 KiB       |
|        25 | innodb/fts_optimize_thread                  |                 12 | 1.86 KiB
          |  158 bytes        |  648 bytes        | 1.98 KiB        |
|        23 | innodb/srv_master_thread                    |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 24.40 KiB       |
|        24 | innodb/dict_stats_thread                    |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 1.35 KiB        |
|         5 | innodb/io_read_thread                       |                  1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         6 | innodb/io_read_thread                       |                  1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         2 | sql/aws_oscar_log_level_monitor             |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         4 | innodb/io_ibuf_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         7 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         8 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         9 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        10 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        11 | innodb/srv_lra_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        12 | innodb/srv_akp_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        18 | innodb/srv_lock_timeout_thread              |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |  248 bytes      |
|        19 | innodb/srv_error_monitor_thread             |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        20 | innodb/srv_monitor_thread                   |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        21 | innodb/buf_resize_thread                    |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
```

```
|        22 | innodb/btr_search_sys_toggle_thread       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        32 | innodb/dict_persist_metadata_table_thread |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        34 | sql/signal_handler                        |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
+-----------+-------------------------------------------+-------------------
+-------------------+------------------+------------------+-----------------+
831 rows in set (2.48 sec)
```

Come accennato in precedenza, il valore dell'ID del thread (`thd_id`) qui può riferirsi ai thread in background del server o alle connessioni al database. Se vuoi mappare i valori dell'ID del thread alla connessione al database IDs, puoi usare la `performance_schema.threads` tabella o la `sys.processlist` vista, `conn_id` dov'è l'ID di connessione.

```
mysql> SELECT thd_id,conn_id,user,db,command,state,time,last_wait FROM sys.processlist
 WHERE user='reinvent@10.0.4.4';

+--------+---------+------------------+----------+---------+----------------+------
+-------------------------------------------------+
| thd_id | conn_id | user             | db       | command | state          | time |
 last_wait                                        |
+--------+---------+------------------+----------+---------+----------------+------
+-------------------------------------------------+
|    590 |     562 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                           |
|    578 |     550 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                             |
|    579 |     551 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                           |
|    580 |     552 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                        |
|    581 |     553 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                        |
|    582 |     554 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                             |
|    583 |     555 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                             |
|    584 |     556 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                        |
|    585 |     557 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                           |
```

```
|    586 |     558 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    587 |     559 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                             |
.
                                                   .
.
                                                   .
.
                                                   .
|    323 |     295 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                               |
|    324 |     296 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    325 |     297 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                             |
|    326 |     298 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                          |
|    438 |     410 | reinvent@10.0.4.4 | sysbench | Execute | System lock    |    0 |
 wait/lock/table/sql/handler                        |
|    280 |     252 | reinvent@10.0.4.4 | sysbench | Sleep   | starting       |    0 |
 wait/io/socket/sql/client_connection               |
|     98 |      70 | reinvent@10.0.4.4 | sysbench | Query   | freeing items  |    0 |
 NULL                                               |
+--------+---------+-------------------+----------+---------+----------------+------
+-------------------------------------------------+
804 rows in set (5.51 sec)
```

Ora interrompiamo il `sysbench` carico di lavoro, che chiude le connessioni e libera la memoria. Ricontrollando gli eventi, possiamo confermare che la memoria è stata liberata, ma ci dice `high_alloc` comunque qual è il limite massimo. La `high_alloc` colonna può essere molto utile per identificare brevi picchi di utilizzo della memoria, dai quali potrebbe non essere possibile identificare immediatamente l'utilizzo`current_alloc`, poiché mostra solo la memoria attualmente allocata.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+---------------+--------------
+------------------+-----------+------------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
```

```
+----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
+----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Se si desidera eseguire il ripristinohigh_alloc, è possibile troncare le tabelle di riepilogo della performance_schema memoria, ma in questo modo viene ripristinata tutta la strumentazione della memoria. Per ulteriori informazioni, vedere le caratteristiche generali della tabella dello schema delle prestazioni nella documentazione di MySQL.

Nell'esempio seguente, possiamo vedere che high_alloc viene ripristinato dopo il troncamento.

```
mysql> TRUNCATE `performance_schema`.`memory_summary_global_by_event_name`;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

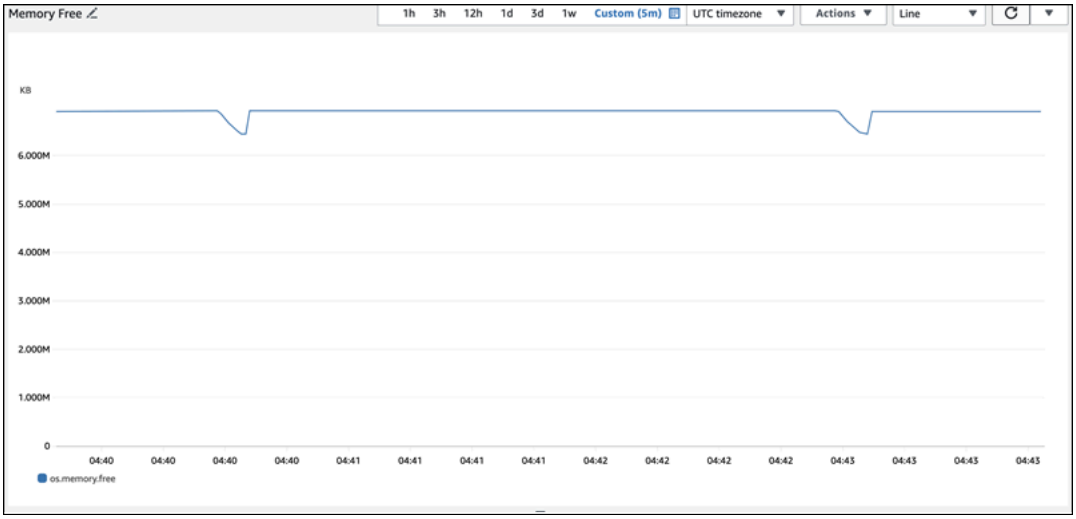
+----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                   | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |         17 | 253.80 KiB | 14.93 KiB      |
+----------------------------------------------+---------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Esempio 2: picchi di memoria transitori

Un altro evento comune sono i brevi picchi di utilizzo della memoria su un server di database. Possono trattarsi di cadute periodiche della memoria liberabilesys.memory_global_by_current_bytes, difficili da risolvere current_alloc in quanto la memoria è già stata liberata.

> **Note**
>
> Se le statistiche dello schema delle prestazioni sono state reimpostate o l'istanza del database è stata riavviata, queste informazioni non saranno disponibili in o p. `sys` `erformance_schema` Per conservare queste informazioni, ti consigliamo di configurare la raccolta di metriche esterne.

Il seguente grafico della `os.memory.free` metrica di Enhanced Monitoring mostra brevi picchi di 7 secondi nell'utilizzo della memoria. Enhanced Monitoring consente di eseguire il monitoraggio a intervalli di appena 1 secondo, il che è perfetto per rilevare picchi transitori come questi.



Per aiutare a diagnosticare la causa dell'utilizzo della memoria, possiamo utilizzare una combinazione delle viste di riepilogo della `sys` memoria e delle [tabelle di `high_alloc` riepilogo delle istruzioni Performance Schema per cercare di identificare le sessioni e le](#) connessioni che causano problemi.

Come previsto, poiché l'utilizzo della memoria non è attualmente elevato, nella visualizzazione `sys` dello schema sottostante non è possibile individuare i principali autori di violazioni. `current_alloc`

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+-----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
```

```
+-------------------------------------------------------------------------------
+---------------+---------------+------------------+------------+------------
+----------------+
| memory/innodb/hash0hash                                                      |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/os0event                                                       |
 439372 | 60.34 MiB     |  144 bytes        |     439372 | 60.34 MiB  |  144 bytes
 |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/mysys/KEY_CACHE                                                       |
     3 | 16.00 MiB     | 5.33 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/performance_schema/events_statements_history_long                     |
     1 | 14.34 MiB     | 14.34 MiB         |          1 | 14.34 MiB  | 14.34 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error           |
   257 | 13.07 MiB     | 52.06 KiB         |        257 | 13.07 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     1 | 11.81 MiB     | 11.81 MiB         |          1 | 11.81 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.digest_text         |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.sql_text            |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
+-------------------------------------------------------------------------------
+---------------+---------------+------------------+------------+------------
+----------------+
10 rows in set (0.01 sec)
```

Espandendo la visualizzazione in base all'ordine`high_alloc`, ora possiamo vedere che il `memory/temptable/physical_ram` componente è un ottimo candidato in questo caso. Al suo massimo, ha consumato 515,00 MiB.

Come suggerisce il nome, l'utilizzo della memoria `memory/temptable/physical_ram` degli strumenti per il motore di TEMP archiviazione in MySQL, introdotto in MySQL 8.0. Per ulteriori informazioni su come MySQL utilizza le tabelle temporanee, vedere Uso interno delle tabelle temporanee in MySQL nella documentazione di MySQL.

> **Note**
>
> Stiamo usando la vista in questo esempio. `sys.x$memory_global_by_current_bytes`

```
mysql> SELECT event_name, format_bytes(current_alloc) AS "currently allocated",
 sys.format_bytes(high_alloc) AS "high-water mark"
FROM sys.x$memory_global_by_current_bytes ORDER BY high_alloc DESC LIMIT 10;

+-------------------------------------------------------------------------------
+--------------------+-----------------+
| event_name                                                                    |
 currently allocated | high-water mark |
+-------------------------------------------------------------------------------
+--------------------+-----------------+
| memory/temptable/physical_ram                                                 | 4.00
 MiB            | 515.00 MiB      |
| memory/innodb/hash0hash                                                       | 79.07
 MiB           | 79.07 MiB       |
| memory/innodb/os0event                                                        | 63.95
 MiB           | 63.95 MiB       |
| memory/performance_schema/events_statements_summary_by_digest                | 40.28
 MiB           | 40.28 MiB       |
| memory/mysys/KEY_CACHE                                                        | 16.00
 MiB           | 16.00 MiB       |
| memory/performance_schema/events_statements_history_long                      | 14.34
 MiB           | 14.34 MiB       |
| memory/performance_schema/events_errors_summary_by_thread_by_error           | 13.07
 MiB           | 13.07 MiB       |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name | 11.81
 MiB           | 11.81 MiB       |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    | 9.77
 MiB            | 9.77 MiB        |
| memory/performance_schema/events_statements_history_long.sql_text            | 9.77
 MiB            | 9.77 MiB        |
+-------------------------------------------------------------------------------
+--------------------+-----------------+
10 rows in set (0.00 sec)
```

NelEsempio 1: utilizzo continuo ed elevato della memoria, abbiamo controllato l'utilizzo corrente della memoria per ogni connessione per determinare quale connessione è responsabile dell'utilizzo della memoria in questione. In questo esempio, la memoria è già stata liberata, quindi non è utile controllare l'utilizzo della memoria per le connessioni correnti.

Per approfondire e trovare le dichiarazioni, gli utenti e gli host offensivi, utilizziamo il Performance Schema. Lo schema delle prestazioni contiene più tabelle di riepilogo delle istruzioni suddivise in diverse dimensioni, come nome dell'evento, statement digest, host, thread e utente. Ogni

visualizzazione ti consentirà di approfondire dove vengono eseguite determinate istruzioni e cosa stanno facendo. Questa sezione è dedicata a`MAX_TOTAL_MEMORY`, ma è possibile trovare ulteriori informazioni su tutte le colonne disponibili nella documentazione delle tabelle di riepilogo delle istruzioni dello schema delle prestazioni.

```
mysql> SHOW TABLES IN performance_schema LIKE 'events_statements_summary_%';

+-----------------------------------------------------------+
| Tables_in_performance_schema (events_statements_summary_%) |
+-----------------------------------------------------------+
| events_statements_summary_by_account_by_event_name         |
| events_statements_summary_by_digest                        |
| events_statements_summary_by_host_by_event_name            |
| events_statements_summary_by_program                       |
| events_statements_summary_by_thread_by_event_name          |
| events_statements_summary_by_user_by_event_name            |
| events_statements_summary_global_by_event_name             |
+-----------------------------------------------------------+
7 rows in set (0.00 sec)
```

Per prima cosa controlliamo `events_statements_summary_by_digest` per vedere`MAX_TOTAL_MEMORY`.

Da ciò possiamo vedere quanto segue:

- La query con digest `20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a` sembra essere un buon candidato per questo utilizzo della memoria. Il `MAX_TOTAL_MEMORY` numero è 537450710, che corrisponde al picco massimo registrato durante l'evento. `memory/temptable/physical_ram` `sys.x$memory_global_by_current_bytes`
- È stato eseguito quattro volte (`COUNT_STAR`), la prima alle 04:08:34.943 256 del 26/03/2020 e l'ultima alle 04:43:06.998 310 del 2024-03-26.

```
mysql> SELECT SCHEMA_NAME,DIGEST,COUNT_STAR,MAX_TOTAL_MEMORY,FIRST_SEEN,LAST_SEEN
FROM performance_schema.events_statements_summary_by_digest ORDER BY MAX_TOTAL_MEMORY
 DESC LIMIT 5;

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

```
| SCHEMA_NAME | DIGEST                                                
```

```
1 row in set (0.01 sec)
```

Per ulteriori informazioni sulla `events_statements_summary_by_digest` tabella e su altre tabelle di riepilogo delle istruzioni Performance Schema, vedere Tabelle di riepilogo delle dichiarazioni nella documentazione di MySQL.

Puoi anche eseguire un'istruzione EXPLAIN o EXPLAIN ANALYZE per visualizzare maggiori dettagli.

> **Note**
>
> `EXPLAIN ANALYZE`può fornire più informazioni di`EXPLAIN`, ma esegue anche la query, quindi fai attenzione.

```
-- EXPLAIN
mysql> EXPLAIN WITH RECURSIVE cte (n) AS (SELECT 1  FROM sbtest1 UNION ALL SELECT id +
 1 FROM sbtest1) SELECT * FROM cte;

+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
| id | select_type | table      | partitions | type  | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra       |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | ALL   | NULL          | NULL | NULL    |
 NULL | 19221520 |   100.00 | NULL        |
|  2 | DERIVED     | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
|  3 | UNION       | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

-- EXPLAIN format=tree
mysql> EXPLAIN format=tree WITH RECURSIVE cte (n) AS (SELECT 1 FROM sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
```

```
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
1 row in set (0.00 sec)

-- EXPLAIN ANALYZE
mysql> EXPLAIN ANALYZE WITH RECURSIVE cte (n) AS (SELECT 1 from sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6) (actual
 time=6666..9201 rows=20e+6 loops=1)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6) (actual
 time=6666..6666 rows=20e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0365..2006 rows=10e+6 loops=1)
        -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0311..2494 rows=10e+6 loops=1)
1 row in set (10.53 sec)
```

Ma chi lo gestiva? Possiamo vedere nello schema delle prestazioni che l'`destructive_operator`utente aveva `MAX_TOTAL_MEMORY` di 537450710, che corrisponde ancora una volta ai risultati precedenti.

> **Note**
>
> Lo schema delle prestazioni è archiviato in memoria, quindi non deve essere considerato l'unica fonte per il controllo. Se è necessario mantenere una cronologia delle istruzioni eseguite e da quali utenti, si consiglia di abilitare Aurora Advanced Auditing. Se è inoltre necessario conservare le informazioni sull'utilizzo della memoria, si consiglia di configurare il monitoraggio per esportare e archiviare questi valori.

```
mysql> SELECT USER,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_user_by_event_name
ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+----------------------+---------------------------+------------+------------------+
| USER                 | EVENT_NAME                | COUNT_STAR | MAX_TOTAL_MEMORY |
+----------------------+---------------------------+------------+------------------+
| destructive_operator | statement/sql/select      |          4 |        537450710 |
| rdsadmin             | statement/sql/select      |       4172 |          3290981 |
| rdsadmin             | statement/sql/show_tables |          2 |          3615821 |
```

```
| rdsadmin             | statement/sql/show_fields |          2 |          3459965 |
| rdsadmin             | statement/sql/show_status |         75 |          1914976 |
+----------------------+---------------------------+------------+------------------+
5 rows in set (0.00 sec)

mysql> SELECT HOST,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_host_by_event_name
WHERE HOST != 'localhost' AND COUNT_STAR>0 ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+------------+----------------------+------------+------------------+
| HOST       | EVENT_NAME           | COUNT_STAR | MAX_TOTAL_MEMORY |
+------------+----------------------+------------+------------------+
| 10.0.8.231 | statement/sql/select |          4 |        537450710 |
+------------+----------------------+------------+------------------+
1 row in set (0.00 sec)
```

Esempio 3: la memoria liberabile diminuisce continuamente e non viene recuperata

Il motore di database InnoDB utilizza una serie di eventi di tracciamento della memoria specializzati per diversi componenti. Questi eventi specifici consentono il tracciamento granulare dell'utilizzo della memoria nei principali sottosistemi InnoDB, ad esempio:

- `memory/innodb/buf0buf`— Dedicato al monitoraggio delle allocazioni di memoria per il pool di buffer InnoDB.
- `memory/innodb/ibuf0ibuf`— Tiene traccia in particolare delle modifiche alla memoria relative al buffer di modifica di InnoDB.

Per identificare i principali consumatori di memoria, possiamo interrogare: `sys.memory_global_by_current_bytes`

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.28 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
```

```
| memory/innodb/fil0fil                                              | 198.49 MiB    |
| memory/sql/binlog_io_cache                                         | 128.00 MiB    |
| memory/innodb/mem0mem                                              | 96.82 MiB     |
| memory/innodb/dict0dict                                            | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                         | 88.00 MiB     |
+--------------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

I risultati mostrano che `memory/innodb/memory` è il principale consumatore, utilizzando 5,28 GiB di memoria attualmente allocata. Questo evento funge da categoria per le allocazioni di memoria tra vari componenti InnoDB non associati a eventi di attesa più specifici, `memory/innodb/buf0buf` come menzionato in precedenza.

Dopo aver stabilito che i componenti InnoDB sono i principali consumatori di memoria, possiamo approfondire le specifiche utilizzando il seguente comando MySQL:

```
SHOW ENGINE INNODB STATUS \G;
```

Il comando SHOW ENGINE INNODB STATUS fornisce un rapporto completo sullo stato del motore di archiviazione InnoDB, incluse statistiche dettagliate sull'utilizzo della memoria per diversi componenti InnoDB. Può aiutare a identificare quali strutture o operazioni specifiche di InnoDB consumano più memoria. Per ulteriori informazioni, consulta le strutture in memoria di InnoDB nella documentazione MySQL.

Analizzando la `BUFFER POOL AND MEMORY` sezione del rapporto sullo stato di InnoDB, vediamo che 5.051.647.748 byte (4,7 GiB) sono allocati alla cache degli oggetti del dizionario, che rappresenta l'89% della memoria tracciata da. `memory/innodb/memory`

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5051647748
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

La cache degli oggetti del dizionario è una cache globale condivisa che archivia in memoria gli oggetti del dizionario di dati a cui si accedeva in precedenza per consentire il riutilizzo degli oggetti e migliorare le prestazioni. L'elevata allocazione di memoria alla cache degli oggetti del dizionario suggerisce un gran numero di oggetti di database nella cache del dizionario di dati.

Ora che sappiamo che la cache del dizionario dei dati è un consumatore primario, procediamo a ispezionare la cache del dizionario dei dati per individuare eventuali tabelle aperte. Per trovare il numero di tabelle nella cache delle definizioni delle tabelle, interroga la variabile di stato globale open_table_definition.

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 20000 |
+------------------------+-------+
1 row in set (0.00 sec)
```

Per ulteriori informazioni, consulta la pagina relativa ad apertura e chiusura delle tabelle in MySQL nella documentazione di MySQL.

È possibile limitare il numero di definizioni di tabella nella cache del dizionario dei dati limitando il `table_definition_cache` parametro nel cluster DB o nel gruppo di parametri dell'istanza DB. Per Aurora MySQL, questo valore funge da limite flessibile per il numero di tabelle nella cache delle definizioni delle tabelle. Il valore predefinito dipende dalla classe dell'istanza ed è impostato come segue:

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Quando il numero di tabelle supera il `table_definition_cache` limite, un meccanismo usato meno di recente (LRU) elimina e rimuove le tabelle dalla cache. Tuttavia, le tabelle coinvolte nelle relazioni con chiavi esterne non vengono inserite nell'elenco LRU, impedendone la rimozione.

Nel nostro scenario attuale, eseguiamo FLUSH TABLES per cancellare la cache delle definizioni delle tabelle. Questa azione comporta un calo significativo della variabile di stato globale OPEN_TABLE_DEFINITIONS, da 20.000 a 12, come mostrato di seguito:

```
mysql> show global status like 'open_table_definitions';
```

```
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 12    |
+------------------------+-------+
1 row in set (0.00 sec)
```

Nonostante questa riduzione, osserviamo che l'allocazione di memoria per `memory/innodb/memory` rimane elevata a 5,18 GiB e anche la memoria del dizionario allocata rimane invariata. Ciò è evidente dai seguenti risultati delle interrogazioni:

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.18 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5001599639
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

Questo utilizzo persistentemente elevato della memoria può essere attribuito alle tabelle coinvolte nelle relazioni con chiavi esterne. Queste tabelle non vengono inserite nell'elenco LRU per la rimozione, il che spiega perché l'allocazione di memoria rimane elevata anche dopo aver svuotato la cache delle definizioni delle tabelle.

Per risolvere il problema:

1. Rivedi e ottimizza lo schema del database, in particolare le relazioni con chiavi esterne.
2. Valuta la possibilità di passare a una classe di istanza DB più grande con più memoria per ospitare gli oggetti del dizionario.

Seguendo questi passaggi e comprendendo i modelli di allocazione della memoria, è possibile gestire meglio l'utilizzo della memoria nell'istanza DB Aurora MySQL e prevenire potenziali problemi di prestazioni dovuti alla pressione della memoria.

Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL

Il parametro a livello di istanza `aurora_oom_response` di Aurora MySQL può consentire all'istanza database di monitorare la memoria di sistema e stimare la memoria utilizzata da varie istruzioni e connessioni. Se la memoria del sistema sta esaurendo, può eseguire un elenco di azioni per tentare di liberare quella memoria. Lo fa nel tentativo di evitare il riavvio del database a causa di problemi out-of-memory (OOM). Il parametro a livello di istanza esegue una serie di azioni separate da virgole eseguite da un'istanza DB quando la memoria è scarsa. Il `aurora_oom_response` parametro è supportato per le versioni 2 e 3 di Aurora MySQL.

Per il parametro è possibile utilizzare i seguenti valori e combinazioni di essi. `aurora_oom_response` Una stringa vuota indica che non viene intrapresa alcuna azione e disattiva di fatto la funzionalità, lasciando il database soggetto al riavvio di OOM.

- `decline`— Rifiuta nuove interrogazioni quando l'istanza DB ha poca memoria.
- `kill_connect`— Chiude le connessioni al database che consumano una grande quantità di memoria e termina le transazioni correnti e le istruzioni DDL (Data Definition Language). Questa risposta non è supportata per Aurora MySQL versione 2.

  Per ulteriori informazioni, vedere l'istruzione KILL nella documentazione di MySQL.
- `kill_query`— Termina le query in ordine decrescente di consumo di memoria finché la memoria dell'istanza non supera la soglia bassa. Le istruzioni DDL non sono terminate.

  Per ulteriori informazioni, vedere l'istruzione KILL nella documentazione di MySQL.

- `print`— Stampa solo le query che consumano una grande quantità di memoria.
- `tune`: ottimizza le cache delle tabelle interne per restituire un po' di memoria al sistema. Aurora MySQL riduce la memoria utilizzata per le cache, ad esempio e in condizioni di memoria insufficiente. `table_open_cache` `table_definition_cache` Eventualmente, Aurora MySQL riporta l'utilizzo della memoria alla normalità quando il sistema non è più in condizioni di memoria insufficiente.

  Per ulteriori informazioni, consulta table_open_cache e table_definition_cache nella documentazione di MySQL.
- `tune_buffer_pool`— Riduce la dimensione del buffer pool per liberare parte della memoria e renderla disponibile al server del database per l'elaborazione delle connessioni. Questa risposta è supportata per Aurora MySQL versione 3.06 e successive.

  È necessario eseguire l'associazione `tune_buffer_pool` con uno dei due valori `kill_query` o `kill_connect` nel valore del parametro. `aurora_oom_response` In caso contrario, il ridimensionamento del pool di buffer non avverrà, anche se si include il valore del `tune_buffer_pool` parametro.

Nelle versioni di Aurora MySQL precedenti alla 3.06, per le classi di istanze DB con memoria inferiore o uguale a 4 GiB, quando l'istanza è sotto pressione di memoria, le azioni predefinite includono,, e. `print` `tune` `decline` `kill_query` Per le classi di istanze DB con memoria superiore a 4 GiB, il valore del parametro è vuoto per impostazione predefinita (disabilitato).

In Aurora MySQL versione 3.06 e successive, per le classi di istanze DB con memoria inferiore o uguale a 4 GiB, Aurora MySQL chiude anche le connessioni che consumano più memoria (). `kill_connect` Per le classi di istanze DB con memoria superiore a 4 GiB, il valore del parametro predefinito è. `print`

In Aurora MySQL versione 3.09 e successive, per le classi di istanze DB con memoria superiore a 4 GiB, il valore del parametro predefinito è. `print,decline,kill_connect`

Se si verificano spesso out-of-memory problemi, l'utilizzo della memoria può essere monitorato utilizzando tabelle di riepilogo della memoria quando questa opzione è abilitata. `performance_schema`

Per i CloudWatch parametri di Amazon relativi a OOM, consultaParametri a livello di istanza per Amazon Aurora. Per le variabili di stato globali relative a OOM, consultaAurora Le mie variabili di stato SQL globali.

Registrazione per i database Aurora MySQL

I log MySQL di Aurora forniscono informazioni essenziali sull'attività e sugli errori del database. Abilitando questi registri, è possibile identificare e risolvere i problemi, comprendere le prestazioni del database e controllare l'attività del database. Si consiglia di abilitare questi log per tutte le istanze DB Aurora MySQL per garantire prestazioni e disponibilità ottimali dei database. È possibile abilitare i seguenti tipi di registrazione. Ogni registro contiene informazioni specifiche che possono portare all'individuazione degli impatti sull'elaborazione del database.

- Errore: Aurora MySQL scrive nel registro degli errori solo all'avvio, all'arresto e quando rileva errori. Un'istanza database può andare avanti ore senza che ci siano nuove voci scritte nel file di log degli errori. Se non vedi voci recenti, significa che il server non ha riscontrato errori che generano una voce di registro. La registrazione degli errori è abilitata per impostazione predefinita. Per ulteriori informazioni, consulta Registri degli errori Aurora MySQL.
- Generale: il registro generale fornisce informazioni dettagliate sull'attività del database, incluse tutte le istruzioni SQL eseguite dal motore di database. Per ulteriori informazioni sull'attivazione della registrazione generale e sull'impostazione dei parametri di registrazioneRegistri generali e delle query lente di Aurora MySQL, vedere The general query log nella documentazione di MySQL.

  > **Note**
  >
  > I log generali possono crescere fino a diventare molto grandi e occupare lo spazio di archiviazione. Per ulteriori informazioni, consulta Rotazione e conservazione dei registri per Autora MySQL.

- Interrogazione lenta: il registro delle query lente è costituito da istruzioni SQL che impiegano più di long_query_time per essere eseguite e richiedono l'esame di almeno le righe min_examed_row_limit. È possibile utilizzare il log delle query lente per trovare le query che richiedono molto tempo di esecuzione e sono quindi idonee all'ottimizzazione.

  Il valore predefinito per `long_query_time` è di 10 secondi. Ti consigliamo di iniziare con un valore elevato per identificare le query più lente, quindi di procedere verso il basso per ottimizzarle.

  È inoltre possibile utilizzare parametri correlati, ad esempio e. `log_slow_admin_statements` `log_queries_not_using_indexes` Confronta `rows_examined` con`rows_returned`. Se `rows_examined` è molto maggiore di`rows_returned`, allora quelle query possono potenzialmente essere bloccanti.

In Aurora MySQL versione 3, è possibile abilitare l'ottenimento di maggiori dettagli. `log_slow_extra` Per ulteriori informazioni, consulta Slow Query Log Contents nella documentazione di MySQL. È inoltre possibile modificare `long_query_time` a livello di sessione per eseguire il debug in modo interattivo dell'esecuzione delle query, il che è particolarmente utile se abilitato a livello globale. `log_slow_extra`

Per ulteriori informazioni sull'attivazione della registrazione lenta delle query e sull'impostazione dei parametri di registrazione, vedereRegistri generali e delle query lente di Aurora MySQL, e The slow query log nella documentazione di MySQL.

- Audit: il registro di controllo monitora e registra l'attività del database. La registrazione di controllo per Aurora MySQL è denominata Advanced Auditing. Per abilitare Advanced Auditing, è necessario impostare determinati parametri del cluster DB. Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.
- Binario: il log binario (binlog) contiene eventi che descrivono le modifiche del database, come le operazioni di creazione delle tabelle e le modifiche ai dati della tabella. Contiene anche eventi relativi a istruzioni che potrebbero aver potuto apportare modifiche (ad esempio, un DELETE che non corrisponde a nessuna riga), a meno che non venga utilizzata la registrazione basata su righe. Il registro binario contiene anche informazioni sul tempo impiegato da ciascuna istruzione per l'aggiornamento dei dati.

  L'esecuzione di un server con la registrazione binaria abilitata rallenta leggermente le prestazioni. Tuttavia, i vantaggi del log binario, che consente di configurare la replica e le operazioni di ripristino, in genere superano questa lieve riduzione delle prestazioni.

  > **Note**
  >
  > Aurora MySQL non richiede la registrazione binaria per le operazioni di ripristino.

  Per ulteriori informazioni sull'attivazione della registrazione binaria e sull'impostazione del formato binlog, vedereConfigurazione di Aurora per database Single-AZ, e Il log binario nella documentazione di MySQL.

Puoi pubblicare i log di errore, generici, lenti, di interrogazione e di controllo su Amazon CloudWatch Logs. Per ulteriori informazioni, consulta Pubblicazione dei log del database su Amazon CloudWatch Logs.

Un altro strumento utile per riepilogare file di log lenti, generali e binari è. pt-query-digest

Risoluzione dei problemi di connessione per i database Aurora My SQL My SQL

Garantire una connettività affidabile tra le applicazioni e l'istanza RDS DB è fondamentale per il corretto funzionamento dei carichi di lavoro. Tuttavia, i problemi di connettività possono insorgere a causa di vari fattori, come configurazioni di rete, problemi di autenticazione o vincoli di risorse. Questa guida mira a fornire un approccio completo alla risoluzione dei problemi di connettività con Aurora My SQL My. SQL

Indice

-
-
-
  - Monitoraggio aggiuntivo per Aurora My SQL
-
-
-
  - Esempio 1: risoluzione dei tentativi di connessione non riusciti
  - Esempio 2: Risoluzione dei problemi di disconnessione anomala del client

L'identificazione della categoria specifica del problema di connettività può aiutare a restringere le potenziali cause e guidare il processo di risoluzione dei problemi. Ogni categoria potrebbe richiedere approcci e tecniche diversi per la diagnosi e la risoluzione. I problemi di connettività dei database possono essere generalmente classificati nelle seguenti categorie.

Errori ed eccezioni di connessione

> Gli errori e le eccezioni di connessione possono verificarsi per vari motivi, ad esempio stringhe di connessione errate, errori di autenticazione, interruzioni della rete o problemi del server di database. Le cause possono includere parametri di connessione non configurati correttamente, credenziali non valide, interruzioni della rete o arresti anomali o riavvii del server di database. Anche i gruppi di sicurezza configurati in modo errato, le impostazioni del cloud privato virtuale (VPC), le liste di controllo degli accessi alla rete (ACLs) e le tabelle di routing associate alle sottoreti possono causare problemi di connessione.

Limite di connessione raggiunto

Questo problema si verifica quando il numero di connessioni simultanee al server del database supera il limite massimo consentito. I server di database hanno in genere un limite massimo di connessioni configurabile definito dal parametro max_connections nei cluster e nei gruppi di parametri di istanza. Imponendo un limite di connessione, il server di database garantisce di disporre di risorse sufficienti (ad esempio memoria e handle di file) per gestire le connessioni esistenti in modo efficiente e fornire prestazioni accettabili. CPU Le cause possono includere perdite di connessione nell'applicazione, un pool di connessioni inefficiente o un aumento imprevisto delle richieste di connessione.

Timeout di connessione

I timeout di connessione si verificano quando l'applicazione client non è in grado di stabilire una connessione con il server del database entro un periodo di timeout specificato. Le cause più comuni includono problemi di rete, sovraccarico del server, regole del firewall e impostazioni di connessione non configurate correttamente.

Timeout di connessione inattivi

Le connessioni inattive che rimangono inattive per un periodo prolungato potrebbero essere chiuse automaticamente dal server del database per risparmiare risorse. Questo timeout è in genere configurabile utilizzando `wait_timeout` and e `interactive_timeout parameters` deve essere regolato in base ai modelli di utilizzo della connessione dell'applicazione. Le cause possono includere la logica dell'applicazione che lascia le connessioni inattive per periodi prolungati o una gestione impropria delle connessioni.

Disconnessione intermittente delle connessioni esistenti

Questa classe di errori si riferisce a uno scenario in cui le connessioni stabilite tra un'applicazione client e il database vengono interrotte o disconnesse in modo imprevisto a intervalli irregolari, nonostante siano attive e in uso. Queste disconnessioni si verificano in modo intermittente, nel senso che si verificano a intervalli irregolari e non in modo coerente. Le cause possono includere le seguenti:

- Problemi relativi al server di database, ad esempio riavvii o failover
- Gestione impropria della connessione all'applicazione
- Problemi relativi al bilanciamento del carico e al proxy
- Instabilità della rete
- Problemi con componenti o middleware di terze parti coinvolti nel percorso di connessione

- Timeout di esecuzione delle query
- Limiti di risorse sul lato server o client

L'identificazione della causa principale attraverso il monitoraggio, la registrazione e l'analisi completi è fondamentale, mentre l'implementazione di meccanismi adeguati di gestione degli errori, pool di connessioni e tentativi di ripetizione può aiutare a mitigare l'impatto di queste disconnessioni intermittenti sulla funzionalità e sull'esperienza utente dell'applicazione.

La raccolta di dati completi relativi ai componenti dell'applicazione, del database, della rete e dell'infrastruttura è fondamentale per risolvere efficacemente i problemi di connettività tra un'applicazione e un database Aurora My SQL RDS . Raccogliendo registri, configurazioni e informazioni diagnostiche pertinenti, si ottengono informazioni preziose che possono aiutare a identificare la causa principale dei problemi di connettività e guidarvi verso una risoluzione appropriata.

I log e le configurazioni di rete, come le regole dei gruppi di sicurezza, VPC le impostazioni e le tabelle di routing, sono essenziali per identificare potenziali colli di bottiglia o configurazioni errate relative alla rete che potrebbero impedire all'applicazione di stabilire una connessione corretta con il database. Analizzando questi componenti di rete, è possibile assicurarsi che le porte necessarie siano aperte, che gli indirizzi IP siano consentiti e che le configurazioni di routing siano impostate correttamente.

Timestamp

Registra i timestamp esatti in cui si verificano i problemi di connettività. Questo può aiutare a identificare i modelli o a correlare i problemi con altri eventi o attività.

Registri del motore DB

Oltre ai registri generali del database, esamina i log del motore di database (ad esempio, My SQL error log e slow query log) per eventuali informazioni o errori pertinenti che potrebbero essere correlati ai problemi di connettività intermittente. Per ulteriori informazioni, consulta Registrazione per i database Aurora MySQL.

Registri delle applicazioni client

Raccogli log dettagliati dalle applicazioni client che si connettono al database. I log delle applicazioni forniscono visibilità sui tentativi di connessione, sugli errori e su qualsiasi informazione rilevante dal punto di vista dell'applicazione, il che può rivelare problemi relativi alle

stringhe di connessione, alle credenziali di autenticazione o alla gestione della connessione a livello di applicazione.

I log del database, d'altra parte, offrono informazioni sugli errori sul database, sulle query lente o sugli eventi che potrebbero contribuire ai problemi di connettività. Per ulteriori informazioni, consulta Registrazione per i database Aurora MySQL.

Variabili di ambiente del client

Controlla se eventuali variabili di ambiente o impostazioni di configurazione sul lato client potrebbero influire sulla connessione al database, come le impostazioni proxy,SSL/TLSsettings o altre variabili pertinenti.

Versioni della libreria client

Assicurati che il client utilizzi le versioni più recenti di tutti i driver, le librerie o i framework di database utilizzati per la connettività del database. Le versioni obsolete possono presentare problemi noti o problemi di compatibilità.

Acquisizione dalla rete del client

Esegui un'acquisizione di rete sul lato client utilizzando uno strumento come Wireshark o `tcpdump` durante i periodi in cui si verificano problemi di connettività. Questo può aiutare a identificare eventuali problemi o anomalie relativi alla rete sul lato client.

Topologia della rete client

Comprendi la topologia di rete del client, inclusi eventuali firewall, sistemi di bilanciamento del carico o altri componenti come RDS Proxy o Proxy SQL che effettuano connessioni al database anziché essere il client a effettuare connessioni direttamente.

Impostazioni del sistema operativo del client

Controlla le impostazioni del sistema operativo del client che potrebbero influire sulla connettività di rete, come le regole del firewall, le impostazioni degli adattatori di rete e qualsiasi altra impostazione pertinente.

Configurazione del pool di connessioni

Se utilizzi un meccanismo di pool di connessioni nella tua applicazione, rivedi le impostazioni di configurazione e monitora le metriche del pool (ad esempio, connessioni attive, connessioni inattive e timeout di connessione) per assicurarti che il pool funzioni correttamente. Controlla anche le impostazioni del pool, come la dimensione massima del pool, la dimensione minima

del pool e le impostazioni di convalida della connessione, per assicurarti che siano configurate correttamente.

Stringa di connessione

La stringa di connessione include in genere parametri come il nome host o l'endpoint, il numero di porta, il nome del database e le credenziali di autenticazione. L'analisi della stringa di connessione può aiutare a identificare potenziali configurazioni errate o impostazioni errate che potrebbero causare problemi di connettività. Ad esempio, un nome host o un numero di porta errati possono impedire al client di raggiungere l'istanza del database, mentre credenziali di autenticazione non valide possono causare errori di autenticazione e il rifiuto della connessione. Inoltre, la stringa di connessione può rivelare problemi relativi al pool di connessioni, ai timeout o ad altre impostazioni specifiche della connessione che potrebbero contribuire a problemi di connettività. Fornire la stringa di connessione completa utilizzata dall'applicazione client può aiutare a individuare eventuali errori di configurazione sul client.

Parametri del database

Monitora le metriche del database come CPU l'utilizzo, l'utilizzo della memoria e l'I/O del disco durante i periodi in cui si verificano problemi di connettività. Questi possono aiutare a identificare se l'istanza DB presenta un conflitto di risorse o problemi di prestazioni.

DB engine version (Versione motore del database

Nota la versione del motore Aurora My SQL RDS DB. AWS rilascia regolarmente aggiornamenti che risolvono problemi noti, vulnerabilità di sicurezza e introducono miglioramenti delle prestazioni. Pertanto, consigliamo vivamente di eseguire l'aggiornamento alle ultime versioni disponibili, poiché questi aggiornamenti spesso includono correzioni di bug e miglioramenti specificamente correlati a connettività, prestazioni e stabilità. Fornire le informazioni sulla versione del database, insieme agli altri dettagli raccolti, può aiutare Supporto a diagnosticare e risolvere efficacemente i problemi di connettività.

Metriche di rete

Raccogli parametri di rete come latenza, perdita di pacchetti e velocità effettiva durante i periodi in cui si verificano problemi di connettività. Strumenti come `ping` e strumenti di monitoraggio della rete possono aiutare a raccogliere questi dati. `traceroute`

Informazioni sulla fonte e sul cliente

Determina gli indirizzi IP degli application server, dei load balancer o di qualsiasi altro componente che avvia le connessioni al database. Potrebbe trattarsi di un singolo indirizzo IP o di un intervallo

di indirizzi IP (CIDRnotazione). Se l'origine è un'EC2istanza Amazon, è utile anche esaminare il tipo di istanza, la zona di disponibilità, l'ID di sottorete e i gruppi di sicurezza associati all'istanza e i dettagli dell'interfaccia di rete come l'indirizzo IP privato e l'indirizzo IP pubblico.

Analizzando a fondo i dati raccolti, è possibile identificare configurazioni errate, vincoli di risorse, interruzioni della rete o altri problemi di base che causano problemi di connettività intermittenti o persistenti. Queste informazioni consentono di intraprendere azioni mirate, come la regolazione delle configurazioni, la risoluzione di problemi di rete o la gestione delle connessioni a livello di applicazione.

Per monitorare e risolvere i problemi di connettività, puoi utilizzare le seguenti metriche e funzionalità.

CloudWatch metriche

- `CPUUtilization`— Un CPU utilizzo elevato dell'istanza DB può rallentare l'esecuzione delle query, con conseguenti timeout o rifiuti di connessione.
- `DatabaseConnections`— Monitora il numero di connessioni attive all'istanza DB. Un numero elevato di connessioni vicino al massimo configurato può indicare potenziali problemi di connettività o l'esaurimento del pool di connessioni.
- `FreeableMemory`— La scarsa disponibilità di memoria può causare problemi di prestazioni e problemi di connettività a causa di limiti di risorse.
- `NetworkReceiveThroughpute NetworkTransmitThroughput` — Picchi o cali insoliti del throughput di rete possono indicare problemi di connettività o strozzature della rete.

Parametri Performance Insights

Per risolvere i problemi di connettività in Aurora My SQL RDS for My Insights, analizza le metriche del database come le seguenti:

- Aborted_clients
- Aborted_connects
- Connessioni
- max_connections
- Threads_connected
- Threads_created
- Threads_running

Queste metriche possono aiutarti a identificare i punti critici di connessione, rilevare problemi di rete o di autenticazione, ottimizzare il pool di connessioni e garantire una gestione efficiente dei thread. Per ulteriori informazioni, consulta Contatori di Performance Insights per Aurora MySQL.

Funzionalità di Performance Insights

- Caricamento del database: visualizza il carico del database nel tempo e mettilo in correlazione con problemi di connettività o peggioramento delle prestazioni.
- SQLStatistiche: analizza SQL le statistiche per identificare query o operazioni di database inefficienti che potrebbero contribuire a problemi di connettività.
- Interrogazioni principali: identifica e analizza le query che richiedono più risorse, in modo da identificare potenziali problemi di prestazioni o query di lunga durata che potrebbero causare problemi di connettività.

Monitorando queste metriche e sfruttando Performance Insights, è possibile ottenere visibilità sulle prestazioni dell'istanza di database, sull'utilizzo delle risorse e sui potenziali colli di bottiglia che potrebbero causare problemi di connettività. Per esempio:

- Un valore `DatabaseConnections` vicino al limite massimo può indicare l'esaurimento del pool di connessioni o una gestione impropria della connessione, con conseguenti problemi di connettività.
- Un valore elevato `CPUUtilization` o basso `FreeableMemory` può indicare vincoli di risorse, che possono causare un'esecuzione lenta delle query e timeout o rifiuti di connessione.
- L'analisi delle query principali e delle SQLstatistiche può aiutare a identificare le query inefficienti o che richiedono molte risorse e che potrebbero contribuire a problemi di connettività.

Inoltre, il monitoraggio dei CloudWatch registri e l'impostazione degli allarmi possono aiutarvi a identificare e rispondere in modo proattivo ai problemi di connettività prima che si aggravino.

È importante notare che, sebbene queste metriche e questi strumenti possano fornire informazioni preziose, devono essere utilizzati insieme ad altre procedure di risoluzione dei problemi.

Monitoraggio aggiuntivo per Aurora My SQL

CloudWatch metriche

- `AbortedClients`— Tiene traccia del numero di connessioni client che non sono state chiuse correttamente.

- `AuroraSlowConnectionHandleCount`— Tiene traccia del numero di operazioni lente di gestione della connessione, indicando potenziali problemi di connettività o rallentamenti delle prestazioni.
- `AuroraSlowHandshakeCount`— Misura il numero di operazioni di handshake lente, che può anche essere un indicatore di problemi di connettività.
- `ConnectionAttempts`— Misura il numero di tentativi di connessione effettuati all'istanza Aurora My SQL DB.

Variabili di stato globali

`Aurora_external_connection_count`— Mostra il numero di connessioni al database all'istanza DB, escluse le connessioni di RDS servizio utilizzate per i controlli dello stato del database.

Monitorando questi parametri e le variabili di stato globali, puoi ottenere visibilità sui modelli di connessione, sugli errori e sui potenziali colli di bottiglia che potrebbero causare problemi di connettività con la tua istanza Amazon Aurora My. SQL

Ad esempio, un numero elevato di `AbortedClients` o `AuroraSlowConnectionHandleCount` può indicare problemi di connettività.

Inoltre, l'impostazione di CloudWatch allarmi e notifiche può aiutarvi a identificare e rispondere in modo proattivo ai problemi di connettività prima che si aggravino e influiscano sulle prestazioni dell'applicazione.

Di seguito sono riportati alcuni errori di connettività comuni per i SQL database Aurora My SQL RDS , insieme ai relativi codici di errore e spiegazioni.

Codice di errore 1040: troppe connessioni

Questo errore si verifica quando il client tenta di stabilire un numero di connessioni superiore al massimo consentito dal server del database. Le cause possibili sono:

- Configurazione errata del pool di connessioni: se utilizzi un meccanismo di pool di connessioni, assicurati che la dimensione massima del pool non sia troppo alta e che le connessioni vengano rilasciate correttamente al pool.
- Configurazione dell'istanza di database: verifica l'impostazione del numero massimo di connessioni consentite per l'istanza di database e modificala, se necessario, impostando il parametro. `max_connections`

- Concorrenza elevata: se più client o applicazioni si connettono contemporaneamente al database, è possibile raggiungere il limite massimo di connessioni consentito.

Codice errore 1045: accesso negato per l'utente '...' @ '...' (utilizzando la password: YES /NO)

Questo errore indica un errore di autenticazione durante il tentativo di connessione al database. Le cause possibili sono:

- Compatibilità del plug-in di autenticazione: verifica se il plug-in di autenticazione utilizzato dal client è compatibile con il meccanismo di autenticazione del server di database.
- Nome utente o password errati: verifica che nella stringa di connessione o nel meccanismo di autenticazione vengano utilizzati il nome utente e la password corretti.
- Autorizzazioni utente: assicurati che l'utente disponga delle autorizzazioni necessarie per connettersi all'istanza del database dall'host o dalla rete specificati.

Codice errore 1049: database sconosciuto '...'

Questo errore indica che il client sta tentando di connettersi a un database che non esiste sul server. Le cause possibili sono:

- Database non creato: assicurati che il database specificato sia stato creato sul server del database.
- Nome del database errato: ricontrolla il nome del database utilizzato nella stringa di connessione o nella query per verificarne la precisione.
- Autorizzazioni utente: verifica che l'utente disponga delle autorizzazioni necessarie per accedere al database specificato.

Codice errore 1153: pacchetto ricevuto più grande dei byte «max_allowed_packet»

Questo errore si verifica quando il client tenta di inviare o ricevere dati che superano la dimensione massima del pacchetto consentita dal server del database. Le cause possibili sono:

- Interrogazioni o set di risultati di grandi dimensioni: se si eseguono query che coinvolgono grandi quantità di dati, è possibile che il limite di dimensione dei pacchetti venga superato.
- Impostazioni delle dimensioni dei pacchetti non configurate correttamente: controlla l'`max_allowed_packet`impostazione sul server del database e modificala se necessario.
- Problemi di configurazione della rete: assicurati che la configurazione di rete (ad esempio, la MTU dimensione) consenta le dimensioni dei pacchetti richieste.

Codice errore 1226: l'utente '...' ha superato la risorsa 'max_user_connections' (valore corrente:...)

Questo errore indica che l'utente ha superato il numero massimo di connessioni simultanee consentite dal server del database. Le possibili cause includono le seguenti:

- Configurazione errata del pool di connessioni: se si utilizza un meccanismo di pool di connessioni, assicuratevi che la dimensione massima del pool non sia troppo alta per il limite di connessione dell'utente.
- Configurazione dell'istanza del database: verifica l'`max_user_connections`impostazione per l'istanza del database e modificala se necessario.
- Concorrenza elevata: se più client o applicazioni si connettono contemporaneamente al database utilizzando lo stesso utente, è possibile raggiungere il limite di connessione specifico dell'utente.

Codice di errore 2003: impossibile connettersi al mio SQL server su '...' (10061)

Questo errore si verifica in genere quando il client non è in grado di stabilire una connessione TCP /IP con il server del database. Può essere causato da vari problemi, come i seguenti:

- Stato dell'istanza del database: assicurati che l'istanza del database sia nello `available` stato in cui si trova e che non sia sottoposta a operazioni di manutenzione o backup.
- Regole del firewall: verifica se eventuali firewall (sistema operativo, rete o gruppo di sicurezza) bloccano la connessione sulla porta specificata (in genere 3306 per My). SQL
- Nome host o endpoint errati: assicurati che il nome host o l'endpoint utilizzato nella stringa di connessione sia corretto e corrisponda all'istanza del database.
- Problemi di connettività di rete: verifica che il computer client possa raggiungere l'istanza del database attraverso la rete. Verifica eventuali interruzioni di rete, problemi di routing VPC o configurazioni errate della sottorete.

Codice di errore 2005: sconosciuto Il mio SQL server host '...' (11001)

Questo errore si verifica quando il client non è in grado di risolvere il nome host o l'endpoint del server del database in un indirizzo IP. Le cause possibili sono:

- DNSproblemi di risoluzione: verifica che il computer client sia in grado di risolvere correttamente il nome host utilizzando. DNS Controlla le DNS impostazioni, DNS la cache e prova a utilizzare l'indirizzo IP anziché il nome host.
- Nome host o endpoint errati: ricontrolla il nome host o l'endpoint utilizzato nella stringa di connessione per verificarne la precisione.
- Problemi di configurazione della rete: assicurati che la configurazione di rete del client (ad esempioVPC, le tabelle di sottorete e route) consenta la DNS risoluzione e la connettività all'istanza del database.

Codice di errore 2026: SSL errore di connessione

Questo errore si verifica quando si verifica un problema con la TLS configurazioneSSL/o la convalida del certificato durante il tentativo di connessione. Le cause possibili sono:

- Scadenza del certificato: verifica se il TLS certificatoSSL/utilizzato dal server è scaduto e deve essere rinnovato.
- Problemi di convalida del certificato: verifica che il client sia in grado di convalidare correttamente il TLS certificatoSSL/del server e che il certificato sia affidabile.
- Problemi di configurazione della rete: assicurati che la configurazione di rete consenta il processo di SSL/TLS connections and doesn't block or interfere with the SSL/TLS handshake.
- SSL/TLS configuration mismatch – Make sure that the SSL/TLSle impostazioni (ad esempio, suite di crittografia e versioni di protocollo) sul client e sul server sono compatibili.

Numero massimo connessioni

La regolazione di questi parametri può aiutare a prevenire i problemi di connessione causati dal raggiungimento del limite massimo consentito di connessioni. Assicurati che questi valori siano impostati in modo appropriato in base ai requisiti di concorrenza e ai vincoli di risorse dell'applicazione.

- `max_connections`— Questo parametro specifica il numero massimo di connessioni simultanee consentite all'istanza DB.
- `max_user_connections`— Questo parametro può essere specificato durante la creazione e la modifica dell'utente e imposta il numero massimo di connessioni simultanee consentite per un account utente specifico.

Dimensione del buffer di rete

L'aumento di questi valori può migliorare le prestazioni della rete, in particolare per i carichi di lavoro che comportano trasferimenti di dati o set di risultati di grandi dimensioni. Tuttavia, fai attenzione poiché buffer di dimensioni maggiori possono consumare più memoria.

- `net_buffer_length`— Questo parametro imposta la dimensione iniziale per la connessione del client e i buffer dei risultati, bilanciando l'utilizzo della memoria con le prestazioni delle query.
- `max_allowed_packet`— Questo parametro specifica la dimensione massima di un singolo pacchetto di rete che può essere inviato o ricevuto dall'istanza DB.

Compressione di rete (lato client)

L'attivazione della compressione di rete può ridurre l'utilizzo della larghezza di banda della rete, ma può aumentare il CPU sovraccarico sia sul lato client che sul lato server.

- `compress`— Questo parametro abilita o disabilita la compressione di rete per le comunicazioni client/server.
- `compress_protocol`— Questo parametro specifica il protocollo di compressione da utilizzare per le comunicazioni di rete.

Ottimizzazione delle prestazioni di rete

La regolazione di questi timeout può aiutare a gestire le connessioni inattive e prevenire l'esaurimento delle risorse, ma fai attenzione perché valori bassi possono causare interruzioni premature della connessione.

- `interactive_timeout`— Questo parametro specifica il numero di secondi in cui il server attende l'attività su una connessione interattiva prima di chiuderla.
- `wait_timeout`— Questo parametro determina il numero di secondi in cui il server attende l'attività su una connessione non interattiva prima di chiuderla.

Impostazioni di timeout di rete

La regolazione di questi timeout può aiutare a risolvere i problemi relativi alle connessioni lente o che non rispondono. Attenzione però a non impostarli su valori troppo bassi, in quanto ciò può causare interruzioni premature della connessione.

- `net_read_timeout`— Questo parametro specifica il numero di secondi di attesa di ulteriori dati da una connessione prima di terminare l'operazione di lettura.
- `net_write_timeout`— Questo parametro determina il numero di secondi di attesa per la scrittura di un blocco su una connessione prima di terminare l'operazione di scrittura.

Esempio 1: risoluzione dei tentativi di connessione non riusciti

I tentativi di connessione possono fallire per diversi motivi, tra cui errori di autenticazioneSSL, errori di TLS handshake, `max_connections` limite raggiunto e vincoli di risorse sull'istanza DB.

È possibile tenere traccia del numero di connessioni non riuscite da Performance Insights o utilizzando il comando seguente.

```
mysql> show global status like 'aborted_connects';
```

```
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Aborted_connects | 7     |
+------------------+-------+
1 row in set (0.00 sec)
```

Se il numero `Aborted_connects` aumenta nel tempo, è possibile che l'applicazione presenti problemi di connettività intermittenti.

È possibile utilizzare Aurora Advanced Auditing per registrare le connessioni e le disconnessioni dalle connessioni client. È possibile farlo impostando i seguenti parametri nel gruppo di parametri del cluster DB:

- `server_audit_logging` = `1`
- `server_audit_events` = `CONNECT`

Di seguito è riportato un estratto dei registri di controllo relativi a un accesso non riuscito.

```
1728498527380921,auora-mysql-node1,user_1,172.31.49.222,147189,0,FAILED_CONNECT,,,1045
1728498527380940,auora-mysql-node1,user_1,172.31.49.222,147189,0,DISCONNECT,,,0
```

Dove:

- `1728498527380921`— Il timestamp dell'epoca in cui si è verificato l'accesso non riuscito
- `aurora-mysql-node1`— L'identificatore di istanza del nodo del cluster Aurora SQL My su cui la connessione non è riuscita
- `user_1`— Il nome dell'utente del database per il quale l'accesso non è riuscito
- `172.31.49.222`— L'indirizzo IP privato del client da cui è stata stabilita la connessione
- `147189`— L'ID di connessione dell'accesso non riuscito
- `FAILED_CONNECT`— Indica che la connessione non è riuscita.
- `1045`— Il codice di ritorno. Un valore diverso da zero indica un errore. In questo caso, `1045` corrisponde all'accesso negato.

Per ulteriori informazioni, consulta Codici di errore del server e Codici di errore del client nella SQL documentazione personale.

È inoltre possibile esaminare i log degli SQL errori di Aurora My per eventuali messaggi di errore correlati, ad esempio:

```
2024-10-09T19:26:59.310443Z 220 [Note] [MY-010926] [Server] Access denied for user
 'user_1'@'172.31.49.222' (using password: YES) (sql_authentication.cc:1502)
```

Esempio 2: Risoluzione dei problemi di disconnessione anomala del client

È possibile tenere traccia del numero di disconnessioni anomale dei client da Performance Insights o utilizzando il comando seguente.

```
mysql> show global status like 'aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 9     |
+-----------------+-------+
1 row in set (0.01 sec)
```

Se il numero di connessioni `Aborted_clients` aumenta nel tempo, significa che l'applicazione non chiude correttamente le connessioni al database. Se le connessioni non vengono chiuse correttamente, possono verificarsi perdite di risorse e potenziali problemi di prestazioni. Lasciare le connessioni aperte inutilmente può consumare risorse di sistema, come memoria e descrittori di file, con il rischio che l'applicazione o il server non rispondano o si riavviino.

È possibile utilizzare la seguente query per identificare gli account che non chiudono correttamente le connessioni. Recupera il nome dell'account utente, l'host da cui l'utente si connette, il numero di connessioni non chiuse e la percentuale di connessioni non chiuse.

```
SELECT
    ess.user,
    ess.host,
    (a.total_connections - a.current_connections) - ess.count_star AS not_closed,
    (((a.total_connections - a.current_connections) - ess.count_star) * 100) /
 (a.total_connections - a.current_connections) AS pct_not_closed
FROM
    performance_schema.events_statements_summary_by_account_by_event_name AS ess
    JOIN performance_schema.accounts AS a ON (ess.user = a.user AND ess.host = a.host)
WHERE
    ess.event_name = 'statement/com/quit'
    AND (a.total_connections - a.current_connections) > ess.count_star;
```

```
+----------+---------------+------------+----------------+
| user     | host          | not_closed | pct_not_closed |
+----------+---------------+------------+----------------+
| user1    | 172.31.49.222 |          1 |        33.3333 |
| user1    | 172.31.93.250 |       1024 |        12.1021 |
| user2    | 172.31.93.250 |         10 |        12.8551 |
+----------+---------------+------------+----------------+
3 rows in set (0.00 sec)
```

Dopo aver identificato gli account utente e gli host dai quali le connessioni non sono chiuse, puoi procedere a controllare il codice che non chiude correttamente le connessioni.

Ad esempio, con il SQL connettore My in Python, utilizzate il `close()` metodo dell'oggetto di connessione per chiudere le connessioni. Ecco un esempio di funzione che stabilisce una connessione a un database, esegue una query e chiude la connessione:

```
import mysql.connector

def execute_query(query):
    # Establish a connection to the database
    connection = mysql.connector.connect(
        host="your_host",
        user="your_username",
        password="your_password",
        database="your_database"
    )

    try:
        # Create a cursor object
        cursor = connection.cursor()

        # Execute the query
        cursor.execute(query)

        # Fetch and process the results
        results = cursor.fetchall()
        for row in results:
            print(row)

    finally:
        # Close the cursor and connection
        cursor.close()
```

```
        connection.close()
```

In questo esempio, il `connection.close()` metodo viene chiamato nel `finally` blocco per assicurarsi che la connessione sia chiusa, indipendentemente dal fatto che si verifichi un'eccezione.

Risoluzione dei problemi delle prestazioni delle query per i database Aurora MySQL

MySQL fornisce il controllo dell'ottimizzatore di query tramite variabili di sistema che influiscono sul modo in cui vengono valutati i piani di query, ottimizzazioni commutabili, suggerimenti sull'ottimizzatore e sull'indice e il modello di costo dell'ottimizzatore. Questi punti dati possono essere utili non solo per confrontare diversi ambienti MySQL, ma anche per confrontare i piani di esecuzione delle query precedenti con i piani di esecuzione attuali e per comprendere l'esecuzione complessiva di una query MySQL in qualsiasi momento.

Le prestazioni delle query dipendono da molti fattori, tra cui il piano di esecuzione, lo schema e le dimensioni della tabella, le statistiche, le risorse, gli indici e la configurazione dei parametri. L'ottimizzazione delle query richiede l'identificazione dei punti critici e l'ottimizzazione del percorso di esecuzione.

- Trova il piano di esecuzione per la query e verifica se la query utilizza gli indici appropriati. È possibile ottimizzare la query utilizzando `EXPLAIN` e rivedendo i dettagli di ciascun piano.
- Aurora MySQL versione 3 (compatibile con MySQL 8.0 Community Edition) utilizza un'istruzione. `EXPLAIN ANALYZE` L'`EXPLAIN ANALYZE`istruzione è uno strumento di profilazione che mostra dove MySQL dedica il tempo alla tua query e perché. Con`EXPLAIN ANALYZE`, Aurora MySQL pianifica, prepara ed esegue la query contando le righe e misurando il tempo impiegato in vari punti del piano di esecuzione. Al termine della query, `EXPLAIN ANALYZE` stampa il piano e le relative misurazioni anziché il risultato dell'interrogazione.
- Mantieni aggiornate le statistiche dello schema utilizzando l'`ANALYZE`istruzione. L'ottimizzatore di query a volte può scegliere piani di esecuzione scadenti a causa di statistiche obsolete. Ciò può comportare una riduzione delle prestazioni di una query a causa di stime imprecise della cardinalità sia delle tabelle che degli indici. La `last_update` colonna della tabella innodb_table_stats mostra l'ultima volta che le statistiche dello schema sono state aggiornate, il che è un buon indicatore di «stallo».
- Possono verificarsi altri problemi, come la distorsione della distribuzione dei dati, che non vengono presi in considerazione per la cardinalità della tabella. Per ulteriori informazioni, vedere Stima della complessità di ANALYZE TABLE per le tabelle InnoDB e le statistiche dell'istogramma in MySQL nella documentazione MySQL.

Comprendere il tempo impiegato dalle interrogazioni

Di seguito sono riportati i modi per determinare il tempo impiegato dalle query:

- Profilazione
- Schema delle prestazioni
- Ottimizzatore di query

Profiling

Per impostazione predefinita, la profilazione è disabilitata. Abilita la profilazione, quindi esegui la query lenta e rivedi il profilo.

```
SET profiling = 1;
Run your query.
SHOW PROFILE;
```

1. Identifica la fase in cui viene impiegato più tempo. In base agli stati generali dei thread nella documentazione di MySQL, la lettura e l'elaborazione delle righe per `SELECT` un'istruzione è spesso lo stato di esecuzione più lungo durante la durata di una determinata query. È possibile utilizzare l'`EXPLAIN`istruzione per capire come MySQL esegue questa query.
2. Esamina il log delle query lente `rows_sent` per valutare `rows_examined` e assicurarti che il carico di lavoro sia simile in ogni ambiente. Per ulteriori informazioni, consulta Registrazione per i database Aurora MySQL.
3. Esegui il comando seguente per le tabelle che fanno parte della query identificata:

   ```
   SHOW TABLE STATUS\G;
   ```

4. Acquisisci i seguenti output prima e dopo l'esecuzione della query in ogni ambiente:

   ```
   SHOW GLOBAL STATUS;
   ```

5. Esegui i seguenti comandi in ogni ambiente per vedere se ci sono altre query/sessioni che influenzano le prestazioni di questa query di esempio.

   ```
   SHOW FULL PROCESSLIST;

   SHOW ENGINE INNODB STATUS\G;
   ```

A volte, quando le risorse del server sono occupate, ciò influisce su tutte le altre operazioni sul server, comprese le query. È inoltre possibile acquisire informazioni periodicamente quando vengono eseguite delle query o impostare un `cron` processo per acquisire informazioni a intervalli utili.

Performance Schema

Lo schema delle prestazioni fornisce informazioni utili sulle prestazioni di runtime del server, con un impatto minimo su tali prestazioni. Questo è diverso `dainformation_schema`, che fornisce informazioni sullo schema sull'istanza DB. Per ulteriori informazioni, consulta Panoramica dello schema delle prestazioni per Performance Insights su Aurora MySQL, Amazon RDS MariaDB o MySQL.

Traccia dell'ottimizzatore di query

Per capire perché è stato scelto un particolare piano di query per l'esecuzione, puoi configurare l'accesso `optimizer_trace` all'ottimizzatore di query MySQL.

Esegui una traccia dell'ottimizzatore per mostrare informazioni complete su tutti i percorsi disponibili per l'ottimizzatore e sulla sua scelta.

```
SET SESSION OPTIMIZER_TRACE="enabled=on";
SET optimizer_trace_offset=-5, optimizer_trace_limit=5;

-- Run your query.
SELECT * FROM table WHERE x = 1 AND y = 'A';

-- After the query completes:
SELECT * FROM information_schema.OPTIMIZER_TRACE;
SET SESSION OPTIMIZER_TRACE="enabled=off";
```

Revisione delle impostazioni dell'ottimizzatore delle query

Aurora MySQL versione 3 (compatibile con MySQL 8.0 Community Edition) presenta molte modifiche relative all'ottimizzatore rispetto alla versione 2 di Aurora MySQL (compatibile con MySQL 5.7 Community Edition). Se disponi di valori personalizzati per il, ti consigliamo di esaminare le differenze tra le impostazioni predefinite `optimizer_switch` e di impostare i valori più adatti al tuo carico di lavoro. `optimizer_switch` Si consiglia inoltre di testare le opzioni disponibili per Aurora MySQL versione 3 per esaminare le prestazioni delle query.

> **Note**
>
> Aurora MySQL versione 3 utilizza il valore predefinito della community di 20 per il parametro innodb_stats_persistent_sample_pages.

È possibile utilizzare il `optimizer_switch` seguente comando per mostrare i valori:

```
SELECT @@optimizer_switch\G;
```

La tabella seguente mostra i `optimizer_switch` valori predefiniti per le versioni 2 e 3 di Aurora MySQL.

Impostazione	Aurora MySQL versione 2	Aurora MySQL versione 3
`batched_key_access`	off	off
`block_nested_loop`	on	on
`condition_fanout_filter`	on	on
`derived_condition_pushdown`	–	on
`derived_merge`	on	on
`duplicateweedout`	on	on
`engine_condition_pushdown`	on	on
`firstmatch`	on	on
`hash_join`	off	on
`hash_join_cost_based`	on	–
`hypergraph_optimizer`	–	off
`index_condition_pushdown`	on	on
`index_merge`	on	on

Impostazione	Aurora MySQL versione 2	Aurora MySQL versione 3
index_merge_intersection	on	on
index_merge_sort_union	on	on
index_merge_union	on	on
loosescan	on	on
materialization	on	on
mrr	on	on
mrr_cost_based	on	on
prefer_ordering_index	on	on
semijoin	on	on
skip_scan	–	on
subquery_materialization_cost_based	on	on
subquery_to_derived	–	off
use_index_extensions	on	on
use_invisible_indexes	–	off

Per ulteriori informazioni, consulta Ottimizzazioni commutabili (MySQL 5.7) e Ottimizzazioni commutabili (MySQL 8.0) nella documentazione di MySQL.

Amazon Aurora La mia referenza SQL

Questo riferimento include informazioni sui SQL parametri di Aurora My, sulle variabili di stato e SQL sulle estensioni o differenze generali rispetto al motore di SQL database My della community.

Argomenti

- Parametri di configurazione Aurora MySQL
- Aurora Le mie variabili di stato SQL globali
- Eventi Aurora My wait SQL
- Aurora Il mio thread afferma SQL
- Livelli di isolamento di Aurora MySQL
- Aurora I miei suggerimenti SQL
- Aurora Il mio riferimento alla procedura SQL memorizzata
- Aurora My: tabelle information_schema SQL specifiche

Parametri di configurazione Aurora MySQL

La gestione del cluster di database di Amazon Aurora MySQL è uguale a quella di altre istanze database Amazon RDS, ovvero utilizza i parametri di un gruppo di parametri database. La differenza tra Amazon Aurora e gli altri motori di database consiste nel fatto che un cluster di database contiene più istanze database. Di conseguenza, alcuni dei parametri utilizzati per gestire il cluster di database Aurora MySQL si applicano all'intero cluster. Altri parametri si applicano solo a una particolare istanza database del cluster di database.

Per gestire i parametri a livello di cluster, utilizza i gruppi di parametri del cluster di database. Per gestire i parametri a livello di istanza, utilizza i gruppi di parametri di database. Ogni istanza database in un cluster di database Aurora MySQL è compatibile con il motore del database MySQL. Tuttavia, si applicano alcuni dei parametri del motore del database MySQL a livello di cluster e si gestiscono questi parametri utilizzando i gruppi di parametri del cluster di DB. Non è possibile trovare parametri a livello di cluster nel gruppo di parametri DB per un'istanza in un cluster di database Aurora. Un elenco di parametri a livello di cluster è disponibile più avanti in questo argomento.

Puoi gestire i parametri sia a livello di cluster che a livello di istanza utilizzando l'API Amazon RDS AWS CLI o l' AWS Management Console API Amazon RDS. Sono disponibili comandi separati per la gestione dei parametri a livello di cluster e a livello di istanza. Ad esempio, è possibile utilizzare il comando CLI modify-db-cluster-parameter-group per gestire i parametri a livello di cluster in un

gruppo di parametri del cluster DB. È possibile utilizzare il comando modify-db-parameter-groupCLI per gestire i parametri a livello di istanza in un gruppo di parametri DB per un'istanza DB in un cluster DB.

Puoi visualizzare i parametri a livello di cluster e quelli a livello di istanza nella console oppure tramite l'interfaccia CLI o l'API RDS. Ad esempio, è possibile utilizzare il describe-db-cluster-parameters AWS CLI comando per visualizzare i parametri a livello di cluster in un gruppo di parametri del cluster DB. È possibile utilizzare il comando describe-db-parametersCLI per visualizzare i parametri a livello di istanza in un gruppo di parametri DB per un'istanza DB in un cluster DB.

> **Note**
>
> Ogni gruppo di parametri predefinito contiene i valori predefiniti per tutti i parametri del gruppo. Se il parametro ha come valore "engine default", consulta la documentazione MySQL o PostgreSQL specifica per la versione per il valore predefinito effettivo.
> Salvo diversa indicazione, i parametri elencati nelle tabelle seguenti sono validi per Aurora MySQL versione 2 e 3.

Per ulteriori informazioni sui gruppi di parametri database, consulta . Per regole e restrizioni per i cluster Aurora Serverless v1, consulta Gruppi di parametri per Aurora Serverless v1.

Argomenti

- Parametri a livello di cluster
- Parametri a livello di istanza
- I parametri MySQL seguenti non si applicano ad Aurora MySQL

Parametri a livello di cluster

La tabella seguente mostra tutti i parametri che si applicano all'intero cluster di database Aurora MySQL.

Nome del parametro	Modificabili	Note
`aurora_binlog_read_buffer_size`	Sì	Colpisce solo i cluster che utilizzano la replica binary log (binlog). Per informazioni sulla replica binlog, vedere Replica

Nome del parametro	Modificabili	Note
		tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari). Rimosso da Aurora MySQL versione 3.
`aurora_binlog_replication_max_yield_seconds`	Sì	Colpisce solo i cluster che utilizzano la replica binary log (binlog). Per informazioni sulla replica binlog, vedere Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).
`aurora_binlog_replication_sec_index_parallel_workers`	Sì	Imposta il numero totale di thread paralleli disponibili per applicare le modifiche dell'indice secondario durante la replica delle transazioni per tabelle di grandi dimensioni con più di un indice secondario. Il parametro è impostato su `0` (disabilitato) per impostazione predefinita. Questo parametro è disponibile in Aurora MySQL versione 306 e successive. Per ulteriori informazioni, consulta Ottimizzazione della replica dei log binari per Aurora MySQL.
`aurora_binlog_use_large_read_buffer`	Sì	Colpisce solo i cluster che utilizzano la replica binary log (binlog). Per informazioni sulla replica binlog, vedere Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari). Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
aurora_disable_hash_join	Sì	Imposta questo parametro su ON per disabilitare l'ottimizzazione dell'hash join in Aurora MySQL versione 2.09 o successiva. Non è supportato per la versione 3. Per ulteriori informazioni, consulta Interrogazione parallela per Amazon Aurora My SQL.
aurora_enable_replica_log_compression	Sì	Per ulteriori informazioni, consulta Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL. Non si applica ai cluster che fanno parte di un database globale Aurora. Rimosso da Aurora MySQL versione 3.
aurora_enable_repl_bin_log_filtering	Sì	Per ulteriori informazioni, consulta Considerazioni sulle prestazioni per la replica Amazon Aurora MySQL. Non si applica ai cluster che fanno parte di un database globale Aurora. Rimosso da Aurora MySQL versione 3.
aurora_enable_staggered_replica_restart	Sì	Questa impostazione è disponibile in Aurora MySQL versione 3, ma non viene utilizzata.
aurora_enable_zdr	Sì	Questa impostazione è attivata per impostazione predefinita in Aurora MySQL 2.10 e versioni successive. Per ulteriori informazioni, consulta Riavvio senza tempi di inattività (ZDR) per Amazon Aurora My SQL.

Nome del parametro	Modificabili	Note
`aurora_enhanced_binlog`	Sì	Impostare il valore di questo parametro su 1 per attivare il file di log binario avanzato in Aurora MySQL versione 3.03.1 e successive. Per ulteriori informazioni, consulta Configurazione di binlog avanzato per Aurora MySQL.
`aurora_full_double_precision_in_json`	Sì	Imposta il valore di questo parametro per consentire l'analisi dei numeri in virgola mobile nei documenti JSON con la massima precisione.
`aurora_jemalloc_background_thread`	Sì	Utilizzate questo parametro per abilitare un thread in background per eseguire operazioni di manutenzione della memoria. I valori consentiti sono `0` (disabilitato) e 1 (abilitato). Il valore predefinito è `0`. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.

Nome del parametro	Modificabili	Note
`aurora_jemalloc_dirty_decay_ms`	Sì	Utilizzate questo parametro per conservare la memoria liberata per un determinato periodo di tempo (in millisecondi). La conservazione della memoria consente un riutilizzo più rapido. I valori consentiti sono `0`–`18446744073709551615`. Il valore predefinito è `10000` (10 secondi). È possibile utilizzare un ritardo più breve per evitare out-of-memory problemi, a scapito di un rallentamento delle prestazioni. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.
`aurora_jemalloc_tcache_enabled`	Sì	Utilizzate questo parametro per servire piccole richieste di memoria (fino a 32 KB) in una cache locale del thread, bypassando le arene di memoria. I valori consentiti sono `0` (disabilitato) e 1 (abilitato). Il valore predefinito è 1. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.
`aurora_load_from_s3_role`	Sì	Per ulteriori informazioni, consulta Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3. Attualmente non disponibile in Aurora MySQL versione 3. Utilizza `aws_default_s3_role`.

Nome del parametro	Modificabili	Note
`aurora_mask_password_hashes_type`	Sì	Questa impostazione è attivata per impostazione predefinita in Aurora MySQL 2.11 e versioni successive. Usa questa impostazione per mascherare gli hash delle password di Aurora MySQL nei log delle query lente e di audit. I valori consentiti sono `0` e `1` (impostazione predefinita). Se è impostata su `1`, le password vengono registrate come `<secret>`. Se è impostata su `0`, le password vengono registrate come valori hash (#).
`aurora_select_into_s3_role`	Sì	Per ulteriori informazioni, consulta Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3. Attualmente non disponibile in Aurora MySQL versione 3. Utilizza `aws_default_s3_role` .

Nome del parametro	Modificabili	Note
`authentication_kerberos_caseins_cmp`	Sì	Controlla il confronto dei nomi utente senza distinzione tra maiuscole e minuscole per il plugin `authentication_kerberos` . Impostalo su `true` per il confronto senza distinzione tra maiuscole e minuscole. Per impostazione predefinita, viene utilizzato il confronto con distinzione tra maiuscole e minuscole (`false`). Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos per Aurora My SQL. Questo parametro è disponibile in Aurora MySQL versione 3.03 e versioni successive.
`auto_increment_increment`	Sì	Nessuno
`auto_increment_offset`	Sì	Nessuno
`aws_default_lambda_role`	Sì	Per ulteriori informazioni, consulta Richiamo di una funzione Lambda da un cluster Amazon Aurora My DB SQL.

Nome del parametro	Modificabili	Note
`aws_default_s3_role`	Sì	Viene utilizzato quando si richiama l'istruzione `LOAD DATA FROM S3`, `LOAD XML FROM S3` o `SELECT INTO OUTFILE S3` dal cluster DB. In Aurora MySQL versione 2, il ruolo IAM specificato in questo parametro viene utilizzato se un ruolo IAM non è specificato per `aurora_load_from_s3_role` o `aurora_select_into_s3_role` per l'istruzione appropriata. In Aurora MySQL versione 3, il ruolo IAM specificato per questo parametro è sempre utilizzato. Per ulteriori informazioni, consulta Associazione di un ruolo IAM a un cluster DB Amazon Aurora MySQL.
`binlog_backup`	Sì	Impostare il valore di questo parametro su 0 per attivare il file di log binario avanzato in Aurora MySQL versione 3.03.1 e successive. È possibile disattivare questo parametro solo in caso di utilizzo di un file di log binario avanzato. Per ulteriori informazioni, consulta Configurazione di binlog avanzato per Aurora MySQL.

Nome del parametro	Modificabili	Note
`binlog_checksum`	Sì	L'API AWS CLI and RDS riporta un valore pari a `None` se questo parametro non è impostato. In tal caso, Aurora MySQL utilizza il valore predefinito del motore, ossia `CRC32`. Questo è diverso dall'impostazione esplicita di `NONE`, che disattiva il checksum.
`binlog-do-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog_format`	Sì	Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).
`binlog_group_commit_sync_delay`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog_group_commit_sync_no_delay_count`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog-ignore-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`binlog_replication_globaldb`	Sì	Impostare il valore di questo parametro su 0 per attivare il file di log binario avanzato in Aurora MySQL versione 3.03.1 e successive. È possibile disattivare questo parametro solo in caso di utilizzo di un file di log binario avanzato. Per ulteriori informazioni, consulta Configurazione di binlog avanzato per Aurora MySQL.
`binlog_row_image`	No	Nessuno

Nome del parametro	Modificabili	Note
binlog_row_metadata	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_row_value_options	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_rows_query_log_events	Sì	Nessuno
binlog_transaction_compression	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_transaction_compression_level_zstd	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_transaction_dependency_history_size	Sì	Questo parametro imposta un limite massimo al numero di hash di riga che vengono conservati in memoria e utilizzati per cercare la transazione dell'ultima modifica di una determinata riga. Una volta raggiunto questo numero di hash, la cronologia viene rimossa. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.
binlog_transaction_dependency_tracking	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
character-set-client-handshake	Sì	Nessuno
character_set_client	Sì	Nessuno
character_set_connection	Sì	Nessuno

Nome del parametro	Modificabili	Note
`character_set_database`	Sì	Il set di caratteri utilizzato dal database predefinito. Il valore predefinito è `utf8mb4`.
`character_set_filesystem`	Sì	Nessuno
`character_set_results`	Sì	Nessuno
`character_set_server`	Sì	Nessuno
`collation_connection`	Sì	Nessuno
`collation_server`	Sì	Nessuno
`completion_type`	Sì	Nessuno
`default_storage_engine`	No	I cluster Aurora MySQL utilizzano il motore di storage InnoDB per tutti i dati.
`enforce_gtid_consistency`	A volte	Modificabile in Aurora MySQL versione 2 e successive.
`event_scheduler`	Sì	Indica lo stato dell'utilità di pianificazione eventi. Modificabile solo a livello di cluster in Aurora MySQL versione 3.
`gtid-mode`	A volte	Modificabile in Aurora MySQL versione 2 e successive.

Nome del parametro	Modificabili	Note
`information_schema_stats_expiry`	Sì	Il numero di secondi dopo il quale il server del database MySQL recupera i dati dal motore di archiviazione e sostituisce i dati nella cache. I valori consentiti sono `0`–`31536000`. Questo parametro si applica ad Aurora MySQL versione 3.
`init_connect`	Sì	Il comando che deve essere eseguito dal server per ogni client che si connette. Utilizza le virgolette doppie (") per le impostazioni per evitare errori di connessione, ad esempio: `SET optimizer_switch="hash_join =off"` In Aurora MySQL versione 3, questo parametro non si applica agli utenti che dispongono del privilegio `CONNECTION_ADMIN` , incluso l'utente master Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.
`innodb_adaptive_hash_index`	Sì	È possibile modificare questo parametro a livello di cluster database in Aurora MySQL versioni 2 e 3. L'indice adattivo Hash non è supportato nelle istanze database di lettura.

Nome del parametro	Modificabili	Note
innodb_aurora_instant_alter_column_allowed	Sì	Controlla se l'algoritmo INSTANT può essere utilizzato per operazioni ALTER COLUMN a livello globale. I valori validi consentiti sono i seguenti: • 0— L'INSTANTalgoritmo non è consentito per ALTER COLUMN le operazioni (OFF). Torna ad altri algoritmi. • 1— L'INSTANTalgoritmo è consentito per ALTER COLUMN le operazioni ()ON. Si tratta del valore di default. Per ulteriori informazioni, consulta la pagina relativa alle operazioni sulle colonne nella documentazione di MySQL. Questo parametro si applica solo ad Aurora MySQL 3.05 e versioni successive.
innodb_autoinc_lock_mode	Sì	Nessuno
innodb_checksums	No	Rimosso da Aurora MySQL versione 3.
innodb_cmp_per_index_enabled	Sì	Nessuno
innodb_commit_concurrency	Sì	Nessuno
innodb_data_home_dir	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.

Nome del parametro	Modificabili	Note
`innodb_deadlock_detect`	Sì	Questa opzione viene utilizzata per disabilitare il rilevamento del deadlock in Aurora MySQL versione 2.11 e successive e versione 3. Sui sistemi ad alta simultaneità, il rilevamento del deadlock può causare un rallentamento quando numerosi thread attendono lo stesso blocco. Per ulteriori informazioni su questo parametro, consulta la documentazione di MySQL.
`innodb_default_row_format`	Sì	Questo parametro definisce il formato di riga predefinito per le tabelle InnoDB (incluse le tabelle temporanee InnoDB create dall'utente). Si applica ad Aurora MySQL versioni 2 e 3. Il valore può essere `DYNAMIC`, `COMPACT` o `REDUNDANT`.
`innodb_file_per_table`	Sì	Questo parametro influisce sul modo in cui è organizzata l'archiviazione della tabella. Per ulteriori informazi oni, consulta Dimensionamento dello storage.

Nome del parametro	Modificabili	Note
innodb_flush_log_at_trx_commit	Sì	Si consiglia vivamente di utilizzare il valore predefinito di1. In Aurora MySQL versione 3, prima di poter impostare questo parametro su un valore diverso da1, è necessario impostare il valore di to. innodb_trx_commit_allow_data_loss 1 Per ulteriori informazioni, consulta Configurazione della frequenza di svuotamento del buffer dei registri.
innodb_ft_max_token_size	Sì	Nessuno
innodb_ft_min_token_size	Sì	Nessuno
innodb_ft_num_word_optimize	Sì	Nessuno
innodb_ft_sort_pll_degree	Sì	Nessuno
innodb_online_alter_log_max_size	Sì	Nessuno
innodb_optimize_fulltext_only	Sì	Nessuno
innodb_page_size	No	Nessuno
innodb_print_all_deadlocks	Sì	Quando è attivo, registra le informazioni su tutti i deadlock di InnoDB nel log degli errori di Aurora MySQL. Per ulteriori informazioni, consulta Riduzione al minimo e risoluzione dei problemi relativi ai deadlock di Aurora My SQL.

Nome del parametro	Modificabili	Note
innodb_purge_batch_size	Sì	Nessuno
innodb_purge_threads	Sì	Nessuno
innodb_rollback_on_timeout	Sì	Nessuno
innodb_rollback_segments	Sì	Nessuno
innodb_spin_wait_delay	Sì	Nessuno
innodb_strict_mode	Sì	Nessuno
innodb_support_xa	Sì	Rimosso da Aurora MySQL versione 3.
innodb_sync_array_size	Sì	Nessuno
innodb_sync_spin_loops	Sì	Nessuno
innodb_stats_include_delete_marked	Sì	Quando questo parametro è abilitato, InnoDB include i record contrassegnati per l'eliminazione durante il calcolo delle statistiche di ottimizzazione persistenti. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.
innodb_table_locks	Sì	Nessuno

Nome del parametro	Modificabili	Note
innodb_trx_commit_allow_data_loss	Sì	In Aurora MySQL versione 3, imposta il valore di questo parametro su in 1 modo da poter modificare il valore di. innodb_flush_log_at_trx_commit Il valore predefinito di innodb_trx_commit_allow_data_loss è 0. Per ulteriori informazioni, consulta Configurazione della frequenza di svuotamento del buffer dei registri.
innodb_undo_directory	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
internal_tmp_disk_storage_engine	Sì	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti sono INNODB e MYISAM. Questo parametro si applica ad Aurora MySQL versione 2.
internal_tmp_mem_storage_engine	Sì	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti sono MEMORY e TempTable . Questo parametro si applica ad Aurora MySQL versione 3.
key_buffer_size	Sì	Cache per chiave per tabelle MyISAM. Per ulteriori informazioni, consultare keycache->cache_lock mutex.

Nome del parametro	Modificabili	Note
`lc_time_names`	Sì	Nessuno
`log_error_suppression_list`	Sì	Speciifica un elenco di codici di errore che non sono registrati nel registro degli errori di MySQL. Ciò consente di ignorare determinate condizioni di errore non critiche per mantenere puliti i registri degli errori. Per ulteriori informazioni, vedere log_error_suppression_list nella documentazione di MySQL. Questo parametro si applica a Aurora MySQL versione 3.03 e successive.
`low_priority_updates`	Sì	Le operazioni `INSERT`, `UPDATE`, `DELETE` e `LOCK TABLE WRITE` attendono fino a quando non ci sono più operazioni `SELECT` in sospeso. Questo parametro ha effetto solo sui motori di archiviazione che utilizzano solo il blocco a livello di tabella (MyISAM, MEMORY, MERGE). Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`lower_case_table_names`	Sì (Aurora MySQL versione 2) Solo al momento della creazione del cluster (Aurora MySQL versione 3)	In Aurora MySQL versione 2.10 e successive 2.x, assicurati di riavviare tutte le istanze di lettura dopo aver modificato questa impostazione e riavviato l'istanza di scrittura. Per informazioni dettagliate, consultar e Riavvio di un cluster Aurora con disponibilità di lettura. In Aurora MySQL versione 3, il valore di questo parametro viene impostato in modo permanente al momento della creazione del cluster. Se si utilizza un valore non predefinito per questa opzione, impostare il gruppo di parametri personalizzati Aurora MySQL versione 3 prima dell'aggiornamento e specificare il gruppo di parametri durante l'operazione di ripristino dello snapshot che crea il cluster versione 3. Con un database globale Aurora basato su Aurora MySQL, non puoi eseguire un aggiornamento locale da Aurora MySQL versione 2 alla versione 3 se il parametro `lower_case_table_names` è attivato. Per ulteriori informazioni sui metodi disponibili all'uso, consulta Aggiornamenti di una versione principale.
`master-info-repository`	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
master_verify_checksum	Sì	Aurora MySQL versione 2. Utilizza source_verify_checksum in Aurora MySQL versione 3.
max_delayed_threads	Sì	Imposta il numero massimo di thread per gestire le istruzioni INSERT DELAYED. Questo parametro si applica ad Aurora MySQL versione 3.
max_error_count	Sì	Il numero massimo di messaggi di errore, avviso e note da archiviare per la visualizzazione. Questo parametro si applica ad Aurora MySQL versione 3.
max_execution_time	Sì	Il timeout per l'esecuzione SELECT delle istruzioni, in millisecondi. Il valore può provenire da —. 0 184467440 73709551615 Se impostato su0, non è previsto alcun timeout. Per ulteriori informazioni, vedere max_execution_time nella documenta zione di MySQL.
min_examined_row_limit	Sì	Utilizza questo parametro per impedire la registrazione delle query che esaminano un numero di righe inferiore a quello specificato.
partial_revokes	No	Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`preload_buffer_size`	Sì	La dimensione del buffer allocato durante il precaricamento degli indici. Questo parametro si applica ad Aurora MySQL versione 3.
`query_cache_type`	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`read_only`	Sì	Quando questo parametro è attivato, il server non consente aggiornamenti tranne quelli eseguiti dai thread di replica. Per Aurora MySQL versione 2, i valori validi sono i seguenti: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` — `ON` per repliche di lettura. Si tratta del valore di default. • `{TrueIfClusterReplica}` — `ON` per cluster di replica come repliche di lettura interregionali, cluster secondari in un database globale Aurora e distribuzioni blu/verdi. Per Aurora MySQL versione 3, i valori validi sono i seguenti: • `0OFF`—. Questo è il valore predefinito. • `1` – `ON` • `{TrueIfClusterReplica}` — `ON` per cluster di replica come repliche di lettura interregionali, cluster secondari in un database globale Aurora e distribuzioni blu/verdi. In Aurora MySQL versione 3, questo parametro non si applica agli utenti che dispongono del privilegio `CONNECTIO N_ADMIN` , incluso l'utente master

Nome del parametro	Modificabili	Note
		Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.
relay-log-space-limit	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
replica_parallel_type	Sì	Questo parametro consente l'esecuzione parallela sulla replica di tutti i thread di cui non è stato eseguito il commit già in fase di preparazione, senza violare la coerenza. Si applica ad Aurora MySQL versione 3. In Aurora MySQL versione 3.03.* e versioni precedenti, il valore predefinito è DATABASE. In Aurora MySQL versione 3.04 e versioni successive, il valore predefinito è LOGICAL_CLOCK.
replica_preserve_commit_order	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
replica_transaction_retries	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
replica_type_conversions	Sì	Questo parametro determina il tipo di conversione utilizzato nelle repliche. I valori consentiti sono ALL_LOSSY , ALL_NON_LOSSY , ALL_SIGNED e ALL_UNSIGNED . Per ulteriori informazioni, consulta l'argomento relativo alla replica con definizioni di tabelle diverse per origine e replica nella documentazione di MySQL. Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`replicate-do-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-do-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-ignore-db`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-ignore-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-wild-do-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`replicate-wild-ignore-table`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`require_secure_transport`	Sì	Questo parametro si applica ad Aurora MySQL versione 2 e 3. Per ulteriori informazioni, consulta Connessioni TLS ai cluster Aurora MySQL DB.
`rpl_read_size`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`server_audit_cw_upload`	No	Questo parametro è obsoleto in Aurora MySQL. Utilizza `server_audit_logs_upload` . Per ulteriori informazioni, consulta Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch .

Nome del parametro	Modificabili	Note
`server_audit_events`	Sì	Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.
`server_audit_excl_users`	Sì	Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.
`server_audit_incl_users`	Sì	Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.
`server_audit_logging`	Sì	Per istruzioni su come caricare i log su Amazon CloudWatch Logs, consulta. Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch
`server_audit_logs_upload`	Sì	Puoi pubblicare i log di controllo su Logs abilitando Advanced Auditing CloudWatch e impostando questo parametro su. 1 Il valore predefinito per il parametro `server_audit_logs_upload` è `0`. Per ulteriori informazioni, consulta Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch .
`server_id`	No	Nessuno
`skip-character-set-client-handshake`	Sì	Nessuno
`skip_name_resolve`	No	Nessuno

Nome del parametro	Modificabili	Note
`slave-skip-errors`	Sì	Si applica solo ai cluster Aurora MySQL versione 2, con compatibilità MySQL 5.7.
`source_verify_checksum`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`sync_frm`	Sì	Rimosso da Aurora MySQL versione 3.
`thread_cache_size`	Sì	Il numero di thread da memorizza re nella cache. Questo parametro si applica ad Aurora MySQL versioni 2 e 3.
`time_zone`	Sì	Per impostazione predefinita, il fuso orario per un cluster Aurora DB è Universal Time Coordinated (UTC). Tuttavia, puoi impostare il fuso orario delle istanze del cluster DB sul fuso orario locale dell'applicazione. Per ulteriori informazioni, consulta Fuso orario locale per i cluster DB Amazon Aurora.
`tls_version`	Sì	Per ulteriori informazioni, consulta Versioni TLS per Aurora MySQL.

Parametri a livello di istanza

La tabella seguente mostra tutti i parametri che si applicano a una specifica istanza database di un cluster di database Aurora MySQL.

Nome del parametro	Modificabili	Note
`activate_all_roles_on_login`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`allow-suspicious-udfs`	No	Nessuno
`aurora_disable_hash_join`	Sì	Imposta questo parametro su `ON` per disabilitare l'ottimizzazione dell'hash join in Aurora MySQL versione 2.09 o successiva. Non è supportato per la versione 3. Per ulteriori informazioni, consulta Interrogazione parallela per Amazon Aurora My SQL.
`aurora_lab_mode`	Sì	Per ulteriori informazioni, consulta Modalità di laboratorio per Amazon Aurora MySQL. Rimosso da Aurora MySQL versione 3.
`aurora_oom_response`	Sì	Questo parametro è supportato per Aurora MySQL versione 2 e 3. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL.
`aurora_parallel_query`	Sì	Imposta su `ON` per abilitare la query parallela in Aurora MySQL versione 2.09 o successive. Il vecchio parametro `aurora_pq` non viene utilizzato in queste versioni. Per ulteriori informazioni, consulta Interrogazione parallela per Amazon Aurora My SQL.

Nome del parametro	Modificabili	Note
`aurora_pq`	Sì	Imposta su `OFF` per disattivare la query parallela per istanze database specifich e nelle versioni di Aurora MySQL precedenti alla 2.09. Nella versione 2.09 o successive, attiva e disattiva la query parallela con `aurora_parallel_qu ery` . Per ulteriori informazioni, consulta Interrogazione parallela per Amazon Aurora My SQL.
`aurora_read_replica_read_co mmitted`	Sì	Abilita il livello di isolamento `READ COMMITTED` per le repliche Aurora e modifica il comportamento di isolament o per ridurre il ritardo di rimozione durante le query a esecuzione prolungat a. Abilita questa impostazione solo sei informato sulle modifiche al comportam ento e su come influiscono sui risultati della query. Ad esempio, questa impostazione utilizza un isolament o meno rigoroso rispetto all'impos tazione predefinita di MySQL. Quando è abilitata, le query a esecuzione prolungata potrebbero visualizzare più di una copia della stessa riga perché Aurora riorganizza i dati della tabella mentre la query è in esecuzione. Per ulteriori informazioni, consulta Livelli di isolamento di Aurora MySQL.

Nome del parametro	Modificabili	Note
`aurora_tmptable_enable_per_table_limit`	Sì	Determina se il parametro `tmp_table_size` controlla le dimensioni massime delle tabelle temporane e in memoria create dal motore di storage `TempTable` in Aurora MySQL versione 3.04 e successive. Per ulteriori informazioni, consulta Limitazione delle dimensioni delle tabelle temporanee interne in memoria.
`aurora_use_vector_instructions`	Sì	Quando questo parametro è abilitato , Aurora MySQL utilizza istruzioni di elaborazione vettoriale ottimizzate fornite da modern CPUs per migliorare le prestazioni sui carichi di lavoro a uso intensivo di I/O. Questa impostazione è abilitata per impostazione predefinita in Aurora MySQL 3.05 e versioni successive.
`autocommit`	Sì	Nessuno
`automatic_sp_privileges`	Sì	Nessuno
`back_log`	Sì	Nessuno
`basedir`	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
`binlog_cache_size`	Sì	Nessuno
`binlog_max_flush_queue_time`	Sì	Nessuno

Nome del parametro	Modificabili	Note
binlog_order_commits	Sì	Nessuno
binlog_stmt_cache_size	Sì	Nessuno
binlog_transaction_compression	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
binlog_transaction_compression_level_zstd	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
bulk_insert_buffer_size	Sì	Nessuno
concurrent_insert	Sì	Nessuno
connect_timeout	Sì	Nessuno
core-file	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
datadir	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
default_authentication_plugin	No	Questo parametro si applica ad Aurora MySQL versione 3.
default_time_zone	No	Nessuno
default_tmp_storage_engine	Sì	Il motore di archiviazione predefinito per le tabelle temporanee.
default_week_format	Sì	Nessuno
delay_key_write	Sì	Nessuno
delayed_insert_limit	Sì	Nessuno
delayed_insert_timeout	Sì	Nessuno

Nome del parametro	Modificabili	Note
delayed_queue_size	Sì	Nessuno
div_precision_increment	Sì	Nessuno
end_markers_in_json	Sì	Nessuno
eq_range_index_dive_limit	Sì	Nessuno
event_scheduler	A volte	Indica lo stato dell'utilità di pianificazione eventi. Modificabile solo a livello di cluster in Aurora MySQL versione 3.
explicit_defaults_for_timestamp	Sì	Nessuno
flush	No	Nessuno
flush_time	Sì	Nessuno
ft_boolean_syntax	No	Nessuno
ft_max_word_len	Sì	Nessuno
ft_min_word_len	Sì	Nessuno
ft_query_expansion_limit	Sì	Nessuno
ft_stopword_file	Sì	Nessuno
general_log	Sì	Per istruzioni sul caricamento dei log in Logs, consulta. CloudWatch Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch
general_log_file	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.

Nome del parametro	Modificabili	Note
`group_concat_max_len`	Sì	Nessuno
`host_cache_size`	Sì	Nessuno
`init_connect`	Sì	Il comando che deve essere eseguito dal server per ogni client che si connette. Utilizza le virgolette doppie (") per le impostazioni per evitare errori di connessione, ad esempio: `SET optimizer_switch="hash_join =off"` In Aurora MySQL versione 3, questo parametro non si applica agli utenti che dispongono del privilegio `CONNECTIO N_ADMIN` , incluso l'utente master di Aurora. Per ulteriori informazioni, consulta Privilegio basato sui ruoli.
`innodb_adaptive_hash_index`	Sì	È possibile modificare questo parametro a livello di istanza database in Aurora MySQL versione 2. È modificabile solo a livello di cluster database in Aurora MySQL versione 3. L'indice adattivo Hash non è supportato nelle istanze database di lettura.
`innodb_adaptive_max_sleep_d elay`	Sì	La modifica di questo parametro non ha alcun effetto, perché `innodb_th read_concurrency` è sempre 0 per Aurora.

Nome del parametro	Modificabili	Note
innodb_aurora_max_partitions_for_range	Sì	In alcuni casi in cui le statistiche persistenti non sono disponibili, è possibile utilizzare questo parametro per migliorare le prestazioni delle stime del numero di righe sulle tabelle partizionate. È possibile impostarlo su un valore compreso tra 0 e 8192, ovvero il valore determina il numero di partizioni da controllare durante la stima del numero di righe. Il valore predefinito è 0, che stima l'utilizzo di tutte le partizioni, coerente con il comportamento predefinito di MySQL. Questo parametro è disponibile per Aurora MySQL versione 3.03.1 e versioni successive.
innodb_autoextend_increment	Sì	Nessuno
innodb_buffer_pool_dump_at_shutdown	No	Nessuno
innodb_buffer_pool_dump_now	No	Nessuno
innodb_buffer_pool_filename	No	Nessuno
innodb_buffer_pool_load_abort	No	Nessuno
innodb_buffer_pool_load_at_startup	No	Nessuno

Nome del parametro	Modificabili	Note
innodb_buffer_pool_load_now	No	Nessuno
innodb_buffer_pool_size	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore DBInstanceClassMemory nella formula, vedi Variabili di formula dei parametri database.
innodb_change_buffer_max_size	No	Aurora MySQL non utilizza affatto il buffer di modifica InnoDB.
innodb_compression_failure_threshold_pct	Sì	Nessuno
innodb_compression_level	Sì	Nessuno
innodb_compression_pad_pct_max	Sì	Nessuno
innodb_concurrency_tickets	Sì	La modifica di questo parametro non ha alcun effetto, perché innodb_thread_concurrency è sempre 0 per Aurora.

Nome del parametro	Modificabili	Note
innodb_deadlock_detect	Sì	Questa opzione viene utilizzata per disabilitare il rilevamento del deadlock in Aurora MySQL versione 2.11 e successive e versione 3. Sui sistemi ad alta simultaneità, il rilevamento del deadlock può causare un rallentamento quando numerosi thread attendono lo stesso blocco. Per ulteriori informazioni su questo parametro, consulta la documentazione di MySQL.
innodb_file_format	Sì	Rimosso da Aurora MySQL versione 3.
innodb_flushing_avg_loops	No	Nessuno
innodb_force_load_corrupted	No	Nessuno
innodb_ft_aux_table	Sì	Nessuno
innodb_ft_cache_size	Sì	Nessuno
innodb_ft_enable_stopword	Sì	Nessuno
innodb_ft_server_stopword_table	Sì	Nessuno
innodb_ft_user_stopword_table	Sì	Nessuno
innodb_large_prefix	Sì	Rimosso da Aurora MySQL versione 3.
innodb_lock_wait_timeout	Sì	Nessuno
innodb_log_compressed_pages	No	Nessuno

Nome del parametro	Modificabili	Note
innodb_lru_scan_depth	Sì	Nessuno
innodb_max_purge_lag	Sì	Nessuno
innodb_max_purge_lag_delay	Sì	Nessuno
innodb_monitor_disable	Sì	Nessuno
innodb_monitor_enable	Sì	Nessuno
innodb_monitor_reset	Sì	Nessuno
innodb_monitor_reset_all	Sì	Nessuno
innodb_old_blocks_pct	Sì	Nessuno
innodb_old_blocks_time	Sì	Nessuno
innodb_open_files	Sì	Nessuno
innodb_print_all_deadlocks	Sì	Quando è attivo, registra le informazioni su tutti i deadlock di InnoDB nel log degli errori di Aurora MySQL. Per ulteriori informazioni, consulta Riduzione al minimo e risoluzione dei problemi relativi ai deadlock di Aurora My SQL.
innodb_random_read_ahead	Sì	Nessuno
innodb_read_ahead_threshold	Sì	Nessuno
innodb_read_io_threads	No	Nessuno

Nome del parametro	Modificabili	Note
innodb_read_only	No	Aurora MySQL gestisce l'istanza di sola lettura e l'istanza read/write state of DB instances based on the type of cluster. For example, a provisioned cluster has one read/write DB (l'istanza principale) e tutte le altre istanze del cluster sono di sola lettura (le repliche Aurora).
innodb_replication_delay	Sì	Nessuno
innodb_sort_buffer_size	Sì	Nessuno
innodb_stats_auto_recalc	Sì	Nessuno
innodb_stats_method	Sì	Nessuno
innodb_stats_on_metadata	Sì	Nessuno
innodb_stats_persistent	Sì	Nessuno
innodb_stats_persistent_sample_pages	Sì	Nessuno
innodb_stats_transient_sample_pages	Sì	Nessuno
innodb_thread_concurrency	No	Nessuno
innodb_thread_sleep_delay	Sì	La modifica di questo parametro non ha alcun effetto, perché innodb_thread_concurrency è sempre 0 per Aurora.

Nome del parametro	Modificabili	Note
`interactive_timeout`	Sì	Aurora valuta il valore minimo di `interactive_timeout` e `wait_timeout` . Quindi usa quel minimo come timeout per terminare tutte le sessioni inattive, sia interattive sia non interattive.
`internal_tmp_disk_storage_engine`	Sì	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti sono `INNODB` e `MYISAM`. Questo parametro si applica ad Aurora MySQL versione 2.
`internal_tmp_mem_storage_engine`	A volte	Controlla quale motore di archiviazione in memoria viene utilizzato per le tabelle temporanee interne. I valori consentiti per le istanze Writer DB `MEMORY` sono e. `TempTable` Per le istanze DB Reader, questo parametro è impostato su `TempTable` e non può essere modificato. Questo parametro si applica ad Aurora MySQL versione 3.
`join_buffer_size`	Sì	Nessuno
`keep_files_on_create`	Sì	Nessuno
`key_buffer_size`	Sì	Cache per chiave per tabelle MyISAM. Per ulteriori informazioni, consultare keycache->cache_lock mutex.
`key_cache_age_threshold`	Sì	Nessuno

Nome del parametro	Modificabili	Note
key_cache_block_size	Sì	Nessuno
key_cache_division_limit	Sì	Nessuno
local_infile	Sì	Nessuno
lock_wait_timeout	Sì	Nessuno
log-bin	No	L'impostazione di binlog_format su STATEMENT , MIXED o ROW imposta automaticamente log-bin su ON. L'impostazione di binlog_format su OFF imposta automaticamente log-bin su OFF. Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).
log_bin_trust_function_creators	Sì	Nessuno
log_bin_use_v1_row_events	Sì	Rimosso da Aurora MySQL versione 3.
log_error	No	Nessuno
log_error_suppression_list	Sì	Speciifica un elenco di codici di errore che non sono registrati nel registro degli errori di MySQL. Ciò consente di ignorare determinate condizioni di errore non critiche per mantenere puliti i registri degli errori. Per ulteriori informazioni, vedere log_error_suppression_list nella documentazione di MySQL. Questo parametro si applica a Aurora MySQL versione 3.03 e successive.

Nome del parametro	Modificabili	Note
`log_output`	Sì	Nessuno
`log_queries_not_using_indexes`	Sì	Nessuno
`log_slave_updates`	No	Aurora MySQL versione 2. Utilizza `log_replica_updates` in Aurora MySQL versione 3.
`log_replica_updates`	No	Aurora MySQL versione 3
`log_throttle_queries_not_using_indexes`	Sì	Nessuno
`log_warnings`	Sì	Rimosso da Aurora MySQL versione 3.
`long_query_time`	Sì	Nessuno
`low_priority_updates`	Sì	Le operazioni `INSERT`, `UPDATE`, `DELETE` e `LOCK TABLE WRITE` attendono fino a quando non ci sono più operazioni `SELECT` in sospeso. Questo parametro ha effetto solo sui motori di archiviazione che utilizzano solo il blocco a livello di tabella (MyISAM, MEMORY, MERGE). Questo parametro si applica ad Aurora MySQL versione 3.
`max_allowed_packet`	Sì	Nessuno
`max_binlog_cache_size`	Sì	Nessuno
`max_binlog_size`	No	Nessuno
`max_binlog_stmt_cache_size`	Sì	Nessuno

Nome del parametro	Modificabili	Note
`max_connect_errors`	Sì	Nessuno
`max_connections`	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore `DBInstanceClassMemory` nella formula, vedi Variabili di formula dei parametri database. Per i valori predefiniti a seconda della classe di istanza, vedi Numero massimo di connessioni a un'istanza database Aurora MySQL.
`max_delayed_threads`	Sì	Imposta il numero massimo di thread per gestire le istruzioni `INSERT DELAYED`. Questo parametro si applica ad Aurora MySQL versione 3.
`max_error_count`	Sì	Il numero massimo di messaggi di errore, avviso e note da archiviare per la visualizzazione. Questo parametro si applica ad Aurora MySQL versione 3.
`max_execution_time`	Sì	Il timeout per l'esecuzione `SELECT` delle istruzioni, in millisecondi. Il valore può provenire da —. `0` `184467440 73709551615` Se impostato su`0`, non è previsto alcun timeout. Per ulteriori informazioni, vedere max_execution_time nella documenta zione di MySQL.

Nome del parametro	Modificabili	Note
max_heap_table_size	Sì	Nessuno
max_insert_delayed_threads	Sì	Nessuno
max_join_size	Sì	Nessuno
max_length_for_sort_data	Sì	Rimosso da Aurora MySQL versione 3.
max_prepared_stmt_count	Sì	Nessuno
max_seeks_for_key	Sì	Nessuno
max_sort_length	Sì	Nessuno
max_sp_recursion_depth	Sì	Nessuno
max_tmp_tables	Sì	Rimosso da Aurora MySQL versione 3.
max_user_connections	Sì	Nessuno
max_write_lock_count	Sì	Nessuno
metadata_locks_cache_size	Sì	Rimosso da Aurora MySQL versione 3.
min_examined_row_limit	Sì	Utilizza questo parametro per impedire la registrazione delle query che esaminano un numero di righe inferiore a quello specificato. Questo parametro si applica ad Aurora MySQL versione 3.
myisam_data_pointer_size	Sì	Nessuno
myisam_max_sort_file_size	Sì	Nessuno
myisam_mmap_size	Sì	Nessuno
myisam_sort_buffer_size	Sì	Nessuno

Nome del parametro	Modificabili	Note
myisam_stats_method	Sì	Nessuno
myisam_use_mmap	Sì	Nessuno
net_buffer_length	Sì	Nessuno
net_read_timeout	Sì	Nessuno
net_retry_count	Sì	Nessuno
net_write_timeout	Sì	Nessuno
old-style-user-limits	Sì	Nessuno
old_passwords	Sì	Rimosso da Aurora MySQL versione 3.
optimizer_prune_level	Sì	Nessuno
optimizer_search_depth	Sì	Nessuno
optimizer_switch	Sì	Per informazioni sulle funzionalità Aurora MySQL che utilizzano questa opzione, consulta Le migliori pratiche con Amazon Aurora My SQL.
optimizer_trace	Sì	Nessuno
optimizer_trace_features	Sì	Nessuno
optimizer_trace_limit	Sì	Nessuno
optimizer_trace_max_mem_size	Sì	Nessuno
optimizer_trace_offset	Sì	Nessuno

Nome del parametro	Modificabili	Note
`performance-schema-consumer-events-waits-current`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro. `performance-schema-consumer-events-waits-current` Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance-schema-instrument`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro`performance-schema-instrument` . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance_schema`	Sì	Se la colonna Source è impostata su`Modified`, Performance Insights gestisce lo schema delle prestazioni. Per ulteriori informazioni sull'attivazione del Performance Schema, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_accounts_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro`performance_schema_accounts_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedere[Determinazione della gestione di Performance Schema da parte di Performance Insights].
`performance_schema_consumer_global_instrumentation`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_consumer_global_instrumentation`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedere[Determinazione della gestione di Performance Schema da parte di Performance Insights].
`performance_schema_consumer_thread_instrumentation`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_consumer_thread_instrumentation`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedere[Determinazione della gestione di Performance Schema da parte di Performance Insights].

Nome del parametro	Modificabili	Note
performance_schema_consumer_events_stages_current	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_stages_current . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_consumer_events_stages_history	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_stages_history . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_consumer_events_stages_history_long	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_stages_history_long . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_consumer_events_statements_current	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_statements_current . Per ulteriori informazioni sull'atti vazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_consumer_events_statements_history	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_statements_history . Per ulteriori informazioni sull'atti vazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_consumer_events_statements_history_long`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro`performance_schema_consumer_events_statements_history_long`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance_schema_consumer_events_waits_history`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_consumer_events_waits_history`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_consumer_events_waits_history_long	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_events_waits_history_long . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_consumer_statements_digest	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_consumer_statements_digest . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_digests_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_digests_size . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_events_stages_history_long_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_events_stages_history_long_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance_schema_events_stages_history_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_events_stages_history_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance_schema_events_statements_history_long_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_events_statements_history_long_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_events_statements_history_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_events_statements_history_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance_schema_events_transactions_history_long_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_events_transactions_history_long_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
`performance_schema_events_transactions_history_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro `performance_schema_events_transactions_history_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_events_waits_history_long_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_events_waits_history_long_size . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_events_waits_history_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_events_waits_history_size . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_hosts_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_hosts_size . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_cond_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_cond_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_cond_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_cond_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_digest_length	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_digest_length. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_file_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_file_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_file_handles	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_file_handles. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_file_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_file_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_index_stat	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_index_stat. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_memory_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_memory_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_metadata_locks	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_metadata_locks. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_mutex_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_mutex_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_mutex_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_mutex_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_prepared_statements_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_prepared_statements_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_program_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_program_instances . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_rwlock_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_rwlock_classes . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_rwlock_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_rwlock_instances . Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_socket_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_socket_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_socket_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_socket_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_sql_text_length	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_sql_text_length. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_stage_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_stage_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_statement_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_statement_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_statement_stack	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_statement_stack. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_table_handles	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_table_handles. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_table_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_table_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_table_lock_stat	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_table_lock_stat. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_max_thread_classes	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_max_thread_classes. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_max_thread_instances	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametroperformance_schema_max_thread_instances. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_session_connect_attrs_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_session_connect_attrs_size. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
performance_schema_setup_actors_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_setup_actors_size. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.
performance_schema_setup_objects_size	Sì	Se la colonna Source per il parametro performance_schema è impostata suModified, lo schema delle prestazioni utilizza il parametro performance_schema_setup_objects_size. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedereDeterminazione della gestione di Performance Schema da parte di Performance Insights.

Nome del parametro	Modificabili	Note
`performance_schema_show_processlist`	Sì	Questo parametro determina quale implementazione `SHOW PROCESSLIST` utilizzare: • L'implementazione predefinita esegue iterazioni tra i thread attivi dall'interno del gestore dei thread mantenendo un mutex globale. Ciò può causare un rallentamento delle prestazioni, in particolare su sistemi occupati. • L'implementazione `SHOW PROCESSLIST` alternativa è basata sulla tabella `processlist` dello schema di prestazioni. Questa implementazione esegue la query sui dati dei thread attivi dallo schema di prestazioni anziché dal gestore dei thread e non richiede un mutex. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3. Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro`performance_schema_show_processlist`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedere[Determinazione della gestione di Performance Schema da parte di Performance Insights]().

Nome del parametro	Modificabili	Note
`performance_schema_users_size`	Sì	Se la colonna Source per il parametro `performance_schema` è impostata su`Modified`, lo schema delle prestazioni utilizza il parametro`performance_schema_users_size`. Per ulteriori informazioni sull'attivazione dello schema delle prestazioni, vedere Determinazione della gestione di Performance Schema da parte di Performance Insights.
`pid_file`	No	Nessuno
`plugin_dir`	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
`port`	No	Aurora MySQL gestisce le proprietà della connessione e applica impostazioni coerenti per tutte le istanze database in un cluster.
`preload_buffer_size`	Sì	La dimensione del buffer allocato durante il precaricamento degli indici. Questo parametro si applica ad Aurora MySQL versione 3.
`profiling_history_size`	Sì	Nessuno
`query_alloc_block_size`	Sì	Nessuno
`query_cache_limit`	Sì	Rimosso da Aurora MySQL versione 3.
`query_cache_min_res_unit`	Sì	Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
query_cache_size	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore DBInstanceClassMemory nella formula, vedi Variabili di formula dei parametri database. Rimosso da Aurora MySQL versione 3.
query_cache_type	Sì	Rimosso da Aurora MySQL versione 3.
query_cache_wlock_invalidate	Sì	Rimosso da Aurora MySQL versione 3.
query_prealloc_size	Sì	Nessuno
range_alloc_block_size	Sì	Nessuno
read_buffer_size	Sì	Nessuno

Nome del parametro	Modificabili	Note
`read_only`	Sì	Quando questo parametro è attivato, il server non consente aggiornamenti tranne quelli eseguiti dai thread di replica. Per Aurora MySQL versione 2, i valori validi sono i seguenti: • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` — `ON` per repliche di lettura. Si tratta del valore di default. • `{TrueIfClusterReplica}` — `ON` per istanze in cluster di replica come repliche di lettura interregionali, cluster secondari in un database globale Aurora e distribuzioni blu/verdi . Ti consigliamo di utilizzare il gruppo di parametri del cluster di database in Aurora MySQL versione 2 per garantire che il parametro `read_only` venga applicato alle nuove istanze di scrittura in caso di failover. **Note** Le istanze di lettura sono sempre di sola lettura, perché Aurora MySQL imposta `innodb_read_only` su `1` su tutte le letture. Pertanto,

Nome del parametro	Modificabili	Note
		`read_only` è ridondante sulle istanze di lettura. È stato rimosso a livello di istanza da Aurora MySQL versione 3.
`read_rnd_buffer_size`	Sì	Nessuno
`relay-log`	No	Nessuno
`relay_log_info_repository`	Sì	Rimosso da Aurora MySQL versione 3.
`relay_log_recovery`	No	Nessuno
`replica_checkpoint_group`	Sì	Aurora MySQL versione 3
`replica_checkpoint_period`	Sì	Aurora MySQL versione 3
`replica_parallel_workers`	Sì	Aurora MySQL versione 3
`replica_pending_jobs_size_max`	Sì	Aurora MySQL versione 3
`replica_skip_errors`	Sì	Aurora MySQL versione 3
`replica_sql_verify_checksum`	Sì	Aurora MySQL versione 3
`safe-user-create`	Sì	Nessuno
`secure_auth`	Sì	Questo parametro è sempre attivato in Aurora MySQL versione 2. Il tentativo di disattivarlo genera un errore. Rimosso da Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
secure_file_priv	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
show_create_table_verbosity	Sì	Abilitando questa variabile, SHOW_CREATE_TABLE visualizza ROW_FORMAT a prescindere che si tratti del formato predefinito. Questo parametro si applica ad Aurora MySQL versione 2.12 e successive e versione 3.
skip-slave-start	No	Nessuno
skip_external_locking	No	Nessuno
skip_show_database	Sì	Nessuno
slave_checkpoint_group	Sì	Aurora MySQL versione 2. Utilizza replica_checkpoint_group in Aurora MySQL versione 3.
slave_checkpoint_period	Sì	Aurora MySQL versione 2. Utilizza replica_checkpoint_period in Aurora MySQL versione 3.
slave_parallel_workers	Sì	Aurora MySQL versione 2. Utilizza replica_parallel_workers in Aurora MySQL versione 3.
slave_pending_jobs_size_max	Sì	Aurora MySQL versione 2. Utilizza replica_pending_jobs_size_max in Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`slave_sql_verify_checksum`	Sì	Aurora MySQL versione 2. Utilizza `replica_sql_verify_checksum` in Aurora MySQL versione 3.
`slow_launch_time`	Sì	Nessuno
`slow_query_log`	Sì	Per istruzioni sul caricamento dei log in Logs, vedere. CloudWatch Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch
`slow_query_log_file`	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.
`socket`	No	Nessuno
`sort_buffer_size`	Sì	Nessuno
`sql_mode`	Sì	Nessuno
`sql_select_limit`	Sì	Nessuno
`stored_program_cache`	Sì	Nessuno
`sync_binlog`	No	Nessuno
`sync_master_info`	Sì	Nessuno
`sync_source_info`	Sì	Questo parametro si applica ad Aurora MySQL versione 3.
`sync_relay_log`	Sì	Rimosso da Aurora MySQL versione 3.
`sync_relay_log_info`	Sì	Nessuno
`sysdate-is-now`	Sì	Nessuno

Nome del parametro	Modificabili	Note
`table_cache_element_entry_ttl`	No	Nessuno
`table_definition_cache`	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore `DBInstanceClassMemory` nella formula, vedi Variabili di formula dei parametri database.
`table_open_cache`	Sì	Il valore predefinito è rappresentato da una formula. Per informazioni dettagliate su come viene calcolato il valore `DBInstanceClassMemory` nella formula, vedi Variabili di formula dei parametri database.
`table_open_cache_instances`	Sì	Nessuno
`temp-pool`	Sì	Rimosso da Aurora MySQL versione 3.
`temptable_max_mmap`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Per informazioni dettagliate, consultare Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.
`temptable_max_ram`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Per informazioni dettagliate, consultare Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.

Nome del parametro	Modificabili	Note
`temptable_use_mmap`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Per informazioni dettagliate, consultare Nuovo comportamento della tabella temporanea in Aurora MySQL versione 3.
`thread_cache_size`	Sì	Il numero di thread da memorizzare nella cache. Questo parametro si applica ad Aurora MySQL versioni 2 e 3.
`thread_handling`	No	Nessuno
`thread_stack`	Sì	Nessuno
`timed_mutexes`	Sì	Nessuno
`tmp_table_size`	Sì	Definisce le dimensioni massime delle tabelle temporanee in memoria create dal motore di storage `MEMORY` in Aurora MySQL versione 3. In Aurora MySQL versione 3.04 e successive, definisce le dimensioni massime delle tabelle temporanee in memoria create dal motore di storage `TempTable` quando `aurora_tmptable_enable_per_table_limit` è `ON`. Per ulteriori informazioni, consulta Limitazione delle dimensioni delle tabelle temporanee interne in memoria.
`tmpdir`	No	Aurora MySQL utilizza istanze gestite in cui non si accede direttamente al filesystem.

Nome del parametro	Modificabili	Note
`transaction_alloc_block_size`	Sì	Nessuno
`transaction_isolation`	Sì	Questo parametro si applica ad Aurora MySQL versione 3. Sostituisce `tx_isolation` .
`transaction_prealloc_size`	Sì	Nessuno
`tx_isolation`	Sì	Rimosso da Aurora MySQL versione 3. È sostituito da `transaction_isolation` .
`updatable_views_with_limit`	Sì	Nessuno
`validate-password`	No	Nessuno
`validate_password_dictionary_file`	No	Nessuno
`validate_password_length`	No	Nessuno
`validate_password_mixed_case_count`	No	Nessuno
`validate_password_number_count`	No	Nessuno
`validate_password_policy`	No	Nessuno
`validate_password_special_char_count`	No	Nessuno

Nome del parametro	Modificabili	Note
`wait_timeout`	Sì	Aurora valuta il valore minimo di `interactive_timeout` e `wait_timeout` . Quindi usa quel minimo come timeout per terminare tutte le sessioni inattive, sia interattive sia non interattive.

I parametri MySQL seguenti non si applicano ad Aurora MySQL

A causa delle differenze tra l'architettura di Aurora MySQL e quella di MySQL, alcuni parametri e variabili di stato MySQL non si applicano ad Aurora MySQL.

I parametri MySQL seguenti non si applicano ad Aurora MySQL. L'elenco non è completo.

- `activate_all_roles_on_login`: questo parametro non si applica ad Aurora MySQL versione 2. E' disponibile in Aurora MySQL versione 3.
- `big_tables`
- `bind_address`
- `character_sets_dir`
- `innodb_adaptive_flushing`
- `innodb_adaptive_flushing_lwm`
- `innodb_buffer_pool_chunk_size`
- `innodb_buffer_pool_instances`
- `innodb_change_buffering`
- `innodb_checksum_algorithm`
- `innodb_data_file_path`
- `innodb_dedicated_server`
- `innodb_doublewrite`
- `innodb_flush_log_at_timeout`: questo parametro non si applica ad Aurora MySQL. Per ulteriori informazioni, consulta Configurazione della frequenza di svuotamento del buffer dei registri.
- `innodb_flush_method`

- innodb_flush_neighbors
- innodb_io_capacity
- innodb_io_capacity_max
- innodb_log_buffer_size
- innodb_log_file_size
- innodb_log_files_in_group
- innodb_log_spin_cpu_abs_lwm
- innodb_log_spin_cpu_pct_hwm
- innodb_log_writer_threads
- innodb_max_dirty_pages_pct
- innodb_numa_interleave
- innodb_page_size
- innodb_redo_log_capacity
- innodb_redo_log_encrypt
- innodb_undo_log_encrypt
- innodb_undo_log_truncate
- innodb_undo_logs
- innodb_undo_tablespaces
- innodb_use_native_aio
- innodb_write_io_threads

Aurora Le mie variabili di stato SQL globali

Aurora My SQL include variabili di stato della community My SQL e variabili esclusive di Aurora. Puoi esaminare queste variabili per scoprire cosa succede all'interno del motore di database. Per ulteriori informazioni sulle variabili di stato nella community MySQL, consulta Server Status Variables nella documentazione della community My SQL 8.0.

È possibile trovare i valori correnti per le variabili di stato SQL globali Aurora My utilizzando un'istruzione come la seguente:

```
show global status like '%aurora%';
```

> **Note**
>
> Le variabili di stato globali vengono cancellate al riavvio del motore DB.

La tabella seguente descrive le variabili di stato globali utilizzate da Aurora MySQL.

Nome	Descrizione
`AuroraDb_commits`	Il numero totale di commit dall'ultimo riavvio.
`AuroraDb_commit_latency`	La latenza del commit aggregata dall'ultimo riavvio.
`AuroraDb_ddl_stmt_duration`	La DDL latenza aggregata dall'ultimo riavvio.
`AuroraDb_select_stmt_duration`	La latenza dell'istruzione `SELECT` aggregata dall'ultimo riavvio.
`AuroraDb_insert_stmt_duration`	La latenza dell'istruzione `INSERT` aggregata dall'ultimo riavvio.
`AuroraDb_update_stmt_duration`	La latenza dell'istruzione `UPDATE` aggregata dall'ultimo riavvio.
`AuroraDb_delete_stmt_duration`	La latenza dell'istruzione `DELETE` aggregata dall'ultimo riavvio.
`Aurora_binlog_io_cache_allocated`	Il numero di byte allocati alla cache I/O binlog.
`Aurora_binlog_io_cache_read_requests`	Il numero di richieste di lettura effettuate alla cache I/O binlog.
`Aurora_binlog_io_cache_reads`	Il numero di richieste di lettura gestite dalla cache I/O binlog.
`Aurora_enhanced_binlog`	Indica se il binlog avanzato è abilitato o disabilitato per questa istanza database. Per ulteriori

Nome	Descrizione
	informazioni, consulta Configurazione di binlog avanzato per Aurora MySQL.
`Aurora_external_connection_count`	Il numero di connessioni al database all'istanza DB, escluse le connessioni di RDS servizio utilizzate per i controlli dello stato del database.
`Aurora_fast_insert_cache_hits`	Un contatore che viene incrementato quando il cursore memorizzato nella cache viene recuperato e verificato. Per ulteriori informazioni sulla cache di inserimento rapido, consulta Miglioramenti alle prestazioni di Amazon Aurora MySQL.
`Aurora_fast_insert_cache_misses`	Un contatore che viene incrementato quando il cursore memorizzato nella cache non è più valido e Aurora esegue un attraversamento di indice normale. Per ulteriori informazioni sulla cache di inserimento rapido, consulta Miglioramenti alle prestazioni di Amazon Aurora MySQL.
`Aurora_fts_cache_memory_used`	La quantità di memoria in byte utilizzata dal sistema di ricerca full-text di InnoDB. Questa variabile si applica a Aurora My SQL versione 3.07 e successive.
`Aurora_fwd_master_dml_stmt_count`	Il numero totale di istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 2 di Aurora My.
`Aurora_fwd_master_dml_stmt_duration`	La durata totale delle istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 2 di Aurora My.

Nome	Descrizione
`Aurora_fwd_master_errors_rpc_timeout`	Il numero di volte in cui non è stato possibile stabilire una connessione inoltrata sull'istanza di scrittura.
`Aurora_fwd_master_errors_session_limit`	Il numero di query inoltrate che vengono rifiutate a causa di `session full` sull'istanza di scrittura.
`Aurora_fwd_master_errors_session_timeout`	Il numero di volte in cui una sessione di inoltro viene terminata a causa di un timeout sull'istanza di scrittura.
`Aurora_fwd_master_open_sessions`	Il numero di sessioni inoltrate sull'istanza database di scrittura. Questa variabile si applica alla SQL versione 2 di Aurora My.
`Aurora_fwd_master_select_stmt_count`	Il numero totale di istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 2 di Aurora My.
`Aurora_fwd_master_select_stmt_duration`	La durata totale delle istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 2 di Aurora My.
`Aurora_fwd_writer_dml_stmt_count`	Il numero totale di istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 3 di Aurora My.
`Aurora_fwd_writer_dml_stmt_duration`	La durata totale delle istruzioni DML inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 3 di Aurora My.

Nome	Descrizione
`Aurora_fwd_writer_errors_rpc_timeout`	Il numero di volte in cui non è stato possibile stabilire una connessione inoltrata sull'istanza di scrittura.
`Aurora_fwd_writer_errors_session_limit`	Il numero di query inoltrate che vengono rifiutate a causa di `session full` sull'istanza di scrittura.
`Aurora_fwd_writer_errors_session_timeout`	Il numero di volte in cui una sessione di inoltro viene terminata a causa di un timeout sull'istanza di scrittura.
`Aurora_fwd_writer_open_sessions`	Il numero di sessioni inoltrate sull'istanza database di scrittura. Questa variabile si applica alla SQL versione 3 di Aurora My.
`Aurora_fwd_writer_select_stmt_count`	Il numero totale di istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 3 di Aurora My.
`Aurora_fwd_writer_select_stmt_duration`	La durata totale delle istruzioni `SELECT` inoltrate a questa istanza database di scrittura. Questa variabile si applica alla SQL versione 3 di Aurora My.
`Aurora_lockmgr_buffer_pool_memory_used`	La quantità di memoria del pool di buffer in byte utilizzata da Aurora My SQL lock manager.
`Aurora_lockmgr_memory_used`	La quantità di memoria in byte utilizzata da Aurora SQL My lock manager.

Nome	Descrizione
`Aurora_ml_actual_request_cnt`	Le richieste aggregate contano tra Aurora ySQLmakes M e i servizi di machine learning Aurora per tutte le query eseguite dagli utenti dell'istanza DB. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.
`Aurora_ml_actual_response_cnt`	Il numero di risposte aggregate che Aurora SQL My riceve dai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.
`Aurora_ml_cache_hit_cnt`	Il numero aggregato di accessi alla cache interna che Aurora SQL My riceve dai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.
`Aurora_ml_logical_request_cnt`	Il numero di richieste logiche valutate dall'istanza database per essere inviate ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato. A seconda del fatto che sia stato utilizzato il batching, questo valore può essere superiore a `Aurora_ml_actual_request_cnt` . Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.

Nome	Descrizione
`Aurora_ml_logical_response_cnt`	Il numero di risposte aggregate che Aurora SQL My riceve dai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.
`Aurora_ml_retry_request_cnt`	Il numero di richieste ripetute inviate dall'ista nza database ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato. Per ulteriori informazioni, consulta Utilizzo dell'appr endimento automatico di Amazon Aurora con Aurora My SQL.
`Aurora_ml_single_request_cnt`	Il conteggio aggregato delle funzioni di machine learning di Aurora valutate in modalità non batch per tutte le query eseguite dagli utenti dell'istanza database. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL.
`aurora_oom_avoidance_recove ry_state`	Indica se Aurora out-of-memory (OOM) Evitance Recovery si trova nello `INACTIVE` stato `ACTIVE` o per questa istanza DB. Questa variabile si applica a Aurora My SQL versione 3.06.0 e successive.

Nome	Descrizione
aurora_oom_reserved_mem_enter_kb	Rappresenta la soglia per l'immissione dello RESERVED stato nel meccanismo di gestione di Aurora. OOM Quando la memoria disponibile sul server scende al di sotto di questa soglia, aurora_oom_status cambia inRESERVED, a indicare che il server si sta avvicinando a un livello critico di utilizzo della memoria. Questa variabile si applica a Aurora My SQL versione 3.06.0 e successive.
aurora_oom_reserved_mem_exit_kb	Rappresenta la soglia per uscire RESERVED dallo stato nel meccanismo di gestione di Aurora. OOM Quando la memoria disponibile sul server supera questa soglia, aurora_oom_status torna allo stato attualeNORMAL, indicando che il server è tornato a uno stato più stabile con risorse di memoria sufficienti. Questa variabile si applica a Aurora My SQL versione 3.06.0 e successive.

Nome	Descrizione
`aurora_oom_status`	Rappresenta lo OOM stato corrente di questa istanza DB. Quando il valore è`NORMAL`, indica che ci sono risorse di memoria sufficienti. Se il valore cambia in`RESERVED`, indica che la memoria disponibile sul server è insuffici ente. Le azioni vengono intraprese in base alla configurazione `aurora_oom_response` dei parametri. Per ulteriori informazioni, consulta Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL. Questa variabile si applica a Aurora My SQL versione 3.06.0 e successive.
`Aurora_pq_bytes_returned`	Il numero di byte per le strutture di dati tuple trasmesse al nodo head nel corso delle query in parallelo. Dividi questo valore per 16.384 per compararlo a `Aurora_pq_pages_pu shed_down` .
`Aurora_pq_max_concurrent_re quests`	Il numero massimo di sessioni di query in parallelo eseguibili simultaneamente su questa istanza database Aurora. Si tratta di un numero fisso che dipende dalla classe dell'istanza AWS DB.
`Aurora_pq_pages_pushed_down`	Il numero di pagine di dati (ognuna con una dimensione fissa di 16 KiB) per le quali una query in parallelo ha evitato una trasmissione di rete al nodo head.

Nome	Descrizione
`Aurora_pq_request_attempted`	Il numero di sessioni di query in parallelo necessarie. Questo valore può rappresentare più di una sessione per query, a seconda di SQL costrutti come sottoquery e join.
`Aurora_pq_request_executed`	Il numero di sessioni di query in parallelo riuscite.
`Aurora_pq_request_failed`	Il numero di sessioni di query in parallelo che hanno restituito un errore al client. In alcuni casi, una richiesta di query in parallelo può non riuscire, ad esempio a causa di un problema nel livello di storage. In questi casi, viene eseguito un nuovo tentativo per la parte di query non riuscita utilizzando il meccanismo di query non in parallelo. Se anche il nuovo tentativo non riesce, viene restituito un errore al client e questo contatore viene incrementato.
`Aurora_pq_request_in_progress`	Il numero di sessioni di query in parallelo attualmente in corso. Questo numero si applica all'istanza database Aurora a cui sei connesso e non all'intero cluster di database Aurora. Per determinare se un'istanza database è prossima al relativo limite di simultaneità, compara questo valore a `Aurora_pq_max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	Il numero di volte che una query in parallelo non è stata scelta per una query. Questo valore è la somma di vari altri contatori più granulari . Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.

Nome	Descrizione
`Aurora_pq_request_not_chosen_below_min_rows`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero di righe nella tabella. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_column_bit`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele a causa di un tipo di dati non supportato nell'elenco delle colonne proiettate.
`Aurora_pq_request_not_chosen_column_geometry`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con il tipo di `GEOMETRY` dati. Per informazioni sulle SQL versioni di Aurora My che rimuovono questa limitazione, consulta. Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL
`Aurora_pq_request_not_chosen_column_lob`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene colonne con un tipo di dati `LOB` o colonne `VARCHAR` memorizzate esternamente a causa della lunghezza dichiarata. Per informazioni sulle SQL versioni di Aurora My che rimuovono questa limitazione, consulta. Aggiornamento dei cluster di query parallele ad Aurora My versione 3 SQL

Nome	Descrizione
`Aurora_pq_request_not_chosen_column_virtual`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella contiene una colonna virtuale.
`Aurora_pq_request_not_chosen_custom_charset`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di colonne con un set di caratteri personalizzato.
`Aurora_pq_request_not_chosen_fast_ddl`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella viene attualmente modificata da un'istruzione fast DDL`ALTER`.
`Aurora_pq_request_not_chosen_ few_pages_outside_buffer_pool`	Il numero di volte che una query in parallelo non è stata scelta anche se meno del 95% dei dati della tabella era già nel pool di buffer, in quanto non c'erano dati non memorizzati nel buffer sufficienti per giustificare l'utilizzo della funzione di query in parallelo.
`Aurora_pq_request_not_chosen_full_text_index`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella dispone di indici full-text.

Nome	Descrizione
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	Il numero di volte che una query in parallelo non è stata scelta in quanto un'elevata percentuale di dati di tabella (attualmente maggiore del 95%) era già nel pool di buffer. In questi casi, l'ottimizzatore determina che la lettura dei dati del pool di buffer è più efficace. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_index_hint`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query include un hint di indice.
`Aurora_pq_request_not_chosen_innodb_table_format`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la tabella utilizza un formato di riga InnoDB non supportato. La query parallela Aurora si applica solo ai formati di riga `COMPACT`, `REDUNDANT` e `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	Il numero di richieste di query in parallelo che hanno utilizzato il percorso di elaborazione di query non in parallelo in seguito all'avvio della query in una transazione di lunga durata. Un'istruzione `EXPLAIN` può incrementare questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_no_where_clause`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query non include alcuna clausola `WHERE`.

Nome	Descrizione
`Aurora_pq_request_not_chosen_range_scan`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza una scansione di intervallo su un indice.
`Aurora_pq_request_not_chosen_row_length_too_long`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la lunghezza totale combinata di tutte le colonne è troppo lunga.
`Aurora_pq_request_not_chosen_small_table`	Il numero di volte che una query in parallelo non è stata scelta a causa della dimensione globale della tabella, come determinata dal numero di righe e dalla lunghezza media delle stesse. Un'istruzione `EXPLAIN` può increment are questo contatore anche se la query non viene effettivamente eseguita.
`Aurora_pq_request_not_chosen_temporary_table`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa riferimen to a tabelle temporanee che utilizzano i tipi di tabella `MyISAM` o `memory` non supportati.
`Aurora_pq_request_not_chosen_tx_isolation`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query utilizza un livello di isolamento delle transazioni non supportato. Sulle istanze del database di lettura, la query parallela si applica solo ai livelli di isolamento `REPEATABLE READ` e `READ COMMITTED` .

Nome	Descrizione
`Aurora_pq_request_not_chosen_update_delete_stmts`	Numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallele perché la query fa parte di un'istruzione `UPDATE` o `DELETE`.
`Aurora_pq_request_not_chosen_unsupported_access`	Il numero di richieste di query in parallelo che utilizzano il percorso di elaborazione di query non in parallelo in quanto la clausola `WHERE` non soddisfa i criteri delle query in parallelo . Ciò può avvenire se la query non richiede l'analisi di un grande volume di dati oppure se la query è un'istruzione `DELETE` o `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	Il numero di richieste di query parallele che utilizzano il percorso di elaborazione delle query non parallelo perché il cluster Aurora SQL My DB non utilizza una configurazione di archiviazione del cluster Aurora supportata. Per ulteriori informazioni, consulta Limitazioni. Questo parametro si applica a Aurora My SQL versione 3.04 e successive.
`Aurora_pq_request_throttled`	Il numero di volte che una query in parallelo non è stata scelta a causa del numero massimo di query in parallelo simultanee già in esecuzione su una particolare istanza database Aurora.
`Aurora_repl_bytes_received`	Numero di byte replicati su un'istanza del database Aurora My SQL reader dall'ultimo riavvio. Per ulteriori informazioni, consulta Replica con Amazon Aurora MySQL.

Nome	Descrizione
`Aurora_reserved_mem_exceeded_incidents`	Il numero di volte dall'ultimo riavvio in cui il motore ha superato i limiti di memoria prenotata . Se `aurora_oom_response` è configurata, questa soglia definisce quando out-of-memory (OOM) vengono attivate le attività di evitamento. Per ulteriori informazioni sulla SQL OOM risposta Aurora My, vedere. Risoluzione dei out-of-memory problemi relativi ai database Aurora MySQL
`aurora_temptable_max_ram_allocation`	La quantità massima di memoria, in byte, utilizzata in qualsiasi momento dalle tabelle temporanee interne dall'ultimo riavvio.
`aurora_temptable_ram_allocation`	La quantità corrente di memoria, in byte, utilizzata dalle tabelle temporanee interne.
`Aurora_thread_pool_thread_count`	Il numero attuale di thread nel pool di thread di Aurora. Per ulteriori informazioni sul pool di thread in Aurora MySQL, vedere. Pool di thread
`Aurora_tmz_version`	Indica la versione corrente delle informazioni sul fuso orario utilizzate dal cluster database. I valori seguono il formato Internet Assigned Numbers Authority (IANA):`YYYYsuffix` , ad esempio `2022a` e`2023c`. Questo parametro si applica alla SQL versione 2.12 e successive di Aurora My e alla versione 3.04 e successive.
`Aurora_zdr_oom_threshold`	Rappresenta la soglia di memoria, in kilobyte (KB), per un'istanza Aurora DB per avviare un riavvio senza downtime (ZDR) per il ripristino da potenziali problemi relativi alla memoria.

Nome	Descrizione
server_aurora_das_running	Indica se i flussi di attività del database (DAS) sono abilitati o disabilitati su questa istanza DB. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

Le mie variabili di SQL stato che non si applicano ad Aurora My SQL

A causa delle differenze architettoniche tra Aurora My SQL e MySQL, alcune variabili di SQL stato My non si applicano ad Aurora My. SQL

Le seguenti variabili di SQL stato non si applicano ad Aurora My. SQL L'elenco non è completo.

- innodb_buffer_pool_bytes_dirty
- innodb_buffer_pool_pages_dirty
- innodb_buffer_pool_pages_flushed

Aurora My SQL versione 3 rimuove le seguenti variabili di stato presenti in Aurora My versione 2: SQL

- AuroraDb_lockmgr_bitmaps0_in_use
- AuroraDb_lockmgr_bitmaps1_in_use
- AuroraDb_lockmgr_bitmaps_mem_used
- AuroraDb_thread_deadlocks
- available_alter_table_log_entries
- Aurora_lockmgr_memory_used
- Aurora_missing_history_on_replica_incidents
- Aurora_new_lock_manager_lock_release_cnt
- Aurora_new_lock_manager_lock_release_total_duration_micro
- Aurora_new_lock_manager_lock_timeout_cnt
- Aurora_total_op_memory

- Aurora_total_op_temp_space
- Aurora_used_alter_table_log_entries
- Aurora_using_new_lock_manager
- Aurora_volume_bytes_allocated
- Aurora_volume_bytes_left_extent
- Aurora_volume_bytes_left_total
- Com_alter_db_upgrade
- Compression
- External_threads_connected
- Innodb_available_undo_logs
- Last_query_cost
- Last_query_partial_plans
- Slave_heartbeat_period
- Slave_last_heartbeat
- Slave_received_heartbeats
- Slave_retried_transactions
- Slave_running
- Time_since_zero_connections

Queste variabili My SQL status sono disponibili in Aurora My SQL versione 2, ma non sono disponibili in Aurora My versione 3: SQL

- Innodb_redo_log_enabled
- Innodb_undo_tablespaces_total
- Innodb_undo_tablespaces_implicit
- Innodb_undo_tablespaces_explicit
- Innodb_undo_tablespaces_active

Eventi Aurora My wait SQL

Di seguito sono riportati alcuni eventi di attesa comuni per Aurora My. SQL

> **Note**
>
> Per informazioni sull'ottimizzazione di Aurora SQL My performance utilizzando gli eventi di attesa, consulta. Regolazione di Aurora MySQL con eventi di attesa
> Per informazioni sulle convenzioni di denominazione utilizzate negli eventi My SQL wait, vedi Convenzioni di denominazione degli strumenti Performance Schema nella documentazione My. SQL

cpu

Il numero di connessioni attive pronte per l'esecuzione è costantemente superiore al numero di. vCPUs Per ulteriori informazioni, consulta cpu.

io/aurora_redo_log_flush

Una sessione sta mantenendo i dati permanenti nell'archiviazione Aurora. In genere, questo evento di attesa riguarda un'operazione di I/O di scrittura in SQL Aurora My. Per ulteriori informazioni, consulta io/aurora_redo_log_flush.

io/aurora_respond_to_client

L'elaborazione delle query è stata completata e i risultati vengono restituiti al client dell'applicazione per le seguenti versioni di Aurora My: 2.10.2 e SQL versioni 2.10 successive, 2.09.3 e versioni 2.09 successive e 2.07.7 e versioni 2.07 successive. Confrontare la larghezza di banda di rete della classe di istanza database con la dimensione del set di risultati restituito. Inoltre, controlla i tempi di risposta lato client. Se il client non risponde e non è in grado di elaborare i pacchetti, possono verificarsi perdite di pacchetti e ritrasmissioni. TCP TCP Questa situazione influisce negativamente sulla larghezza di banda della rete. Nelle versioni inferiori a 2.10.2, 2.09.3 e 2.07.7, l'evento di attesa include erroneamente il tempo di inattività. Per informazioni su come ottimizzare il database quando questa attesa è importante, vedere io/aurora_respond_to_client.

io/file/csv/data

I thread scrivono su tabelle nel formato value () separato da virgole. CSV Controlla l'utilizzo della tua tabella. CSV Una causa tipica di questo evento è l'impostazione `log_output` in una tabella.

io/file/sql/binlog

Un thread è in attesa di un file di log binario (binlog) scritto su disco.

io/redo_log_flush

Una sessione sta mantenendo i dati permanenti nell'archiviazione Aurora. In genere, questo evento di attesa riguarda un'operazione di I/O di scrittura in SQL Aurora My. Per ulteriori informazioni, consulta io/redo_log_flush.

io/socket/sql/client_connection

Il programma `mysqld` è impegnato a creare thread per gestire le nuove connessioni client in arrivo. Per ulteriori informazioni, consulta io/socket/sql/client_connessione.

io/table/sql/handler

Il motore è in attesa di accedere a una tabella. Questo evento si verifica indipendentemente dal fatto che i dati siano memorizzati nella cache nel buffer pool o se si accede su disco. Per ulteriori informazioni, consulta io/table/sql/handler.

lock/table/sql/handler

Questo evento di attesa è un gestore eventi di attesa di blocco di tabella. Per ulteriori informazioni sugli eventi relativi agli atomi e alle molecole nello schema delle prestazioni, vedete gli eventi relativi agli atomi e alle molecole dello schema delle prestazioni nella documentazione My. SQL

synch/cond/innodb/row_lock_wait

Più istruzioni Data Manipulation Language (DML) accedono alle stesse righe del database contemporaneamente. Per ulteriori informazioni, consulta synch/cond/innodb/row_blocca_aspetta.

synch/cond/innodb/row_lock_wait_cond

Più DML istruzioni accedono alle stesse righe del database contemporaneamente. Per ulteriori informazioni, consulta synch/cond/innodb/row_lock_wait_cond.

MDLsynch/cond/sql/ _context:: _wait_status COND

I thread sono in attesa di un blocco dei metadati di tabella. Il motore utilizza questo tipo di blocco per gestire l'accesso simultaneo a uno schema di database e garantire la coerenza dei dati. Per ulteriori informazioni, vedere Ottimizzazione delle operazioni di blocco nella documentazione personale. SQL Per informazioni su come ottimizzare il database quando questo evento è importante, vedere synch/cond/sql/MDL_contesto:: COND _wait_status.

synch/cond/sql/ _ _MYSQL:: _done BIN LOG COND

Hai attivato la registrazione binaria. Potrebbe esserci un' elevata velocità effettiva di esecuzione del commit, un numero elevato di transazioni o repliche che leggono i binlog. Prendi in considerazione l'utilizzo di istruzioni multiriga o di crezione di bundle di istruzioni in un'unica

transazione. In Aurora, usa i database globali anziché la replica del log binario oppure usa i parametri `aurora_binlog_*`.

synch/mutex/innodb/aurora_lock_thread_slot_futex

Più DML istruzioni accedono alle stesse righe del database contemporaneamente. Per ulteriori informazioni, consulta synch/mutex/innodb/aurora_lock_thread_slot_futex.

synch/mutex/innodb/buf_pool_mutex

Il pool di buffer non è abbastanza grande da contenere il set di dati funzionante. Oppure il carico di lavoro accede alle pagine da una tabella specifica, che porta a contendenze nel buffer pool. Per ulteriori informazioni, consulta synch/mutex/innodb/buf_pool_mutex.

synch/mutex/innodb/fil_system_mutex

Il processo è in attesa di accedere alla cache di memoria tablespace. Per ulteriori informazioni, consulta synch/mutex/innodb/fil_sistema_mutex.

synch/mutex/innodb/trx_sys_mutex

Le operazioni consistono nel controllare, aggiornare, eliminare o aggiungere transazioni IDs in InnoDB in modo coerente o controllato. Queste operazioni richiedono una chiamata mutex `trx_sys`, che viene tracciata dalla strumentazione Performance Schema. Le operazioni includono la gestione del sistema di transazioni all'avvio o all'arresto del database, i rollback, le operazioni di cancellazione degli annullamenti, l'accesso in lettura delle righe e i carichi del pool di buffer. L'elevato carico del database con un numero elevato di transazioni comporta la frequente comparsa di questo evento di attesa. Per ulteriori informazioni, consulta synch/mutex/innodb/trx_sys_mutex.

KEYsynch/mutex/mysys/ _: :cache_lock CACHE

Il mutex controlla l'accesso alla cache delle chiavi per My tables. `keycache->cache_lock` ISAM Sebbene Aurora My SQL non consenta l'utilizzo di Le mie ISAM tabelle per archiviare dati persistenti, vengono utilizzate per archiviare tabelle temporanee interne. Valuta se controllare i conteggi dello stato `created_tmp_disk_tables` o `created_tmp_tables`, perché in determinate situazioni, le tabelle temporanee vengono scritte su disco quando non possono più essere contenute nella memoria.

FILEsynch/mutex/sql/ _AS_:: _offsets TABLE LOCK

Il motore acquisisce questo mutex durante l'apertura o la creazione di un file di metadati di tabella. Quando questo evento di attesa si verifica con frequenza eccessiva, il numero di tabelle create o aperte è aumentato.

synch/mutex/sql/ FILE _AS_:: TABLE _shim_lists LOCK

Il motore acquisisce questo mutex durante l'esecuzione di operazioni come `reset_size`,`detach_contents`, oppure `add_contents` sulla struttura interna che tiene traccia delle tabelle aperte. Il mutex sincronizza l'accesso al contenuto dell'elenco. Quando questo evento di attesa si verifica con alta frequenza, indica un cambiamento improvviso nel set di tabelle a cui si accedeva in precedenza. Il motore deve accedere a nuove tabelle o lasciare andare il contesto relativo alle tabelle precedentemente accessibili.

synch/mutex/sql/ LOCK _apri

Il numero di tabelle aperte dalle sessioni supera la dimensione della cache delle definizioni di tabella o della cache aperta della tabella. Aumenta le dimensioni di queste cache. Per ulteriori informazioni, vedi How My opens and closes tables. SQL

synch/mutex/sql/ _table_cache LOCK

Il numero di tabelle aperte dalle sessioni supera la dimensione della cache delle definizioni di tabella o della cache aperta della tabella. Aumenta le dimensioni di queste cache. Per ulteriori informazioni, vedi How My opens and close tables. SQL

synch/mutex/sql/ LOG

In questo evento di attesa, ci sono thread in attesa di un blocco di log. Ad esempio, un thread potrebbe attendere un blocco per scrivere nel file di log delle query con tempi di risposta molto lunghi.

sincronizzazione/mutex/sql/ MYSQL _ _:: _commit BIN LOG LOCK

In questo evento di attesa, c'è un thread in attesa di acquisire un blocco per eseguire il commit nel log binario. Può verificarsi un conflitto della registrazione binaria nei database con frequenza di modifica molto alta. A seconda della versione di My in usoSQL, vengono utilizzati alcuni blocchi per proteggere la consistenza e la durabilità del registro binario. In RDS for MySQL, i log binari vengono utilizzati per la replica e il processo di backup automatizzato. In Aurora MySQL, i log binari non sono necessari per la replica o i backup nativi. Sono disabilitati per impostazione predefinita, ma possono essere abilitati e usati per la replica esterna o l'acquisizione dei dati delle modifiche. Per ulteriori informazioni, consulta The binary log nella documentazione My. SQL

MYSQLsync/mutex/sql/ _ _:: _dump_thread_metrics_collection BIN LOG LOCK

Se la registrazione binaria è attivata, il motore acquisisce questo mutex quando stampa le metriche dei thread di dump attivi nel registro degli errori del motore e sulla mappa delle operazioni interne.

sync/mutex/sql/ MYSQL _ BIN _LOG:: LOCK _inactive_binlogs_map

Se la registrazione binaria è attivata, il motore acquisisce questo mutex quando aggiunge, elimina o cerca l'elenco dei file binlog dietro l'ultimo.

sync/mutex/sql/ MYSQL _ BIN _LOG:: LOCK _io_cache

Se la registrazione binaria è attivata, il motore acquisisce questo mutex durante le operazioni della cache I/O di Aurora binlog: allocazione, ridimensionamento, liberazione, scrittura, lettura, rimozione e accesso alle informazioni della cache. Se questo evento si verifica frequentemente, il motore accede alla cache in cui sono memorizzati gli eventi binlog. Per ridurre i tempi di attesa, riduci i commit. Prova a raggruppare più istruzioni in una singola transazione.

sincronizzazione/mutex/sql/ MYSQL _ BIN _LOG:: LOCK _log

Hai attivato la registrazione binaria. Potrebbe esserci un'elevata velocità effettiva di esecuzione del commit, molte transazioni nell'esecuzione del commit o repliche che leggono i binlog. Prendi in considerazione l'utilizzo di istruzioni multiriga o di crezione di bundle di istruzioni in un'unica transazione. In Aurora, usa i database globali anziché la replica dei log binari o usa i parametri `aurora_binlog_*`.

sincronizzazione/mutex/sql/ SERVER THREAD _:LOCK: _sync

Il `SERVER_THREAD::LOCK_sync` mutex viene acquisito durante la pianificazione, l'elaborazione o l'avvio di thread per la scrittura di file. L'eccessiva occorrenza di questo evento di attesa indica una maggiore attività di scrittura nel database.

synch/mutex/sql/:lock TABLESPACES

Il motore acquisisce il mutex `TABLESPACES:lock` durante le seguenti operazioni di tablespace: creare, eliminare, troncare ed estendere. L'eccessiva occorrenza di questo evento di attesa indica un'alta frequenza di operazioni di tablespace. Un esempio è il caricamento di una grande quantità di dati nel database.

synch/rwlock/innodb/dict

In questo evento di attesa, ci sono thread in attesa di un oggetto rwlock nel dizionario dei dati InnoDB.

synch/rwlock/innodb/dict_operation_lock

In questo evento di attesa, ci sono thread che mantengono blocchi sulle operazioni del dizionario dei dati InnoDB.

synch/rwlock/innodb/dict sys RW lock

Un numero elevato di istruzioni simultanee del linguaggio di controllo dei dati (DCLs) nel codice del linguaggio di definizione dei dati () vengono attivate contemporaneamente. DDLs Riduci la dipendenza dell'applicazione DDLs durante la normale attività dell'applicazione.

synch/rwlock/innodb/index_tree_rw_lock

Un gran numero di istruzioni simili del linguaggio di manipolazione dei dati (DML) accedono allo stesso oggetto di database contemporaneamente. Prova a usare istruzioni multiriga. Inoltre, distribuisci il carico di lavoro su diversi oggetti di database. Ad esempio, implementa il partizionamento.

synch/rwlock/innodb/dict_operation_lock

Un numero elevato di istruzioni simultanee del linguaggio di controllo dei dati (DCLs) nel linguaggio di definizione dei dati code (DDLs) viene attivato contemporaneamente. Riduci la dipendenza dell'applicazione DDLs durante la normale attività dell'applicazione.

synch/sxlock/innodb/dict_sys_lock

Un numero elevato di istruzioni simultanee del linguaggio di controllo dei dati (DCLs) nel codice del linguaggio di definizione dei dati (DDLs) viene attivato contemporaneamente. Riduci la dipendenza dell'applicazione DDLs durante la normale attività dell'applicazione.

synch/sxlock/innodb/hash_table_locks

La sessione non è in grado di trovare pagine nel pool di buffer. Il motore deve leggere un file o modificare l'elenco usato meno di recente (LRU) per il buffer pool. Considera di aumentare le dimensioni della cache del buffer e migliorare i percorsi di accesso per le query pertinenti.

synch/sxlock/innodb/index_tree_rw_lock

Molte istruzioni simili del linguaggio di manipolazione dei dati (DML) accedono allo stesso oggetto di database contemporaneamente. Prova a usare istruzioni multiriga. Inoltre, distribuisci il carico di lavoro su diversi oggetti di database. Ad esempio, implementa il partizionamento.

Per ulteriori informazioni sulla risoluzione dei problemi di attesa di sincronizzazione, vedi Perché la mia istanza My SQL DB mostra un numero elevato di sessioni attive in attesa di eventi di SYNCH attesa in Performance Insights? .

Aurora Il mio thread afferma SQL

Di seguito sono riportati alcuni stati del thread comune per Aurora My. SQL

Verifica delle autorizzazioni

Il thread sta verificando se il server dispone dei privilegi necessari per eseguire l'istruzione.

controllo della cache delle query per la query

Il server sta verificando se la query corrente è presente nella cache delle query.

ripulito

Questo è lo stato finale di una connessione il cui lavoro è completo ma che non è stato chiuso dal client. La soluzione migliore è chiudere esplicitamente la connessione nel codice. Oppure puoi impostare un valore inferiore per `wait_timeout` nel gruppo di parametri.

chiusura tabelle

Il thread sta scaricando i dati della tabella modificati su disco e chiude le tabelle utilizzate. Se non si tratta di un'operazione rapida, verificare le metriche di consumo della larghezza di banda di rete rispetto alla larghezza di banda di rete della classe di istanza. Inoltre, verificare che i valori dei parametri per `table_open_cachee` `table_definition_cache` il parametro consentano di aprire contemporaneamente una quantità sufficiente di tabelle in modo che il motore non debba aprire e chiudere frequentemente le tabelle. Questi parametri influenzano il consumo di memoria sull'istanza.

conversione in My HEAP ISAM

La query sta convertendo una tabella temporanea da memoria a su disco. Questa conversione è necessaria perché le tabelle temporanee create da My SQL nelle fasi intermedie dell'elaborazione delle query sono diventate troppo grandi per la memoria. Verifica i valori di `tmp_table_size` e `max_heap_table_size`. Nelle versioni successive, il nome dello stato del thread è `converting HEAP to ondisk`.

conversione HEAP su disco

Il thread sta convertendo una tabella temporanea interna da una tabella in memoria a una tabella su disco.

copia nella tabella tmp

Il thread sta elaborando un'istruzione `ALTER TABLE`. Questo stato si verifica dopo che la tabella con la nuova struttura è stata creata ma prima che le righe vengano copiate. Per un thread in questo stato, è possibile utilizzare lo schema delle prestazioni per ottenere informazioni sullo stato di avanzamento dell'operazione di copia.

creazione di indice di ordinamento

Aurora My esegue SQL un ordinamento perché non può utilizzare un indice esistente per soddisfare la `GROUP BY` clausola `ORDER BY` or di una query. Per ulteriori informazioni, consulta creazione di indice di ordinamento.

Creazione di tabelle

Il thread sta creando una tabella permanente o temporanea.

commit ritardato ok fatto

Un commit asincrono in Aurora My SQL ha ricevuto un riconoscimento ed è completo.

ritardato commit ok avviato

Il SQL thread Aurora My ha avviato il processo di commit asincrono ma è in attesa di conferma. Di solito questo è il tempo di commit autentico di una transazione.

invio ritardato ok fatto

Un SQL thread di lavoro Aurora My collegato a una connessione può essere liberato mentre viene inviata una risposta al client. Il thread può iniziare altri lavori. Lo stato `delayed send ok` significa che la conferma asincrona al client è stata completata.

invio ritardato ok avviato

Un SQL thread di lavoro Aurora My ha inviato una risposta in modo asincrono a un client ed è ora libero di lavorare per altre connessioni. La transazione ha avviato un processo di commit asincrono che non è ancora stato riconosciuto.

executing

Il thread ha iniziato a eseguire un'istruzione.

liberazione elementi

Il thread ha eseguito un comando. Alcuni liberazioni degli elementi eseguiti durante questo stato coinvolgono la cache delle query. Questo stato è solitamente seguito dalla pulizia.

init

Questo stato si verifica prima dell'inizializzazione di `ALTER TABLE`,`DELETE`,`INSERT`,`SELECT`, oppure istruzioni `UPDATE`. Le azioni in questo stato includono lo svuotamento del log binario o del log InnoDB e alcune operazioni di pulizia della cache delle query.

Source ha inviato tutti i binlog alla replica; in attesa di ulteriori aggiornamenti

Il nodo primario ha terminato la sua parte della replica. Il thread è in attesa di eseguire altre query in modo che possa scrivere nel log binario (binlog).

apertura tabella

Il thread sta tentando di aprire una tabella. Questa operazione è veloce a meno che un'istruzione `ALTER TABLE` o `LOCK TABLE` debba finire o superare il valore di `table_open_cache`.

Ottimizzazione

Il server sta eseguendo le ottimizzazioni iniziali per una query.

preparazione

Questo stato si verifica durante l'ottimizzazione delle query.

fine query

Questo stato si verifica dopo l'elaborazione di una query ma prima dello stato di liberazione degli elementi.

rimozione di duplicati

Aurora My non è SQL riuscita a ottimizzare un'`DISTINCT`operazione nella fase iniziale di una query. Aurora My SQL deve rimuovere tutte le righe duplicate prima di inviare il risultato al client.

ricerca di righe per l'aggiornamento

Il thread sta trovando tutte le righe corrispondenti prima di aggiornarle. Questa fase è necessaria se il `UPDATE` sta cambiando l'indice utilizzato dal motore per trovare le righe.

invio dell'evento binlog a slave

Il thread ha letto un evento dal log binario e lo sta inviando alla replica.

invio di risultati memorizzati nella cache al client

Il server sta prendendo il risultato di una query dalla cache delle query e la invia al client.

Invio dei dati

Il thread sta leggendo ed elaborando le righe per un'istruzione `SELECT` ma non ha ancora iniziato a inviare dati al client. Il processo sta identificando quali pagine contengono i risultati necessari per soddisfare la query. Per ulteriori informazioni, consulta invio dei dati.

invio al client

Il server sta scrivendo un pacchetto sul client. Nelle SQL versioni precedenti di My, questo evento di attesa era etichettato. `writing to net`

avvio in corso

Questa è la prima fase all'inizio dell'esecuzione dell'istruzione.

statistiche

Il server sta calcolando le statistiche per sviluppare un piano di esecuzione delle query. Se un thread è in questo stato per un lungo periodo, il server è probabilmente associato a disco durante l'esecuzione di altri lavori.

archiviazione dei risultati nella cache delle query

Il server sta memorizzando il risultato di una query nella cache delle query.

blocco del sistema

Il thread ha chiamato `mysql_lock_tables`, ma lo stato del thread non è stato aggiornato dalla chiamata. Questo stato generale si verifica per molte ragioni.

aggiorna

Il thread si sta preparando per iniziare ad aggiornare la tabella.

aggiornamento

Il thread sta cercando righe e le sta aggiornando.

blocco utente

Il thread ha emesso una chiamata `GET_LOCK`. Il thread ha richiesto un blocco consultivo e lo sta aspettando o sta pianificando di richiederlo.

in attesa di ulteriori aggiornamenti

Il nodo primario ha terminato la sua parte della replica. Il thread è in attesa di eseguire altre query in modo che possa scrivere nel log binario (binlog).

in attesa del blocco dei metadati dello schema

Questa è un'attesa per un blocco dei metadati.

in attesa del blocco dei metadati della funzione memorizzata

Questa è un'attesa per un blocco dei metadati.

in attesa del blocco dei metadati della procedura archiviata

Questa è un'attesa per un blocco dei metadati.

in attesa dello scarico della tabella

Il thread sta eseguendo `FLUSH TABLES` e sta aspettando che tutti i thread chiudano le loro tabelle. Oppure il thread ha ricevuto una notifica che la struttura sottostante per una tabella è cambiata, quindi deve riaprire la tabella per ottenere la nuova struttura. Per riaprire la tabella, il thread deve attendere che tutti gli altri thread abbiano chiuso la tabella. Questa notifica avviene se un altro thread ha utilizzato una delle seguenti istruzioni sulla tabella: `FLUSH TABLES`,`ALTER TABLE`,`RENAME TABLE`,`REPAIR TABLE`,`ANALYZE TABLE`, oppure `OPTIMIZE TABLE`.

in attesa di blocco a livello tabella

Una sessione tiene un blocco su un tavolo mentre un'altra sessione tenta di acquisire lo stesso blocco sulla stessa tabella.

in attesa del blocco dei metadati della tabella

Aurora My SQL utilizza il blocco dei metadati per gestire l'accesso simultaneo agli oggetti del database e garantire la coerenza dei dati. In questo evento di attesa, una sessione tiene un blocco di metadati su una tabella mentre un'altra sessione tenta di acquisire lo stesso blocco sulla stessa tabella. Quando lo schema delle prestazioni è abilitato, questo stato del thread viene segnalato come evento di attesa `synch/cond/sql/MDL_context::COND_wait_status`.

scrittura su rete

Il server sta scrivendo un pacchetto sulla rete. Nelle SQL versioni successive di My, questo evento di attesa è etichettato. `Sending to client`

Livelli di isolamento di Aurora MySQL

Scopri come le istanze database in un cluster Aurora MySQL implementano la proprietà di isolamento del database. In questo argomento viene spiegato come il comportamento predefinito di Aurora MySQL si bilancia tra coerenza rigorosa e prestazioni elevate. Puoi utilizzare queste informazioni per decidere quando modificare le impostazioni predefinite in base alle caratteristiche del tuo carico di lavoro.

Livelli di isolamento disponibili per le istanze di scrittura

È possibile utilizzare i livelli di isolamento `REPEATABLE READ`, `READ COMMITTED`, `READ UNCOMMITTED` e `SERIALIZABLE` sull'istanza primaria di un cluster di database Aurora MySQL. Questi livelli di isolamento funzionano allo stesso modo in Aurora MySQL e in RDS for MySQL.

Livello di isolamento REPEATABLE READ per le istanze di lettura

Per impostazione predefinita, le istanze database Aurora MySQL configurate come repliche di Aurora di sola lettura utilizzano sempre il livello di isolamento `REPEATABLE READ`. Queste istanze database ignorano qualsiasi istruzione `SET TRANSACTION ISOLATION LEVEL` e continuano a usare il livello di isolamento `REPEATABLE READ`.

Livello di isolamento READ COMMITTED per le istanze di lettura

Se l'applicazione include un carico di lavoro ad alta intensità di scrittura nell'istanza primaria e query a esecuzione prolungata nelle repliche di Aurora, è possibile che si verifichi un notevole ritardo di eliminazione. Un ritardo di eliminazione si verifica quando la garbage collection interna è bloccata da query a esecuzione prolungata. Il sintomo è un valore elevato per `history list length` nell'output del comando `SHOW ENGINE INNODB STATUS`. È possibile monitorare questo valore utilizzando il parametro `RollbackSegmentHistoryListLength` in CloudWatch. Un notevole ritardo dell'eliminazione può ridurre l'efficacia degli indici secondari, diminuire le prestazioni complessive delle query e usare inutilmente lo spazio di archiviazione.

Se si verificano questi problemi, puoi usare l'impostazione di configurazione a livello di sessione Aurora MySQL, `aurora_read_replica_read_committed`, per utilizzare il livello di isolamento `READ COMMITTED` nelle repliche di Aurora. L'applicazione di questa impostazione può aiutare a ridurre i rallentamenti e lo spazio usato inutilmente che possono derivare dall'esecuzione di query a esecuzione prolungata contemporaneamente alle transazioni che modificano le tabelle.

Prima di usare questa impostazione, ti consigliamo di comprendere il funzionamento di Aurora MySQL specifico dell'isolamento `READ COMMITTED`. Il comportamento `READ COMMITTED` della replica di Aurora è conforme allo standard ANSI SQL. Tuttavia, l'isolamento è meno restrittivo del comportamento `READ COMMITTED` tipico di MySQL che potresti già conoscere. Pertanto, potresti vedere risultati diversi per la query in `READ COMMITTED` su una replica di lettura di Aurora MySQL rispetto alla stessa query in `READ COMMITTED` nell'istanza primaria Aurora MySQL o in RDS per MySQL. Potresti utilizzare l'impostazione `aurora_read_replica_read_committed` per questi casi d'uso, come un report completo che esegue la scansione di un database molto grande. Potresti

invece evitarla per brevi query con piccoli set di risultati, dove la precisione e la ripetibilità sono importanti.

Il livello di isolamento `READ COMMITTED` non è disponibile per le sessioni all'interno di un cluster secondario in un Aurora Global Database che utilizzano la funzionalità di inoltro di scrittura. Per informazioni sull'inoltro di scrittura, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Utilizzo di READ COMMITTED per le letture

Per usare il livello di isolamento `READ COMMITTED` per le repliche di Aurora, configura l'impostazione `aurora_read_replica_read_committed` su `ON`. Usa questa impostazione a livello di sessione mentre è connessa a una replica di Aurora specifica. Per farlo, esegui i comandi SQL seguenti.

```
set session aurora_read_replica_read_committed = ON;
set session transaction isolation level read committed;
```

Puoi usare temporaneamente questa impostazione di configurazione per eseguire query interattive una tantum. Potresti anche voler eseguire un'applicazione per la creazione di report o l'analisi dei dati che beneficia del livello di isolamento `READ COMMITTED`, lasciando invariata l'impostazione predefinita per altre applicazioni.

Quando l'impostazione `aurora_read_replica_read_committed` è attivata, usa il comando `SET TRANSACTION ISOLATION LEVEL` per specificare il livello di isolamento per le transazioni appropriate.

```
set transaction isolation level read committed;
```

Differenze nel comportamento di READ COMMITTED nelle repliche di Aurora

L'impostazione `aurora_read_replica_read_committed` rende il livello di isolamento `READ COMMITTED` disponibile per una replica di Aurora con un comportamento coerente che viene ottimizzato per le transazioni a esecuzione prolungata. Il livello di isolamento `READ COMMITTED` nelle repliche di Aurora ha un isolamento meno restrittivo di quello nelle istanze primarie di Aurora. Per questo motivo, abilita l'impostazione solo in repliche di Aurora in cui sai che le query possono accettare la possibilità di determinati tipi di risultati incoerenti.

Le tue query possono riscontrare alcuni tipi di anomalie di lettura quando l'impostazione `aurora_read_replica_read_committed` è attivata. Due tipi di anomalie sono particolarmente importanti per comprendere e gestire il codice dell'applicazione. Una lettura non ripetibile si verifica

quando viene eseguita un'altra transazione durante l'esecuzione della query. Una query a esecuzione prolungata può visualizzare dati diversi all'inizio della query rispetto a quelli visualizzati alla fine. Una lettura fantasma si verifica quando altre transazioni causano la riorganizzazione delle righe esistenti mentre la query è in esecuzione e una o più righe vengono lette due volte dalla query.

Le query potrebbero presentare conteggi di riga incoerenti a seguito delle letture fantasma. Le tue query potrebbero anche restituire risultati incompleti o incoerenti a causa delle letture non ripetibili. Ad esempio, supponiamo che un'operazione di join si riferisca alle tabelle che sono modificate contemporaneamente dalle istruzioni SQL come `INSERT` o `DELETE`. In questo caso, la query di join potrebbe leggere una riga da una tabella ma non la riga corrispondente da un'altra tabella.

Lo standard ANSI SQL consente entrambi questi comportamenti per il livello di isolamento `READ COMMITTED`. Tuttavia, questi comportamenti sono diversi dalla tipica implementazione di MySQL di `READ COMMITTED`. Pertanto, prima di abilitare l'impostazione `aurora_read_replica_read_committed`, controlla qualsiasi codice SQL esistente per verificare se funziona come previsto nel modello di coerenza più flessibile.

Il conteggio delle righe e altri risultati potrebbero non essere propriamente coerenti nel livello di isolamento `READ COMMITTED` mentre questa impostazione è abilitata. Pertanto, in genere abiliti l'impostazione solo durante l'esecuzione di query analitiche che aggregano grandi quantità di dati e non richiedono la precisione assoluta. Se non hai questo tipo di query a esecuzione prolungata insieme a un carico di lavoro ad alta intensità di scrittura, probabilmente non è necessaria l'impostazione `aurora_read_replica_read_committed`. Senza la combinazione di query a esecuzione prolungata e carico di lavoro ad alta intensità di scrittura, è improbabile che si verifichino problemi con la lunghezza dell'elenco della cronologia.

Example Query che mostrano il comportamento di isolamento per READ COMMITTED nelle repliche di Aurora

L'esempio seguente mostra come le query `READ COMMITTED` in una replica di Aurora potrebbero restituire risultati non ripetibili se le transazioni modificano contemporaneamente le tabelle associate. La tabella `BIG_TABLE` contiene 1 milione di righe prima dell'inizio di qualsiasi query. Altre istruzioni DML (Data Manipulation Language) aggiungono, rimuovono o modificano le righe mentre sono in esecuzione.

Le query nell'istanza primaria Aurora nel livello di isolamento `READ COMMITTED` producono risultati prevedibili. Tuttavia, il sovraccarico di mantenere la visualizzazione di lettura coerente per la durata di ogni query a esecuzione prolungata può portare a costose operazioni di garbage collection in un secondo momento.

Le query nella replica di Aurora nel livello di isolamento `READ COMMITTED` sono ottimizzate per ridurre al minimo questo sovraccarico di garbage collection. Il compromesso sta nei risultati che possono variare a seconda che le query recuperino o meno le righe aggiunte, rimosse o riorganizzate dalle transazioni che eseguono il commit mentre la query è in esecuzione. Le query possono prendere in considerazione queste righe, ma non sono obbligate a farlo. A scopo dimostrativo, le query controllano solo il numero di righe nella tabella utilizzando la funzione `COUNT(*)`.

Orario	Istruzione DML nell'istanza primaria Aurora	Query nell'istanza primaria Aurora con READ COMMITTED	Query nella replica di Aurora con READ COMMITTED
T1	`INSERT INTO big_table SELECT * FROM other_table LIMIT 1000000; COMMIT;`		
T2		Q1: `SELECT COUNT(*) FROM big_table;`	Q2: `SELECT COUNT(*) FROM big_table;`
T3	`INSERT INTO big_table (c1, c2) VALUES (1, 'one more row'); COMMIT;`		
T4		Se Q1 finisce ora, il risultato è 1.000.000.	Se Q2 finisce ora, il risultato è 1.000.000 o 1.000.001.
T5	`DELETE FROM big_table LIMIT 2; COMMIT;`		
T6		Se Q1 finisce ora, il risultato è 1.000.000.	Se Q2 finisce ora, il risultato è 1.000.000 o

Orario	Istruzione DML nell'istanza primaria Aurora	Query nell'istanza primaria Aurora con READ COMMITTED	Query nella replica di Aurora con READ COMMITTED
			1.000.001 o 999.999 o 999.998.
T7	`UPDATE big_table SET c2 = CONCAT(c2 ,c2,c2); COMMIT;`		
T8		Se Q1 finisce ora, il risultato è 1.000.000.	Se Q2 finisce ora, il risultato è 1.000.000 o 1.000.001 o 999.999 oppure possibilmente un numero più alto.
T9		Q3: `SELECT COUNT(*) FROM big_table;`	Q4: `SELECT COUNT(*) FROM big_table;`
T10		Se Q3 finisce ora, il risultato è 999.999.	Se Q4 finisce ora, il risultato è 999.999.
T11		Q5: `SELECT COUNT(*) FROM parent_table p JOIN child_tab le c ON (p.id = c.id) WHERE p.id = 1000;`	Q6: `SELECT COUNT(*) FROM parent_table p JOIN child_tab le c ON (p.id = c.id) WHERE p.id = 1000;`

Orario	Istruzione DML nell'istanza primaria Aurora	Query nell'istanza primaria Aurora con READ COMMITTED	Query nella replica di Aurora con READ COMMITTED
T12	`INSERT INTO parent_table (id, s) VALUES (1000, 'hello'); INSERT INTO child_table (id, s) VALUES (1000, 'world'); COMMIT;`		
T13		Se Q5 finisce ora, il risultato è 0.	Se Q6 finisce ora, il risultato è 0 o 1.

Se le query finiscono rapidamente, prima che qualsiasi altra transazione esegua istruzioni e commit DML, i risultati sono prevedibili e uguali tra l'istanza primaria e la replica di Aurora. Esaminiamo in dettaglio le differenze di comportamento, a partire dalla prima query.

I risultati di Q1 sono prevedibili perché `READ COMMITTED` nell'istanza primaria usa un modello di consistenza elevata simile al livello di isolamento `REPEATABLE READ`.

I risultati per Q2 possono variare a seconda delle transazioni sottoposte al commit durante l'esecuzione della query. Ad esempio, supponiamo che altre transazioni eseguano istruzioni DML ed eseguano il commit mentre le query sono in esecuzione. In questo caso, la query nella replica di Aurora con il livello di isolamento `READ COMMITTED` potrebbe o meno prendere in considerazione le modifiche. I conteggi delle righe non sono prevedibili come nel livello di isolamento `REPEATABLE READ`. Inoltre, non sono prevedibili come le query in esecuzione sotto il livello di isolamento `READ COMMITTED` nell'istanza primaria o in un'istanza RDS for MySQL.

L'istruzione `UPDATE` a T7 in effetti non cambia il numero di righe della tabella. Tuttavia, modificando la lunghezza di una colonna di lunghezza variabile, questa istruzione può causare la riorganizzazione interna delle righe. Una transazione `READ COMMITTED` a esecuzione prolungata potrebbe visualizzare la versione precedente di una riga e, successivamente, all'interno della stessa query, visualizzare la nuova versione della stessa riga. La query può anche ignorare sia la versione

precedente che quella nuova della riga, quindi il conteggio delle righe potrebbe essere diverso da quello previsto.

I risultati di Q5 e Q6 potrebbero essere identici o leggermente diversi. La query Q6 nella replica di Aurora in `READ COMMITTED` è in grado di vedere, ma non è obbligata a vedere, le nuove righe che vengono confermate mentre la query è in esecuzione. Potrebbe anche vedere la riga di una tabella, ma non dell'altra tabella. Se la query di join non trova una riga corrispondente in entrambe le tabelle, restituisce un conteggio pari a zero. Se la query trova entrambe le nuove righe in `PARENT_TABLE` e `CHILD_TABLE`, la query restituisce un conteggio pari a uno. In una query a esecuzione prolungata, le ricerche dalle tabelle unite potrebbero avvenire in momenti ampiamente separati.

> **Note**
>
> Queste differenze di comportamento dipendono dal momento in cui vengono eseguite le transazioni e vengono elaborate dalle query le righe della tabella sottostante. Pertanto, è molto probabile che si notino tali differenze nelle query di report che richiedono minuti o ore e che vengono eseguite in cluster Aurora che elaborano contemporaneamente transazioni OLTP. Questi sono i tipi di carichi di lavoro misti che beneficiano maggiormente del livello di isolamento `READ COMMITTED` nelle repliche di Aurora.

Aurora I miei suggerimenti SQL

Puoi usare i SQL suggerimenti con Aurora Le SQL mie query per ottimizzare le prestazioni. È inoltre possibile utilizzare i suggerimenti per impedire che i piani di esecuzione per query importanti vengano modificati a causa di condizioni imprevedibili.

> **Tip**
>
> Per verificare l'effetto di un suggerimento su una query, esaminare il piano di query prodotto dall'istruzione `EXPLAIN`. Confrontare i piani di query con e senza il suggerimento.

In Aurora My SQL versione 3, puoi usare tutti i suggerimenti disponibili in My SQL Community Edition 8.0. Per ulteriori informazioni su questi suggerimenti, consulta Optimizer Hints nel My Reference Manual. SQL

I seguenti suggerimenti sono disponibili nella versione 2 di Aurora SQL My. Questi suggerimenti si applicano alle query che utilizzano la funzionalità hash join in Aurora My SQL versione 2, in particolare alle query che utilizzano l'ottimizzazione delle query parallele.

PQ, NO_PQ

Specifica se forzare l'ottimizzatore a utilizzare una query parallela per tabella o per query.

PQ forza l'ottimizzazione a utilizzare la query parallela per le tabelle specificate o per l'intera query (blocco). NO_PQ impedisce all'ottimizzazione di utilizzare la query parallela per le tabelle specificate o per l'intera query (blocco).

Questo suggerimento è disponibile in Aurora SQL My versione 2.11 e successive. Gli esempi seguenti mostrano come utilizzare questo suggerimento.

> **Note**
>
> La specificazione del nome di una tabella impone all'ottimizzazione di applicare il suggerimento PQ/NO_PQ solo su quelle tabelle selezionate. La mancata specificazione del nome di una tabella impone il suggerimento PQ/NO_PQ su tutte le tabelle interessate dal blocco di query.

```
EXPLAIN SELECT /*+ PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;

EXPLAIN SELECT /*+ NO_PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;
```

HASH_, NO_ _ JOIN HASH JOIN

Attiva o disattiva la possibilità dell'ottimizzazione di query parallele di scegliere se utilizzare il metodo di ottimizzazione hash join per una query. `HASH_JOIN` consente all'ottimizzazione di utilizzare hash join se tale meccanismo è più efficiente. `NO_HASH_JOIN` impedisce all'ottimizzazione di utilizzare hash join per la query. Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive. Non ha effetto nella SQL versione 3 di Aurora My.

Gli esempi seguenti mostrano come utilizzare questo suggerimento.

```
EXPLAIN SELECT/*+ HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ NO_HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_ JOIN _PROBING, NO_ _ _ HASH JOIN PROBING

In una query hash join, indica se utilizzare la tabella specificata per il lato probe del join. La query verifica se i valori delle colonne della tabella di generazione esistono nella tabella probe, anziché leggerne l'intero contenuto. È possibile utilizzare `HASH_JOIN_PROBING` e `HASH_JOIN_BUILDING` per specificare la modalità di elaborazione delle query di join hash senza riordinare le tabelle all'interno del testo della query. Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive. Non ha effetto nella SQL versione 3 di Aurora My.

Gli esempi seguenti mostrano come utilizzare questo suggerimento. Specificare il suggerimento `HASH_JOIN_PROBING` per la tabella `T2` ha lo stesso effetto di specificare `NO_HASH_JOIN_PROBING` per la tabella `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_PROBING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_PROBING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_ JOIN _BUILDING, NO_ _ _ HASH JOIN BUILDING

In una query hash join, indica se utilizzare la tabella specificata per il lato build del join. La query elabora tutte le righe di questa tabella per creare l'elenco dei valori di colonna per un riferimento incrociato con l'altra tabella. È possibile utilizzare `HASH_JOIN_PROBING` e `HASH_JOIN_BUILDING` per specificare la modalità di elaborazione delle query di join hash senza

riordinare le tabelle all'interno del testo della query. Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive. Non ha effetto nella SQL versione 3 di Aurora My.

L'esempio seguente mostra come utilizzare questo suggerimento. Specificare il suggerimento `HASH_JOIN_BUILDING` per la tabella `T2` ha lo stesso effetto di specificare `NO_HASH_JOIN_BUILDING` per la tabella `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_BUILDING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_BUILDING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

JOIN_FIXED_ORDER

Specifica che le tabelle della query vengono unite in base all'ordine in cui sono elencate nella query. È utile con le query che coinvolgono almeno tre tabelle. È inteso come sostituto del SQL `STRAIGHT_JOIN` suggerimento My ed è equivalente al suggerimento My SQL JOIN_ FIXED _ ORDER. Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_FIXED_ORDER() */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_ORDER

Specifica l'ordine join per le tabelle nella query. È utile con le query che coinvolgono almeno tre tabelle. È equivalente al suggerimento My SQL JOIN _. ORDER Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_ORDER (t4, t2, t1, t3) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_PREFIX

Specifica le tabelle da inserire per prime nell'ordine join. È utile con le query che coinvolgono almeno tre tabelle. È equivalente al suggerimento My SQL JOIN _. PREFIX Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_PREFIX (t4, t2) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_SUFFIX

Specifica le tabelle da inserire per ultime nell'ordine join. È utile con le query che coinvolgono almeno tre tabelle. È equivalente al suggerimento My SQL JOIN _. SUFFIX Questo suggerimento è disponibile in Aurora SQL My versione 2.08 e successive.

L'esempio seguente mostra come utilizzare questo suggerimento.

```
EXPLAIN SELECT /*+ JOIN_SUFFIX (t1) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

Per informazioni sull'utilizzo delle query di hash join, vedere Ottimizzazione di grandi query di join Aurora MySQL con hash join.

Aurora Il mio riferimento alla procedura SQL memorizzata

È possibile gestire il cluster Aurora My SQL DB richiamando le stored procedure integrate.

Argomenti

- Raccolta e mantenimento della cronologia dello status globale
- Configurazione, avvio e arresto della replica dei log binari (binlog)
- Terminare una sessione o una query
- Replica delle transazioni utilizzando GTIDs
- Rotazione dei log delle interrogazioni
- Impostazione e visualizzazione della configurazione del registro binario

Raccolta e mantenimento della cronologia dello status globale

Amazon RDS fornisce una serie di procedure che scattano istantanee dei valori delle variabili di stato nel tempo e le scrivono in una tabella, insieme a eventuali modifiche dall'ultima istantanea. Questa infrastruttura si chiama cronologia di stato globale. Per ulteriori informazioni, consulta Gestione della cronologia di stato globale.

Le seguenti stored procedure gestiscono il modo in cui la cronologia di stato globale viene raccolta e gestita.

Argomenti

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

Acquisisce uno snapshot on demand per la cronologia di stato globale.

Sintassi

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

Disabilita gli snapshot creati mediante la cronologia di stato globale.

Sintassi

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

Disattiva la rotazione della tabella `mysql.global_status_history`.

Sintassi

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

Abilita la cronologia di stato globale per acquisire snapshot predefiniti agli intervalli specificati da `rds_set_gsh_collector`.

Sintassi

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

Attiva la rotazione dei contenuti della tabella `mysql.global_status_history` su `mysql.global_status_history_old` agli intervalli specificati da `rds_set_gsh_rotation`.

Sintassi

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

Ruota i contenuti della tabella `mysql.global_status_history` su `mysql.global_status_history_old` a richiesta.

Sintassi

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

Specifica l'intervallo, espresso in minuti, tra gli snapshot acquisiti mediante la cronologia di stato globale.

Sintassi

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parametri

intervalPeriod

L'intervallo, in minuti, tra gli snapshot. Il valore predefinito è 5.

mysql.rds_set_gsh_rotation

Specifica l'intervallo, in giorni, tra le conversioni della tabella `mysql.global_status_history`.

Sintassi

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parametri

intervalPeriod

L'intervallo, in giorni, tra le conversioni delle tabelle. Il valore predefinito è 7.

Configurazione, avvio e arresto della replica dei log binari (binlog)

È possibile chiamare le seguenti stored procedure mentre si è connessi all'istanza principale in un cluster Aurora MySQL. Queste procedure controllano il modo in cui le transazioni vengono replicate da un database esterno in Aurora SQL My o da Aurora SQL My a un database esterno.

Argomenti

- mysql.rds_disable_session_binlog (Aurora My versione 2) SQL
- mysql.rds_enable_session_binlog (Aurora My versione 2) SQL
- mysql.rds_import_binlog_ssl_material
- mysql.rds_next_master_log SQL
- mysql.rds_next_source_log SQL
- mysql.rds_remove_binlog_ssl_material
- mysql.rds_reset_external_master Aurora My versione 2) SQL SQL
- mysql.rds_reset_external_source SQL
- mysql.rds_set_binlog_source_ssl (Aurora La mia versione 3) SQL
- mysql.rds_set_external_master SQL
- mysql.rds_set_external_source () SQL
- mysql.rds_set_external_master_with_auto_position (Aurora La mia versione 2) SQL
- mysql.rds_set_external_source_with_auto_position (Aurora My versione 3) SQL
- mysql.rds_set_master_auto_position SQL
- mysql.rds_set_read_only (Aurora My versione 3) SQL
- mysql.rds_set_session_binlog_format (Aurora La mia versione 2) SQL
- mysql.rds_set_source_auto_position SQL
- mysql.rds_skip_repl_error
- mysql.rds_start_replication
- mysql.rds_start_replication_until (Aurora My versione 3) SQL
- mysql.rds_stop_replication

mysql.rds_disable_session_binlog (Aurora My versione 2) SQL

Disattiva la registrazione binaria per la sessione corrente impostando la variabile `sql_log_bin` su `OFF`.

Sintassi

```
CALL mysql.rds_disable_session_binlog;
```

Parametri

Nessuno

Note per l'utilizzo

Per un cluster Aurora My SQL DB, si chiama questa stored procedure mentre si è connessi all'istanza principale.

Per Aurora, questa procedura è supportata per Aurora My SQL versione 2.12 e versioni successive compatibili con My 5.7. SQL

> **Note**
>
> In Aurora My SQL versione 3, puoi usare il seguente comando per disabilitare la registrazione binaria per la sessione corrente se disponi del privilegio: `SESSION_VARIABLES_ADMIN`
>
> ```
> SET SESSION sql_log_bin = OFF;
> ```

mysql.rds_enable_session_binlog (Aurora My versione 2) SQL

Attiva la registrazione binaria per la sessione corrente impostando la variabile `sql_log_bin` su `ON`.

Sintassi

```
CALL mysql.rds_enable_session_binlog;
```

Parametri

Nessuno

Note per l'utilizzo

Per un cluster Aurora My SQL DB, si chiama questa stored procedure mentre si è connessi all'istanza principale.

Per Aurora, questa procedura è supportata per Aurora My SQL versione 2.12 e versioni successive compatibili con My 5.7. SQL

> **Note**
>
> In Aurora My SQL versione 3, è possibile utilizzare il seguente comando per abilitare la registrazione binaria per la sessione corrente se si dispone del privilegio: `SESSION_VARIABLES_ADMIN`
>
> ```
> SET SESSION sql_log_bin = ON;
> ```

mysql.rds_import_binlog_ssl_material

Importa il certificato dell'autorità di certificazione, il certificato client e la chiave client in un cluster Aurora My SQL DB RDS for My SQL DB . Le informazioni sono necessarie per la SSL comunicazione e la replica crittografata.

> **Note**
>
> Attualmente, questa procedura è supportata per Aurora My SQL versione 2:2.09.2, 2.10.0, 2.10.1 e 2.11.0 e versione 3:3.01.1 e successive.

Sintassi

```
CALL mysql.rds_import_binlog_ssl_material (
  ssl_material
);
```

Parametri

ssl_material

JSONpayload che contiene il contenuto dei seguenti file in formato.pem per un client My: SQL

- «ssl_ca»:»» *Certificate authority certificate*
- «certificato SSL»:»» *Client certificate*
- «chiave_ssl»:»» *Client key*

Note per l'utilizzo

Prepara la replica crittografata prima di eseguire questa procedura:

- Se non hai SSL abilitato l'istanza esterna del database My SQL source e non hai preparato una chiave client e un certificato client, abilitalali SSL sul server My SQL database e genera la chiave client e il certificato client richiesti.
- Se SSL è abilitato sull'istanza del database di origine esterno, fornisci una chiave client e un certificato per il cluster Aurora My SQL DB. Se non li possiedi, genera una nuova chiave e un nuovo certificato per il cluster Aurora My SQL DB. Per firmare il certificato client, è necessario disporre della chiave di autorità di certificazione utilizzata per la configurazione SSL sull'istanza esterna del database My SQL source.

Per ulteriori informazioni, consulta Creazione di SSL certificati e chiavi utilizzando openssl nella documentazione personaleSQL.

> **Important**
>
> Dopo esserti preparato per la replica crittografata, utilizza una SSL connessione per eseguire questa procedura. La chiave client non deve essere trasferita mediante una connessione non sicura.

Questa procedura importa SSL informazioni da un SQL database My esterno in un cluster Aurora My SQL DB. Le SSL informazioni sono in file in formato.pem che contengono SSL le informazioni per il cluster Aurora My DB. SQL Durante la replica crittografata, il cluster Aurora SQL My DB funge da client verso il server del database SQL My. I certificati e le chiavi per il SQL client Aurora My sono in file in formato.pem.

È possibile copiare le informazioni da questi file nel `ssl_material` parametro nel payload corretto. JSON Per supportare la replica crittografata, importa queste SSL informazioni nel cluster Aurora SQL My DB.

Il JSON payload deve essere nel formato seguente.

```
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
ssl_ca_pem_body_code
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
```

```
ssl_cert_pem_body_code
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
ssl_key_pem_body_code
-----END RSA PRIVATE KEY-----\n"}'
```

Esempi

L'esempio seguente importa SSL informazioni in un' Aurora My. SQL Nei file in formato .pem, il codice del corpo è in genere più lungo del codice del corpo riportato nell'esempio.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

mysql.rds_next_master_log SQL

Cambia la posizione del log dell'istanza database di origine all'inizio del successivo log binario nell'istanza database di origine. Utilizza questa procedura solo se ricevi un errore I/O di replica 1236 su una replica di lettura.

Sintassi

```
CALL mysql.rds_next_master_log(
```

```
curr_master_log
);
```

Parametri

curr_master_log

L'indice del file di log master corrente. Ad esempio, se il file corrente è denominato `mysql-bin-changelog.012345`, l'indice è 12345. Per determinare il nome del file di log master corrente, esegui il comando `SHOW REPLICA STATUS` e visualizza il campo `Master_Log_File`.

Note per l'utilizzo

La procedura `mysql.rds_next_master_log` deve essere eseguita dall'utente master.

> **Warning**
>
> Chiama `mysql.rds_next_master_log` solo se la replica non riesce dopo un failover di un'istanza database Multi-AZ DB che è l'origine della replica e il campo `Last_IO_Errno` di `SHOW REPLICA STATUS` segnala l'errore I/O 1236.
> La chiamata di `mysql.rds_next_master_log` può comportare una perdita di dati nella replica di lettura se le transazioni nell'istanza di origine non sono state scritte nel log binario sul disco prima dell'evento di failover.

Esempi

Supponiamo che la replica fallisca su una RDS replica di Aurora SQL My read. L'esecuzione di `SHOW REPLICA STATUS\G` nella replica di lettura restituisce il risultato seguente:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
```

```
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Il campo `Last_IO_Errno` mostra che l'istanza riceve l'errore I/O 1236. Il campo `Master_Log_File` mostra che il nome di file è `mysql-bin-changelog.012345`, il che significa che l'indice del file di log è `12345`. Per risolvere il problema, puoi chiamare `mysql.rds_next_master_log` con il seguente parametro:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log SQL

Cambia la posizione del log dell'istanza database di origine all'inizio del successivo log binario nell'istanza database di origine. Utilizza questa procedura solo se ricevi un errore I/O di replica 1236 su una replica di lettura.

Sintassi

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

Parametri

curr_source_log

L'indice del file di log di origine corrente. Ad esempio, se il file corrente è denominato `mysql-bin-changelog.012345`, l'indice è 12345. Per determinare il nome del file di log di origine corrente, esegui il comando `SHOW REPLICA STATUS` e visualizza il campo `Source_Log_File`.

Note per l'utilizzo

L'utente amministrativo deve eseguire la procedura `mysql.rds_next_source_log`.

Warning

Chiama `mysql.rds_next_source_log` solo se la replica non riesce dopo un failover di un'istanza database Multi-AZ DB che è l'origine della replica e il campo `Last_IO_Errno` di `SHOW REPLICA STATUS` segnala l'errore I/O 1236.
La chiamata di `mysql.rds_next_source_log` può comportare una perdita di dati nella replica di lettura se le transazioni nell'istanza di origine non sono state scritte nel log binario sul disco prima dell'evento di failover. È possibile ridurre la possibilità che ciò accada impostando i parametri dell'istanza di origine e su, anche se ciò potrebbe ridurre le prestazioni. `sync_binlog` `innodb_support_xa` `1`

Esempi

Supponiamo che la replica fallisca su una RDS replica di Aurora SQL My read. L'esecuzione di `SHOW REPLICA STATUS\G` nella replica di lettura restituisce il risultato seguente:

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
```

```
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Il campo `Last_IO_Errno` mostra che l'istanza riceve l'errore I/O 1236. Il campo `Source_Log_File` mostra che il nome di file è `mysql-bin-changelog.012345`, il che significa che l'indice del file di log è `12345`. Per risolvere il problema, puoi chiamare `mysql.rds_next_source_log` con il seguente parametro:

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_remove_binlog_ssl_material

Rimuove il certificato dell'autorità di certificazione, il certificato client e la chiave client per la SSL comunicazione e la replica crittografata. Queste informazioni sono importate utilizzando mysql.rds_import_binlog_ssl_material.

Sintassi

```
CALL mysql.rds_remove_binlog_ssl_material;
```

mysql.rds_reset_external_master Aurora My versione 2) SQL SQL

Riconfigura un'istanza My SQL DB in modo che non sia più una replica di lettura di un'istanza di My in esecuzione SQL esterna ad Amazon. RDS

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su 1. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri DB in .

Sintassi

```
CALL mysql.rds_reset_external_master;
```

Note per l'utilizzo

La procedura `mysql.rds_reset_external_master` deve essere eseguita dall'utente master. Questa procedura deve essere eseguita sull'istanza My SQL DB per essere rimossa come replica di lettura di un'SQListanza My in esecuzione all'esterno di AmazonRDS.

> **Note**
>
> Offriamo queste procedure archiviate principalmente per consentire la replica con SQL istanze My in esecuzione all'esterno di Amazon. RDS Si consiglia di utilizzare Aurora Replicas per gestire la replica all'interno di un cluster Aurora My DB, quando possibile. SQL Per informazioni sulla gestione della replica nei cluster Aurora SQL My DB, vedere. Utilizzo delle repliche di Aurora

Per ulteriori informazioni sull'utilizzo della replica per importare dati da un'istanza di My SQL in esecuzione esterna ad Aurora SQL My, vedere. Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari)

mysql.rds_reset_external_source SQL

Riconfigura un'istanza My SQL DB in modo che non sia più una replica di lettura di un'istanza di My in esecuzione SQL esterna ad Amazon. RDS

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su `1`. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri DB in .

Sintassi

```
CALL mysql.rds_reset_external_source;
```

Note per l'utilizzo

L'utente amministrativo deve eseguire la procedura `mysql.rds_reset_external_source`. Questa procedura deve essere eseguita sull'istanza My SQL DB per essere rimossa come replica di lettura di un'SQListanza My in esecuzione all'esterno di AmazonRDS.

> **Note**
>
> Offriamo queste procedure archiviate principalmente per consentire la replica con SQL istanze My in esecuzione all'esterno di Amazon. RDS Si consiglia di utilizzare Aurora Replicas per gestire la replica all'interno di un cluster Aurora My DB, quando possibile. SQL Per informazioni sulla gestione della replica nei cluster Aurora SQL My DB, vedere. Utilizzo delle repliche di Aurora

mysql.rds_set_binlog_source_ssl (Aurora La mia versione 3) SQL

Abilita la `SOURCE_SSL` crittografia per la replica binlog. Per ulteriori informazioni, vedere CHANGEREPLICATIONSOURCEl'istruzione TO nella documentazione personaleSQL.

Sintassi

```
CALL mysql.rds_set_binlog_source_ssl(mode);
```

Parametri

mode

Un valore che indica se la `SOURCE_SSL` crittografia è abilitata:

- `0`— `SOURCE_SSL` la crittografia è disabilitata. Il valore predefinito è `0`.
- `1`— `SOURCE_SSL` la crittografia è abilitata. È possibile configurare la crittografia utilizzando SSL oTLS.

Note per l'utilizzo

Questa procedura è supportata per Aurora My SQL versione 3.06 e successive.

mysql.rds_set_external_master SQL

Configura un'istanza Aurora in modo che sia una replica di lettura di un'istanza di SQL My in esecuzione esterna ad Amazon. RDS

La procedura `mysql.rds_set_external_master` è obsoleta e verrà rimossa in una versione futura. Usare invece `mysql.rds_set_external_source`.

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su 1. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri DB in .

Sintassi

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'SQListanza My in esecuzione all'esterno di Amazon RDS per diventare l'istanza del database di origine.

host_port

La porta utilizzata dall'SQListanza My in esecuzione esterna ad Amazon RDS per essere configurata come istanza del database di origine. Se la configurazione di rete include la replica delle porte Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta da cui è esposta. SSH

replication_user_name

L'ID di un utente con `REPLICATION CLIENT` e `REPLICATION SLAVE` le autorizzazioni sull'SQListanza My in esecuzione all'esterno di AmazonRDS. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

mysql_binary_log_file_name

Il nome del log binario sull'istanza database di origine che contiene le informazioni relative alla replica.

mysql_binary_log_file_location

La posizione nel log binario `mysql_binary_log_file_name` a partire dalla quale la replica inizia a leggere le informazioni a essa relative.

È possibile determinare il nome e la posizione del file binlog in esecuzione `SHOW MASTER STATUS` sull'istanza del database di origine.

ssl_encryption

Un valore che specifica se la crittografia Secure Socket Layer (SSL) viene utilizzata sulla connessione di replica. 1 specifica di utilizzare la SSL crittografia, 0 specifica di non utilizzare la crittografia. Il valore predefinito è 0.

> **Note**
>
> L'opzione `MASTER_SSL_VERIFY_SERVER_CERT` non è supportata. Questa opzione è impostata su 0, il che significa che la connessione è crittografata, ma i certificati non sono verificati.

Note per l'utilizzo

La procedura `mysql.rds_set_external_master` deve essere eseguita dall'utente master. Questa procedura deve essere eseguita sull'istanza My SQL DB per essere configurata come replica di lettura di un'SQListanza My in esecuzione all'esterno di AmazonRDS.

Prima dell'esecuzione`mysql.rds_set_external_master`, devi configurare l'istanza di My SQL running external RDS to Amazon come istanza di database di origine. Per connetterti all'SQListanza

My in esecuzione esternamente ad AmazonRDS, devi specificare `replication_user_name` `replication_user_password` i valori che indicano un utente di replica che dispone delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` delle autorizzazioni sull'istanza esterna di My. SQL

Per configurare un'istanza esterna di My SQL come istanza del database di origine

1. Utilizzando il SQL client My di tua scelta, connettiti all'istanza esterna di My SQL e crea un account utente da utilizzare per la replica. Di seguito è riportato un esempio.

   Il mio 5.7 SQL

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   Il mio 8.0 SQL

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

   > **Note**
   >
   > Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

2. Sull'istanza esterna di MySQL, concedi `REPLICATION CLIENT` e `REPLICATION SLAVE` privilegi al tuo utente di replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente "repl_user" del dominio:

   Il mio 5.7 SQL

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   Il mio 8.0 SQL

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Per utilizzare la replica crittografata, configura l'istanza del database di origine per utilizzare SSL le connessioni. Importa inoltre il certificato dell'autorità di certificazione, il certificato client e la chiave client nell'istanza database o nel cluster di database utilizzando la procedura mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Offriamo queste procedure archiviate principalmente per consentire la replica con SQL istanze My in esecuzione all'esterno di Amazon. RDS Si consiglia di utilizzare Aurora Replicas per gestire la replica all'interno di un cluster Aurora My DB, quando possibile. SQL Per informazioni sulla gestione della replica nei cluster Aurora SQL My DB, vedere. Utilizzo delle repliche di Aurora

Dopo aver chiamato `mysql.rds_set_external_master` per configurare un'istanza Amazon RDS DB come replica di lettura, puoi richiamare mysql.rds_start_replication la replica di lettura per avviare il processo di replica. Puoi chiamare mysql.rds_reset_external_master Aurora My versione 2) SQL SQL per rimuovere la configurazione della replica di lettura.

Quando `mysql.rds_set_external_master` viene chiamato, Amazon RDS registra l'ora, l'utente e un'azione di `set master` nelle `mysql.rds_replication_status` tabelle `mysql.rds_history` e.

Esempi

Quando viene eseguita su un'istanza My SQL DB, l'esempio seguente configura l'istanza DB come replica di lettura di un'istanza di My SQL in esecuzione esterna ad Amazon. RDS

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source () SQL

Configura un'istanza Aurora in modo che sia una replica di lettura di un'istanza di SQL My in esecuzione esterna ad Amazon. RDS

> **Important**
>
> Per eseguire questa procedura, è necessario abilitare `autocommit`. Per abilitarlo, impostare il parametro `autocommit` su 1. Per ulteriori informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri DB in .

Sintassi

```
CALL mysql.rds_set_external_source (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'SQListanza My in esecuzione all'esterno di Amazon RDS per diventare l'istanza del database di origine.

host_port

La porta utilizzata dall'SQListanza My in esecuzione esterna ad Amazon RDS per essere configurata come istanza del database di origine. Se la configurazione di rete include la replica delle porte Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta da cui è esposta. SSH

replication_user_name

L'ID di un utente con `REPLICATION CLIENT` e `REPLICATION SLAVE` le autorizzazioni sull'SQListanza My in esecuzione all'esterno di AmazonRDS. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

mysql_binary_log_file_name

Il nome del log binario sull'istanza database di origine che contiene le informazioni relative alla replica.

mysql_binary_log_file_location

La posizione nel log binario `mysql_binary_log_file_name` a partire dalla quale la replica inizia a leggere le informazioni a essa relative.

È possibile determinare il nome e la posizione del file binlog in esecuzione `SHOW MASTER STATUS` sull'istanza del database di origine.

ssl_encryption

Un valore che specifica se la crittografia Secure Socket Layer (SSL) viene utilizzata sulla connessione di replica. 1 specifica di utilizzare la SSL crittografia, 0 specifica di non utilizzare la crittografia. Il valore predefinito è 0.

> **Note**
>
> È necessario aver importato un SSL certificato personalizzato utilizzando mysql.rds_import_binlog_ssl_material per abilitare questa opzione. Se non avete importato un SSL certificato personalizzato, impostate questo parametro su 0 e utilizzatelo SSL per mysql.rds_set_binlog_source_ssl (Aurora La mia versione 3) SQL abilitare la replica dei log binari.
> L'opzione `SOURCE_SSL_VERIFY_SERVER_CERT` non è supportata. Questa opzione è impostata su 0, il che significa che la connessione è crittografata, ma i certificati non sono verificati.

Note per l'utilizzo

L'utente amministrativo deve eseguire la procedura `mysql.rds_set_external_source`. Questa procedura deve essere eseguita sull'istanza Aurora My SQL DB per essere configurata come replica di lettura di un'SQListanza My in esecuzione esterna ad Amazon. RDS

Prima dell'esecuzione`mysql.rds_set_external_source`, devi configurare l'istanza di My SQL running external RDS to Amazon come istanza di database di origine. Per connetterti all'SQListanza My in esecuzione esternamente ad AmazonRDS, devi specificare `replication_user_name` `replication_user_password` i valori che indicano un utente di replica che dispone delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` delle autorizzazioni sull'istanza esterna di My. SQL

Per configurare un'istanza esterna di My SQL come istanza del database di origine

1. Utilizzando il SQL client My di tua scelta, connettiti all'istanza esterna di My SQL e crea un account utente da utilizzare per la replica. Di seguito è riportato un esempio.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   > **Note**
   >
   > Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

2. Sull'istanza esterna di MySQL, concedi `REPLICATION SLAVE` i `REPLICATION CLIENT` privilegi al tuo utente di replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente "repl_user" del dominio:

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Per utilizzare la replica crittografata, configura l'istanza del database di origine per utilizzare le connessioni. SSL Importa inoltre il certificato dell'autorità di certificazione, il certificato client e la chiave client nell'istanza database o nel cluster database utilizzando la procedura mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Offriamo queste procedure archiviate principalmente per consentire la replica con SQL istanze My in esecuzione all'esterno di Amazon. RDS Si consiglia di utilizzare Aurora Replicas per gestire la replica all'interno di un cluster Aurora My DB, quando possibile. SQL Per informazioni sulla gestione della replica nei cluster Aurora SQL My DB, vedere. Utilizzo delle repliche di Aurora

Dopo aver chiamato `mysql.rds_set_external_source` per configurare un'istanza di Aurora My SQL RDS DB come replica di lettura, puoi chiamare mysql.rds_start_replication la replica di lettura per avviare il processo di replica. Puoi chiamare mysql.rds_reset_external_source SQL per rimuovere la configurazione della replica di lettura.

Quando `mysql.rds_set_external_source` viene chiamato, Amazon RDS registra l'ora, l'utente e un'azione di `set master` nelle `mysql.rds_replication_status` tabelle `mysql.rds_history` e.

Esempi

Quando viene eseguita su un'istanza Aurora My SQL RDS DB, l'esempio seguente configura l'istanza DB come replica di lettura di un'istanza di My in esecuzione SQL esterna ad Amazon. RDS

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_master_with_auto_position (Aurora La mia versione 2) SQL

Configura un'istanza principale Aurora SQL My per accettare la replica in entrata da un'istanza My esterna. SQL Questa procedura configura anche la replica in base agli identificatori di transazione globali (). GTIDs

Questa procedura non configura la replica ritardata, perché Aurora My SQL non supporta la replica ritardata.

Sintassi

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'SQListanza My in esecuzione all'esterno di Aurora per diventare l'origine di replica.

host_port

La porta utilizzata dall'SQListanza My in esecuzione esterna ad Aurora per essere configurata come origine di replica. Se la configurazione di rete include la replica delle porte Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta da cui è esposto. SSH

replication_user_name

L'ID di un utente con `REPLICATION CLIENT` e `REPLICATION SLAVE` le autorizzazioni sull'SQListanza My in esecuzione esterna ad Aurora. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

ssl_encryption

Questa opzione non è al momento implementata. Il valore predefinito è 0.

Note per l'utilizzo

Per un cluster Aurora My SQL DB, si chiama questa stored procedure mentre si è connessi all'istanza principale.

La procedura `mysql.rds_set_external_master_with_auto_position` deve essere eseguita dall'utente master. L'utente master esegue questa procedura sull'istanza principale di un cluster

Aurora My SQL DB che funge da target di replica. Può essere la destinazione di replica di un'istanza My SQL DB esterna o di un cluster Aurora SQL My DB.

Questa procedura è supportata per Aurora My SQL versione 2. Per Aurora My SQL versione 3, utilizzare invece la procedura mysql.rds_set_external_source_with_auto_position (Aurora My versione 3) SQL.

Prima dell'esecuzione `mysql.rds_set_external_master_with_auto_position`, configura l'istanza esterna di My SQL DB come fonte di replica. Per connetterti all'SQListanza My esterna, specifica i valori per `replication_user_name` e `replication_user_password`. Questi valori devono indicare un utente di replica che dispone delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` delle autorizzazioni sull'istanza My SQL esterna.

Per configurare un'SQListanza My esterna come origine di replica

1. Utilizzando il SQL client My di tua scelta, connettiti all'SQListanza My esterna e crea un account utente da utilizzare per la replica. Di seguito è riportato un esempio.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Sull'SQListanza My esterna, concedi `REPLICATION CLIENT` e `REPLICATION SLAVE` privilegi al tuo utente di replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente `'repl_user'` per il dominio.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Quando chiami `mysql.rds_set_external_master_with_auto_position`, Amazon RDS registra determinate informazioni. Queste informazioni sono l'orario, l'utente e l'operazione di `"set master"` nelle tabelle `mysql.rds_history` e `mysql.rds_replication_status`.

Per saltare una transazione GTID specifica che è nota come causa di un problema, puoi utilizzare la mysql.rds_skip_transaction_with_gtid (Aurora My versione 2 e 3) SQL stored procedure. Per ulteriori informazioni sull'utilizzo della replica GTID basata, vedere. Utilizzo della replica GTID basata

Esempi

Quando viene eseguito su un'istanza primaria di Aurora, l'esempio seguente configura il cluster Aurora in modo che funga da replica di lettura di un'istanza di My running esterna ad Aurora. SQL

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_external_source_with_auto_position (Aurora My versione 3) SQL

Configura un'istanza principale Aurora SQL My per accettare la replica in entrata da un'istanza My esterna. SQL Questa procedura configura anche la replica in base agli identificatori di transazione globali (). GTIDs

Sintassi

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Parametri

host_name

Il nome host o l'indirizzo IP dell'SQListanza My in esecuzione all'esterno di Aurora per diventare l'origine di replica.

host_port

La porta utilizzata dall'SQListanza My in esecuzione esterna ad Aurora per essere configurata come origine di replica. Se la configurazione di rete include la replica delle porte Secure Shell (SSH) che converte il numero di porta, specifica il numero di porta da cui è esposto. SSH

replication_user_name

L'ID di un utente con `REPLICATION CLIENT` e `REPLICATION SLAVE` le autorizzazioni sull'SQListanza My in esecuzione esterna ad Aurora. Ti consigliamo di fornire un account utilizzato unicamente per la replica con l'istanza esterna.

replication_user_password

La password dell'ID utente specificata in `replication_user_name`.

ssl_encryption

Questa opzione non è al momento implementata. Il valore predefinito è 0.

> **Note**
>
> mysql.rds_set_binlog_source_ssl (Aurora La mia versione 3) SQLDa utilizzare SSL per abilitare la replica dei log binari.

Note per l'utilizzo

Per un cluster Aurora My SQL DB, si chiama questa stored procedure mentre si è connessi all'istanza principale.

L'utente amministrativo deve eseguire la procedura `mysql.rds_set_external_source_with_auto_position`. L'utente amministrativo esegue questa procedura sull'istanza principale di un cluster Aurora My SQL DB che funge da destinazione di replica. Può essere la destinazione di replica di un'istanza My SQL DB esterna o di un cluster Aurora SQL My DB.

Questa procedura è supportata per Aurora My SQL versione 3. Questa procedura non configura la replica ritardata, perché Aurora My SQL non supporta la replica ritardata.

Prima dell'esecuzione`mysql.rds_set_external_source_with_auto_position`, configura l'istanza My SQL DB esterna come fonte di replica. Per connetterti all'SQListanza My esterna, specifica i valori per `replication_user_name` e`replication_user_password`. Questi valori devono indicare un utente di replica che dispone delle autorizzazioni `REPLICATION CLIENT` e `REPLICATION SLAVE` delle autorizzazioni sull'istanza My SQL esterna.

Per configurare un'SQListanza My esterna come origine di replica

1. Utilizzando il SQL client My di tua scelta, connettiti all'SQListanza My esterna e crea un account utente da utilizzare per la replica. Di seguito è riportato un esempio.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Sull'SQListanza My esterna, concedi `REPLICATION CLIENT` e `REPLICATION SLAVE` privilegi al tuo utente di replica. L'esempio seguente concede i privilegi `REPLICATION CLIENT` e `REPLICATION SLAVE` su tutti i database per l'utente `'repl_user'` per il dominio.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

Quando chiamimysql.rds_set_external_source_with_auto_position, Amazon RDS registra determinate informazioni. Queste informazioni sono l'orario, l'utente e l'operazione di "set master" nelle tabelle mysql.rds_history e mysql.rds_replication_status.

Per saltare una transazione GTID specifica che è nota come causa di un problema, puoi utilizzare la mysql.rds_skip_transaction_with_gtid (Aurora My versione 2 e 3) SQL stored procedure. Per ulteriori informazioni sull'utilizzo della replica GTID basata, vedere. Utilizzo della replica GTID basata

Esempi

Quando viene eseguito su un'istanza primaria di Aurora, l'esempio seguente configura il cluster Aurora in modo che funga da replica di lettura di un'istanza di My running esterna ad Aurora. SQL

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_master_auto_position SQL

Imposta la modalità di replica in modo che sia basata sulle posizioni dei file di registro binari o sugli identificatori globali delle transazioni (). GTIDs

Sintassi

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

Parametri

auto_position_mode

Un valore che indica se utilizzare la replica della posizione dei file di registro o GTID la replica basata:

- `0` – Usa il metodo di replica basato sulla posizione del file di log binario. Il valore di default è `0`.
- 1— Utilizzare il metodo di replica GTID basato.

Note per l'utilizzo

La procedura `mysql.rds_set_master_auto_position` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora My SQL versione 2.

mysql.rds_set_read_only (Aurora My versione 3) SQL

Attiva o disattiva la `read_only` modalità a livello globale per l'istanza DB.

Sintassi

```
CALL mysql.rds_set_read_only(mode);
```

Parametri

mode

Un valore che indica se la `read_only` modalità è attiva o disattivata a livello globale per l'istanza DB:

- `0`—OFF. L'impostazione predefinita è`0`.
- 1 – ON

Note per l'utilizzo

La `mysql.rds_set_read_only` stored procedure modifica solo il `read_only` parametro. Il `innodb_read_only` parametro non può essere modificato sulle istanze Reader DB.

La modifica `read_only` dei parametri non persiste al riavvio. Per apportare modifiche permanenti `aread_only`, è necessario utilizzare il parametro del cluster `read_only` DB.

Questa procedura è supportata per Aurora My SQL versione 3.06 e successive.

mysql.rds_set_session_binlog_format (Aurora La mia versione 2) SQL

Imposta il formato di log binario per la sessione corrente.

Sintassi

```
CALL mysql.rds_set_session_binlog_format(format);
```

Parametri

format

Un valore che indica il formato di log binario per la sessione corrente:

- STATEMENT— La fonte di replica scrive gli eventi nel log binario in base alle istruzioni. SQL
- ROW: l'origine della replica scrive eventi nel log binario che indicano le modifiche alle singole righe della tabella.
- MIXED— La registrazione si basa generalmente su SQL istruzioni, ma passa alle righe in determinate condizioni. Per ulteriori informazioni, vedere Mixed Binary Logging Format nella documentazione personale. SQL

Note per l'utilizzo

Per un cluster Aurora My SQL DB, si chiama questa stored procedure mentre si è connessi all'istanza principale.

Per utilizzare questa procedura archiviata, la registrazione binaria deve essere configurata per la sessione corrente.

Per Aurora, questa procedura è supportata per Aurora My SQL versione 2.12 e versioni successive compatibili con My 5.7. SQL

mysql.rds_set_source_auto_position SQL

Imposta la modalità di replica in modo che sia basata sulle posizioni dei file di registro binari o sugli identificatori globali delle transazioni (). GTIDs

Sintassi

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

Parametri

auto_position_mode

Un valore che indica se utilizzare la replica della posizione dei file di registro o GTID la replica basata:

- `0` – Usa il metodo di replica basato sulla posizione del file di log binario. Il valore di default è `0`.
- 1— Utilizzare il metodo di replica GTID basato.

Note per l'utilizzo

Per un cluster Aurora My SQL DB, si chiama questa stored procedure mentre si è connessi all'istanza principale.

L'utente amministrativo deve eseguire la procedura `mysql.rds_set_source_auto_position`.

mysql.rds_skip_repl_error

Ignora ed elimina un errore di replica su una replica di lettura di My DB. SQL

Sintassi

```
CALL mysql.rds_skip_repl_error;
```

Note per l'utilizzo

La procedura `mysql.rds_skip_repl_error` deve essere eseguita dall'utente master su una replica di lettura. Per ulteriori informazioni su questa procedura, consulta Ignorare l'errore di replica corrente.

Per determinare se ci sono errori, esegui il comando My. SQL `SHOW REPLICA STATUS \G` Se un errore di replica non è critico, puoi eseguire `mysql.rds_skip_repl_error` per ignorare l'errore. Se vi sono più errori, `mysql.rds_skip_repl_error` elimina il primo, quindi informa della presenza di altri errori. Puoi quindi utilizzare `SHOW REPLICA STATUS\G` per determinare l'operazione corretta per l'errore successivo. Per informazioni sui valori restituiti, vedere l'SHOWREPLICASTATUSistruzione nella SQL documentazione My.

Per ulteriori informazioni sulla risoluzione degli errori di replica con Aurora SQL My, consulta. Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL

Errore di replica interrotta

Quando si chiama la procedura `mysql.rds_skip_repl_error`, è possibile che venga visualizzato un messaggio di errore che indica che la replica è inattiva o disattivata.

Questo messaggio di errore viene visualizzato se si esegue la procedura sull'istanza primaria anziché sulla replica di lettura. È necessario eseguire questa procedura sulla replica di lettura affinché funzioni.

Questo messaggio di errore può essere visualizzato anche quando si esegue la procedura sulla replica di lettura, ma la replica non viene riavviata correttamente.

Se devi ignorare un numero elevato di errori, il ritardo della replica potrebbe superare il periodo di retention predefinito per i file di log binari (binlog). In questo caso, è possibile che si verifichi un errore irreversibile a causa dell'eliminazione dei file binlog prima che vengano riprodotti nella replica di lettura. Questa eliminazione causa l'arresto della replica e non è più possibile chiamare il comando `mysql.rds_skip_repl_error` per ignorare errori di replica.

Puoi limitare questo problema aumentando il numero di ore di retention dei file binlog nell'istanza database di origine. Una volta aumentato il tempo di retention dei file binlog, puoi riavviare la replica e chiamare il comando `mysql.rds_skip_repl_error` secondo necessità.

Per impostare il periodo di retention dei file binlog, usa la procedura mysql.rds_set_configuration e specifica un parametro di configurazione di `'binlog retention hours'` insieme al numero di ore di retention dei file binlog nel cluster di database. Nell'esempio seguente il periodo di retention dei file binlog è impostato su 48 ore.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

Avvia la replica da un cluster instance. SQL

> **Note**
>
> È possibile utilizzare la procedura mysql.rds_start_replication_until (Aurora My versione 3) SQL or mysql.rds_start_replication_until_gtid (Aurora La mia versione 3) SQL stored per avviare la replica da un'istanza My SQL DB e interrompere la replica nella posizione specificata del file di log binario.

Sintassi

```
CALL mysql.rds_start_replication;
```

Note per l'utilizzo

La procedura `mysql.rds_start_replication` deve essere eseguita dall'utente master.

Per importare dati da un'istanza di My SQL external ad AmazonRDS, richiama `mysql.rds_start_replication` la replica di lettura per avviare il processo di replica dopo la chiamata mysql.rds_set_external_master SQL o mysql.rds_set_external_source () SQL per creare la configurazione di replica. Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Per esportare i dati in un'istanza di My SQL external `mysql.rds_stop_replication` su AmazonRDS, chiama `mysql.rds_start_replication` e accendi la replica di lettura per controllare alcune azioni di replica, come l'eliminazione dei log binari. Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Puoi anche chiamare `mysql.rds_start_replication` nella replica di lettura per riavviare un processo di replica arrestato in precedenza chiamando `mysql.rds_stop_replication`. Per ulteriori informazioni, consulta Errore di replica interrotta.

mysql.rds_start_replication_until (Aurora My versione 3) SQL

Avvia la replica da un cluster instance e interrompe la SQL replica nella posizione specificata del file di log binario.

Sintassi

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

Parametri

replication_log_file

Il nome del log binario sull'istanza database di origine che contiene le informazioni relative alla replica.

replication_stop_point

Posizione nel log binario `replication_log_file` in corrispondenza di cui la replica verrà arrestata.

Note per l'utilizzo

La procedura `mysql.rds_start_replication_until` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora My SQL versione 3.04 e successive.

La `mysql.rds_start_replication_until` stored procedure non è supportata per la replica gestita, che include quanto segue:

- Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS
- Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora

Il nome file specificato per il parametro `replication_log_file` deve corrispondere al nome file binlog dell'istanza database di origine.

Quando il parametro `replication_stop_point` specifica una posizione di arresto nel passato, la replica viene arrestata immediatamente.

Esempi

L'esempio seguente avvia la replica e replica le modifiche fino a raggiungere la posizione `120` nel file di log binario `mysql-bin-changelog.000777`.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_stop_replication

Interrompe la replica da un'istanza My SQL DB.

Sintassi

```
CALL mysql.rds_stop_replication;
```

Note per l'utilizzo

La procedura `mysql.rds_stop_replication` deve essere eseguita dall'utente master.

Se stai configurando la replica per importare dati da un'istanza di My SQL in esecuzione esterna ad AmazonRDS, `mysql.rds_stop_replication` richiami la replica di lettura per interrompere il processo di replica una volta completata l'importazione. Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

Se stai configurando la replica per esportare dati su un'istanza di My SQL esterna ad AmazonRDS, chiami `mysql.rds_start_replication` e `mysql.rds_stop_replication` accendi la replica di lettura per controllare alcune azioni di replica, come l'eliminazione dei log binari. Per ulteriori informazioni, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari).

La `mysql.rds_stop_replication` stored procedure non è supportata per la replica gestita, che include quanto segue:

- Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS
- Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora

Terminare una sessione o una query

Le seguenti stored procedure terminano una sessione o una query.

Argomenti

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

Termina una connessione al server MySQL.

Sintassi

```
CALL mysql.rds_kill(processID);
```

Parametri

processID

L'identità del thread di connessione da terminare.

Note per l'utilizzo

Ogni connessione al server MySQL viene eseguita in un thread distinto. Per terminare una connessione, utilizza la procedura `mysql.rds_kill` e passa l'ID di thread di quella connessione. Per ottenere l'ID di thread, utilizza il comando MySQL SHOW PROCESSLIST.

Esempi

L'esempio seguente termina una connessione con l'ID di thread 4243:

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

Termina una query in esecuzione sul server MySQL.

Sintassi

```
CALL mysql.rds_kill_query(processID);
```

Parametri

processID

L'identità del processo o del thread che esegue la query da terminare.

Note per l'utilizzo

Per arrestare una query in esecuzione nel server MySQL, utilizza la procedura `mysql_rds_kill_query` e invia l'ID di connessione del thread che sta eseguendo la query. La procedura termina quindi la connessione.

Per ottenere l'ID, esegui una query sulla tabella INFORMATION_SCHEMA PROCESSLIST MySQL o utilizza il comando MySQL SHOW PROCESSLIST. Il valore nella colonna ID da `SHOW PROCESSLIST` o `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST` è *processID*.

Esempi

L'esempio seguente arresta una query con l'ID di thread di query 230040:

```
CALL mysql.rds_kill_query(230040);
```

Replica delle transazioni utilizzando GTIDs

Le seguenti stored procedure controllano il modo in cui le transazioni vengono replicate utilizzando gli identificatori di transazione globali (GTIDs) con Aurora My. SQL Per informazioni su come utilizzare la replica basata su GTIDs Aurora SQL My, consulta. Utilizzo della replica GTID basata

Argomenti

- mysql.rds_assign_gtids_to_anonymous_transactions (Aurora My versione 3) SQL
- mysql.rds_gtid_purged (Aurora My versione 3) SQL
- mysql.rds_skip_transaction_with_gtid (Aurora My versione 2 e 3) SQL
- mysql.rds_start_replication_until_gtid (Aurora La mia versione 3) SQL

mysql.rds_assign_gtids_to_anonymous_transactions (Aurora My versione 3) SQL

Configura l'opzione `ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS` dell'istruzione `CHANGE REPLICATION SOURCE TO`. Fa sì che il canale di replica assegni un GTID alle transazioni replicate che non ne hanno una. In questo modo, è possibile eseguire la replica binaria dei log da un'origine che non utilizza la replica GTID basata su una replica che la utilizza. Per ulteriori informazioni, vedere CHANGEREPLICATIONSOURCETO Statement and Replication From a Source Without GTIDs to a Replica With GTIDs nel My Reference Manual. SQL

Sintassi

```
CALL mysql.rds_assign_gtids_to_anonymous_transactions(gtid_option);
```

Parametri

`gtid_option`

Valore di stringa. I valori consentiti sono `OFFLOCAL`, o sono specificati. UUID

Note per l'utilizzo

Questa procedura ha lo stesso effetto del rilascio della dichiarazione `CHANGE REPLICATION SOURCE TO ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS =` *`gtid_option`* nella comunità MySQL.

GTIDdeve essere attivato `ON` per essere *`gtid_option`* impostato su `LOCAL` o in modo specificoUUID.

Il valore predefinito è `OFF`, il che significa che la funzione non viene utilizzata.

`LOCAL`assegna unaGTID, inclusa quella della replica UUID (l'`server_uuid`impostazione).

Il passaggio di un parametro che è un UUID assegna un valore GTID che include quanto specificatoUUID, ad esempio l'`server_uuid`impostazione per il server di origine della replica.

Esempi

Per disattivare questa funzione:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('OFF');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: OFF |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Per utilizzare quello proprio della replica: UUID

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('LOCAL');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: LOCAL |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Per utilizzare uno specificoUUID:

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('317a4760-
f3dd-3b74-8e45-0615ed29de0e');
+-----------------------------------------------------------------------------------------------
+
| Message |
+-----------------------------------------------------------------------------------------------
+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: 317a4760-
f3dd-3b74-8e45-0615ed29de0e |
+-----------------------------------------------------------------------------------------------
+
```

```
1 row in set (0.07 sec)
```

mysql.rds_gtid_purged (Aurora My versione 3) SQL

Imposta il valore globale della variabile di sistema su un determinato set `gtid_purged` di identificatori di transazione globali (). GTID La variabile di `gtid_purged` sistema è un GTID set composto da tutte le transazioni che sono state eseguite sul server, ma che non esistono in alcun file di registro binario sul server. GTIDs

Per consentire la compatibilità con My SQL 8.0, esistono due modi per impostare il valore di`gtid_purged`:

- Sostituisci il valore di `gtid_purged` con il GTID set specificato.
- Aggiunge il GTID set specificato al GTID set che `gtid_purged` già contiene.

Sintassi

Per sostituire il valore di `gtid_purged` con il GTID set specificato:

```
CALL mysql.rds_gtid_purged (gtid_set);
```

Per aggiungere il valore di `gtid_purged` al GTID set specificato:

```
CALL mysql.rds_gtid_purged (+gtid_set);
```

Parametri

`gtid_set`

Il valore di *`gtid_set`* deve essere un sovrainsieme del valore corrente di `gtid_purged` e non può intersecarsi con. `gtid_subtract(gtid_executed,gtid_purged)` Cioè, il nuovo GTID set deve includere quelli già presenti GTIDs `gtid_purged` e non può includere quelli `gtid_executed` che non sono ancora GTIDs stati eliminati. Inoltre, il *`gtid_set`* parametro non può includere GTIDs quelle incluse nel `gtid_owned` set globale, vale a dire quelle GTIDs relative alle transazioni attualmente in corso di elaborazione sul server.

Note per l'utilizzo

La procedura `mysql.rds_gtid_purged` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora My SQL versione 3.04 e successive.

Esempi

L'esempio seguente assegna la GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23` alla variabile globale. `gtid_purged`

```
CALL mysql.rds_gtid_purged('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_skip_transaction_with_gtid (Aurora My versione 2 e 3) SQL

Salta la replica di una transazione con l'identificatore di transazione globale specificato (GTID) su un'istanza primaria Aurora.

È possibile utilizzare questa procedura per il disaster recovery quando è noto che una GTID transazione specifica causa un problema. Usa questa stored procedure per saltare la transazione problematica. Esempi di transazioni problematiche includono le transazioni che disabilitano la replica, eliminano dati importanti o con le quali l'istanza database diventa non disponibile.

Sintassi

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

Parametri

`gtid_to_skip`

La transazione GTID di replica da saltare.

Note per l'utilizzo

La procedura `mysql.rds_skip_transaction_with_gtid` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora My SQL versione 2 e 3.

Esempi

L'esempio seguente salta la replica della transazione con. GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid (Aurora La mia versione 3) SQL

Avvia la replica da un cluster instance e interrompe la SQL replica immediatamente dopo l'identificatore di transazione globale specificato (). GTID

Sintassi

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

Parametri

gtid

GTIDDopodiché la replica deve interrompersi.

Note per l'utilizzo

La procedura `mysql.rds_start_replication_until_gtid` deve essere eseguita dall'utente master.

Questa procedura è supportata per Aurora My SQL versione 3.04 e successive.

La `mysql.rds_start_replication_until_gtid` stored procedure non è supportata per la replica gestita, che include quanto segue:

- Replica di cluster Amazon Aurora MySQL DB su più cluster Regioni AWS
- Migrazione di dati da un'istanza RDS for My SQL DB a un cluster Amazon Aurora SQL My DB utilizzando una replica di lettura Aurora

Quando il parametro `gtid` specifica una transazione che è già stata eseguita dalla replica, la procedura viene arrestata immediatamente.

Esempi

L'esempio seguente avvia la replica e replica le modifiche fino a quando non raggiunge il risultato. GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

Rotazione dei log delle interrogazioni

Le seguenti procedure memorizzate ruotano I miei SQL log in tabelle di backup. Per ulteriori informazioni, consulta File di log del database Aurora MySQL.

Argomenti

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

Converte la tabella `mysql.general_log` in una tabella di backup.

Sintassi

```
CALL mysql.rds_rotate_general_log;
```

Note per l'utilizzo

Puoi convertire la tabella `mysql.general_log` in una tabella di backup chiamando la procedura `mysql.rds_rotate_general_log`. Quando le tabelle di log sono convertite, la tabella di log corrente è copiata in una tabella di logo di backup e le voci nella tabella di log corrente sono eliminate. Se una tabella di log di backup esiste, viene eliminata prima che la tabella di log corrente sia copiata nel backup. Puoi eseguire una query sulla tabella di log di backup, se necessario. La tabella di log di backup per la tabella `mysql.general_log` è denominata `mysql.general_log_backup`.

È possibile eseguire questa procedura solo quando il parametro `log_output` è impostato su `TABLE`.

mysql.rds_rotate_slow_log

Converte la tabella `mysql.slow_log` in una tabella di backup.

Sintassi

```
CALL mysql.rds_rotate_slow_log;
```

Note per l'utilizzo

Puoi convertire la tabella `mysql.slow_log` in una tabella di backup chiamando la procedura `mysql.rds_rotate_slow_log`. Quando le tabelle di log sono convertite, la tabella di log corrente è copiata in una tabella di logo di backup e le voci nella tabella di log corrente sono eliminate. Se una tabella di log di backup esiste, viene eliminata prima che la tabella di log corrente sia copiata nel backup.

Puoi eseguire una query sulla tabella di log di backup, se necessario. La tabella di log di backup per la tabella `mysql.slow_log` è denominata `mysql.slow_log_backup`.

Impostazione e visualizzazione della configurazione del registro binario

Le seguenti stored procedure impostano e mostrano i parametri di configurazione, ad esempio per la conservazione dei file di log binari.

Argomenti

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

Specifica il numero di ore di conservazione dei log binari o il numero di secondi di ritardo della replica.

Sintassi

```
CALL mysql.rds_set_configuration(name,value);
```

Parametri

name

Il nome del parametro di configurazione da impostare.

value

Il valore del parametro di configurazione.

Note per l'utilizzo

la procedura archiviata `mysql.rds_set_configuration` supporta i parametri di configurazione seguenti:

- binlog retention hours

I parametri di configurazione vengono archiviati in modo permanente e restano effettivi dopo qualsiasi riavvio o failover dell'istanza database.

binlog retention hours

Il parametro `binlog retention hours` viene utilizzato per specificare il numero di ore di conservazione dei file di log binari. Amazon Aurora normalmente elimina un log binario il prima possibile, ma il log binario potrebbe essere ancora necessario per la replica con un database My SQL esterno ad Aurora.

Il valore predefinito di `binlog retention hours` è `NULL`. Per Aurora MySQL, `NULL` significa che i log binari vengono ripuliti pigramente. Aurora I miei registri SQL binari potrebbero rimanere nel sistema per un certo periodo, che di solito non è più lungo di un giorno.

Per specificare il numero di ore per mantenere i log binari in un cluster di database, usa la stored procedure `mysql.rds_set_configuration` e specifica un periodo con tempo sufficiente per l'esecuzione della replica, come mostrato nell'esempio seguente.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> Non puoi utilizzare il valore `0` per `binlog retention hours`.

Per i cluster DB Aurora My SQL versione 2.11.0 e successive e la versione 3 DB, il `binlog retention hours` valore massimo è 2160 (90 giorni).

Dopo l'impostazione del periodo di retention, monitora l'utilizzo dello storage per l'istanza database per verificare che i log binari conservati non occupino troppo spazio di storage.

mysql.rds_show_configuration

Il numero di ore di retention dei log binari.

Sintassi

```
CALL mysql.rds_show_configuration;
```

Note per l'utilizzo

Per verificare il numero di ore in cui Amazon RDS conserva i log binari, utilizza la `mysql.rds_show_configuration` stored procedure.

Esempi

L'esempio seguente visualizza il periodo di retention:

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

Aurora My: tabelle information_schema SQL specifiche

Aurora My SQL dispone di alcune `information_schema` tabelle specifiche per Aurora.

information_schema.aurora_global_db_instance_status

La tabella `information_schema.aurora_global_db_instance_status` contiene informazioni sullo stato di tutte le istanze database nei cluster database primario e secondario di un database globale. Nella tabella seguente vengono visualizzate le colonne che è possibile utilizzare. Le colonne rimanenti sono solo per uso interno di Aurora.

> **Note**
>
> Questa tabella dello schema di informazioni è disponibile solo con i database globali Aurora My SQL 3.04.0 e versioni successive.

Colonna	Tipo di dati	Descrizione
SERVER_ID	varchar(100)	Identificatore istanza database.
SESSION_ID	varchar(100)	Un identificatore univoco per la sessione corrente. Il valore di `MASTER_SESSION_ID` identifica l'istanza database di lettura (primaria).
AWS_REGION	varchar(100)	L'ambiente Regione AWS in cui viene eseguita questa

Colonna	Tipo di dati	Descrizione
		istanza di database globale. Per l'elenco delle regioni, consulta Disponibilità nelle regioni.
DURABLE_LSN	bigint unsigned	Il numero di sequenza di registro (LSN) è reso durevole nell'archiviazione. Un numero di sequenza di registro (LSN) è un numero sequenziale univoco che identifica un record nel registro delle transazioni del database. LSNssono ordinati in modo tale che un LSN valore più grande rappresenti una transazione successiva.
HIGHEST_LSN_RCVD	bigint unsigned	Il valore massimo LSN ricevuto dall'istanza DB dall'istanza Writer DB.
OLDEST_ READ _ VIEW _ TRX _ID	bigint unsigned	L'ID della transazione più vecchia che può essere rimossa dall'istanza database di scrittura.
OLDEST_READ_VIEW_LSN	bigint unsigned	Il più vecchio LSN usato dall'istanza DB per leggere dallo storage.

Colonna	Tipo di dati	Descrizione
VISIBILITY_ LAG _IN_ MSEC	float(10,0) unsigned	Per le istanze di lettura nel cluster database primario, il ritardo in millisecondi di questa istanza database rispetto all'istanza database di scrittura . Per istanze di lettura in un cluster database secondario, il ritardo in millisecondi di questa istanza database rispetto al volume secondario.

information_schema.aurora_global_db_status

La `information_schema.aurora_global_db_status` tabella contiene informazioni su vari aspetti del ritardo globale del database Aurora, in particolare il ritardo dello storage Aurora sottostante (il cosiddetto ritardo di durabilità) e il ritardo tra l'obiettivo del punto di ripristino (). RPO Nella tabella seguente vengono visualizzate le colonne che è possibile utilizzare. Le colonne rimanenti sono solo per uso interno di Aurora.

> **Note**
>
> Questa tabella dello schema di informazioni è disponibile solo con i database globali Aurora My SQL 3.04.0 e versioni successive.

Colonna	Tipo di dati	Descrizione
AWS_REGION	varchar(100)	Regione AWS In cui viene eseguita questa istanza di database globale. Per l'elenco delle regioni, consulta Disponibilità nelle regioni.
HIGHEST_LSN_WRITTEN	bigint unsigned	Il numero di sequenza di log più alto (LSN) attualmente

Colonna	Tipo di dati	Descrizione
		esistente in questo cluster DB. Un numero di sequenza di registro (LSN) è un numero sequenziale univoco che identifica un record nel registro delle transazioni del database. LSNssono ordinati in modo tale che un LSN valore più grande rappresenti una transazione successiva.
DURABILITY_ LAG _IN_ MILLISECONDS	float(10,0) unsigned	La differenza nei valori di timestamp tra il parametro `HIGHEST_LSN_WRITTEN` su un cluster database secondario e il parametro `HIGHEST_LSN_WRITTEN` sul cluster database primario. Questo valore è sempre 0 sul cluster database primario del database globale Aurora.

Colonna	Tipo di dati	Descrizione
RPO_ _IT_ LAG MILLISECONDS	float(10,0) unsigned	Il ritardo dell'obiettivo del punto di ripristino (RPO). Il RPO ritardo è il tempo necessario per COMMIT archiviare la transazione utente più recente su un cluster DB secondario dopo che è stata archiviata nel cluster DB primario del database globale Aurora. Questo valore è sempre 0 sul cluster database primario del database globale Aurora. In termini semplici, questa metrica calcola l'obiettivo del punto di ripristino per ogni cluster Aurora My SQL DB nel database globale Aurora, ovvero la quantità di dati che potrebbero andare persi in caso di interruzione. Come nel caso del ritardo, viene misurato in termini di tempo. RPO

Colonna	Tipo di dati	Descrizione
LAST_LAG_CALCULATION_TIMESTAMP	datetime	Il timestamp che specifica quando sono stati calcolati i valori per `DURABILITY_LAG_IN_MILLISECONDS` e `RPO_LAG_IN_MILLISECONDS`. Il valore temporale `1970-01-01 00:00:00+00` indica che questo è il cluster di database primario.
OLDEST_ _ READ VIEW _ _ID TRX	bigint unsigned	L'ID della transazione più vecchia che può essere rimossa dall'istanza database di scrittura.

information_schema.replica_host_status

La tabella `information_schema.replica_host_status` contiene informazioni sulla replica. Le colonne che è possibile utilizzare sono mostrate nella tabella seguente. Le colonne rimanenti sono solo per uso interno di Aurora.

Colonna	Tipo di dati	Descrizione
CPU	double	La CPU percentuale di utilizzo dell'host di replica.
È_ CURRENT	tinyint	Se la replica è aggiornata.
LAST_UPDATE_TIMESTAMP	datetime(6)	L'ora dell'ultimo aggiornamento. Utilizzata per determinare se un record è obsoleto.
REPLICA_ _IN_ LAG MILLISECONDS	double	Il ritardo di replica in millisecondi.

Colonna	Tipo di dati	Descrizione
SERVER_ID	varchar(100)	L'ID del server di database.
SESSION_ID	varchar(100)	L'ID della sessione di database. Utilizzato per determinare se un'istanz a database è un'istanza di scrittura o di lettura.

> **Note**
>
> Quando un'istanza di replica rimane indietro, le informazioni interrogate dalla relativa tabelle `information_schema.replica_host_status` potrebbero essere obsolete. In questo caso, ti consigliamo di eseguire una query dall'istanza si scrittura.
> Sebbene la tabella `mysql.ro_replica_status` contenga informazioni simili, ti consigliamo di non utilizzarla.

information_schema.aurora_forwarding_processlist

La tabella `information_schema.aurora_forwarding_processlist` contiene informazioni sui processi coinvolti nell'inoltro di scrittura.

I contenuti di questa tabella sono visibili solo sull'istanza database di scrittura per un cluster database con l'inoltro di scrittura globale o interno al cluster attivato. Viene restituito un set di risultati vuoto sulle istanze database di lettura.

Campo	Tipo di dati	Descrizione
ID	bigint	L'identificatore della connessione sull'istanza DB di scrittura . Questo identificatore è lo stesso valore visualizzato nella colonna `Id`dell'istruzione `SHOW PROCESSLIST` e restituita dalla funzione `CONNECTION_ID()` all'interno del thread.

Campo	Tipo di dati	Descrizione
USER	varchar(32)	L'SQLutente My che ha rilasciato la dichiarazione.
HOST	varchar(255)	Il SQL cliente My che ha rilasciato la dichiarazione. Per le istruzioni inoltrate, questo campo mostra l'indirizzo host del client dell'applicazione che ha stabilito la connessione sull'istanza database di lettura di inoltro.
DB	varchar(64)	Il database predefinito per il thread.
COMMAND	varchar(16)	Il tipo di comando eseguito dal thread per conto del client o `Sleep` se lo stato della sessione è inattivo. Per le descrizioni dei comandi thread, consulta la mia SQL documentazione sui valori dei comandi Thread nella SQL documentazione personale.
TIME	int	Il tempo in secondi in cui il thread è rimasto nello stato corrente.
STATE	varchar(64)	Un'azione, un evento o uno stato che indica cosa sta facendo il thread. Per le descrizioni dei valori di stato, consulta General Thread States nella SQL documentazione My.
INFO	longtext	L'istruzione eseguita dal thread o `NULL` se non sta eseguendo un'istruzione. L'istruzione può essere quella inviata al server o un'istruzione più interna se l' istruzione esegue altre istruzioni.
IS_ FORWARDED	bigint	Indica se il thread viene inoltrato da un'istanza database di lettura.

Campo	Tipo di dati	Descrizione
REPLICA_ _ID SESSION	bigint	L'identificatore di connessione sulla replica di Aurora. Questo identificatore è lo stesso valore visualizzato nella colonna `Id`dell'istruzione `SHOW PROCESSLIST` sull'istanza database di lettura Aurora di inoltro.
REPLICA_I NSTANCE_I DENTIFIER	varchar(6 4)	L'identificatore dell'istanza database del thread di inoltro.
REPLICA_C LUSTER_NAME	varchar(6 4)	L'identificatore cluster database del thread di inoltro. Per l'inoltro di scrittura all'interno del cluster, questo identific atore è lo stesso cluster database dell'istanza database di scrittura.
REPLICA_REGION	varchar(6 4)	La fonte Regione AWS da cui proviene il thread di inoltro. Per l'inoltro di scrittura all'interno del cluster, questa regione è la stessa Regione AWS dell'istanza database di scrittura.

Aggiornamenti del motore di database per Amazon Aurora My SQL

Gli aggiornamenti di Amazon Aurora vengono resi disponibili periodicamente. Gli aggiornamenti vengono applicati ai cluster di database di Aurora durante le finestre di manutenzione del sistema. I tempi per l'applicazione degli aggiornamenti dipendono dalla regione e dall'impostazione della finestra di manutenzione del cluster del DB, ma anche dal tipo di aggiornamento.

Le versioni di Amazon Aurora vengono rese disponibili per tutte le AWS regioni nel corso di più giorni. Alcune Regioni possono mostrare temporaneamente una versione del motore che non è ancora disponibile in una Regione diversa.

Gli aggiornamenti vengono applicati a tutte le istanze in un cluster di database contemporaneamente. Un aggiornamento richiede un riavvio del database per tutte le istanze di un cluster di database, pertanto c'è un tempo di inattività di 20-30 secondi, al termine del quale si potranno ricominciare a usare i cluster di database. Puoi visualizzare o modificare le impostazioni della finestra di manutenzione dalla AWS Management Console.

Per informazioni dettagliate sulle SQL versioni di Aurora My supportate da Amazon Aurora, consulta le note di rilascio per Aurora My. SQL

Di seguito, puoi imparare come scegliere la SQL versione di Aurora My giusta per il tuo cluster, come specificare la versione quando crei o aggiorni un cluster e le procedure per aggiornare un cluster da una versione all'altra con interruzioni minime.

Argomenti

- Verifica dei numeri di versione di Aurora My SQL
- Supporto a lungo termine (LTS) e versioni beta per Amazon Aurora My SQL
- Preparazione per la fine del supporto standard di Amazon Aurora My SQL -Compatible Edition versione 2
- Preparazione alla fine del ciclo di vita di Amazon Aurora My SQL -Compatible Edition versione 1
- Aggiornamento dei cluster Amazon Aurora My DB SQL
- Aggiornamenti e correzioni del motore di database per Amazon Aurora MySQL

Verifica dei numeri di versione di Aurora My SQL

Sebbene Aurora My SQL -Compatible Edition sia compatibile con i motori di SQL database My, Aurora My SQL include funzionalità e correzioni di bug specifiche per particolari versioni di Aurora

My. SQL Gli sviluppatori di applicazioni possono controllare la SQL versione di Aurora My nelle loro applicazioni utilizzando. SQL Gli amministratori del database possono controllare e specificare le versioni di Aurora SQL My durante la creazione o l'aggiornamento di cluster e istanze DB di SQL Aurora My DB.

Argomenti

- Verifica o specifica delle versioni del motore Aurora SQL My tramite AWS
- Verifica delle versioni di Aurora My utilizzando SQL SQL

Verifica o specifica delle versioni del motore Aurora SQL My tramite AWS

Quando si eseguono attività amministrative utilizzando, o AWS Management Console AWS CLI RDSAPI, si specifica la SQL versione Aurora My in un formato alfanumerico descrittivo.

A partire da Aurora My SQL version 2, le versioni del motore Aurora hanno la seguente sintassi.

```
mysql-major-version.mysql_aurora.aurora-mysql-version
```

La parte *mysql-major-version-* è `5.7` o `8.0`. Questo valore rappresenta la versione del protocollo client e il livello generale del supporto delle SQL funzionalità My per la versione Aurora My SQL corrispondente.

*aurora-mysql-version*È un valore punteggiato composto da tre parti: la versione principale di Aurora SQL My, la versione secondaria di Aurora SQL My e il livello di patch. La versione principale è `2` o `3`. Questi valori rappresentano Aurora My SQL compatibile con My SQL 5.7 o 8.0, rispettivamente. La versione secondaria rappresenta la versione di funzionalità della serie 2.x o 3.x. Il livello di patch inizia da `0` per ogni versione secondaria e rappresenta l'insieme successivo di correzioni di bug che si applicano alla versione secondaria. Occasionalmente, una nuova funzionalità viene incorporata in una versione secondaria ma non resa immediatamente visibile. In questi casi, la funzionalità è sottoposta a ottimizzazione e viene resa pubblica in un livello di patch successivo.

Tutte le versioni del motore 2.x Aurora SQL My sono compatibili via cavo con Community SQL My 5.7.12 o versioni successive. Tutte le versioni 3.x Aurora SQL My engine sono compatibili via cavo con SQL My 8.0.23 o versioni successive. Puoi fare riferimento alle note di rilascio della versione 3.x specifica per trovare la versione compatibile con My corrispondente. SQL

Ad esempio, le versioni del motore per Aurora My SQL 3.04.0 e 2.11.2 sono le seguenti.

```
8.0.mysql_aurora.3.04.0
```

```
5.7.mysql_aurora.2.11.2
```

> **Note**
>
> Non c'è one-to-one corrispondenza tra SQL le versioni My della community e le versioni di Aurora SQL My 2.x. Per Aurora My SQL versione 3, esiste una mappatura più diretta. Per verificare quali correzioni di bug e nuove funzionalità sono presenti in una particolare versione di Aurora SQL My, consulta Aggiornamenti del motore di database per Amazon Aurora SQL My versione 3 e Aggiornamenti del motore di database per Amazon Aurora SQL My versione 2 nelle note di rilascio per Aurora My. SQL Per l'elenco cronologico delle nuove funzionalità e versioni, consulta Cronologia dei documenti. Per verificare la versione minima richiesta per una correzione relativa alla sicurezza, consulta Vulnerabilità di sicurezza corrette in Aurora My nelle Note di rilascio per Aurora My SQL. SQL

La versione del SQL motore Aurora My viene specificata in alcuni AWS CLI comandi e RDS API operazioni. Ad esempio, si specifica l'`--engine-version`opzione quando si eseguono i AWS CLI comandi create-db-clustere modify-db-cluster. Il `EngineVersion` parametro viene specificato quando si eseguono le RDS API operazioni C reateDBCluster e odifyDBClusterM.

In Aurora My SQL versione 2 e successive, la versione con motore include AWS Management Console anche la versione Aurora. L'aggiornamento del cluster modifica il valore visualizzato. Questa modifica consente di specificare e controllare le SQL versioni precise di Aurora My, senza la necessità di connettersi al cluster o eseguire alcun SQL comando.

> **Tip**
>
> Per i cluster Aurora gestiti tramite AWS CloudFormation, questa modifica dell'`EngineVersion`impostazione può attivare azioni tramite. AWS CloudFormationPer informazioni su come AWS CloudFormation gestisce le modifiche all'`EngineVersion`impostazione, consulta la documentazione. AWS CloudFormation

Verifica delle versioni di Aurora My utilizzando SQL SQL

I numeri di versione di Aurora che è possibile recuperare nell'applicazione utilizzando le SQL query utilizzano il formato. *`<major version>`*`.`*`<minor version>`*`.`*`<patch version>`* È possibile ottenere questo numero di versione per qualsiasi istanza DB nel SQL cluster Aurora My interrogando

la AURORA_VERSION variabile di sistema. Per ottenere questo numero di versione puoi utilizzare una delle seguenti query.

```
select aurora_version();
select @@aurora_version;
```

Le query producono un output simile al seguente.

```
mysql> select aurora_version(), @@aurora_version;
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

I numeri di versione che la console e RDS API restituiscono utilizzando le tecniche descritte in Verifica o specifica delle versioni del motore Aurora SQL My tramite AWS sono in genere più descrittivi. CLI

Supporto a lungo termine (LTS) e versioni beta per Amazon Aurora My SQL

Aurora My SQL offre supporto a lungo termine (LTS) e versioni beta per alcune versioni del motore Aurora My. SQL

Argomenti

- Aurora I miei rilasci di supporto SQL a lungo termine () LTS
- Versioni beta di Aurora My SQL

Aurora I miei rilasci di supporto SQL a lungo termine () LTS

Ogni nuova SQL versione di Aurora My rimane disponibile per un determinato periodo di tempo e può essere utilizzata quando si crea o si aggiorna un cluster DB. Trascorso questo periodo, devi aggiornare i cluster che utilizzano questa versione. Puoi aggiornare manualmente il cluster prima della fine del periodo di supporto, oppure Aurora può aggiornarlo automaticamente per te quando la sua SQL versione Aurora My non è più supportata.

Aurora designa alcune versioni di Aurora My come SQL versioni con supporto a lungo termine (). LTS I cluster DB che utilizzano LTS release possono rimanere sulla stessa versione più a lungo e subire

meno cicli di aggiornamento rispetto ai cluster che utilizzano versioni non rilasciate. LTS Aurora supporta ogni LTS versione per almeno tre anni dopo la sua disponibilità. Quando è necessario eseguire l'aggiornamento di un cluster DB che è in una LTS versione, Aurora lo aggiorna alla versione successiva. LTS In questo modo, l'intervallo di tempo prima del nuovo aggiornamento del cluster diventa più lungo.

Durante il periodo di validità di una SQL LTS versione di Aurora My, nuovi livelli di patch introducono correzioni a problemi importanti. Questi livelli di patch non comprendono nuove funzionalità. È possibile scegliere se applicare tali patch ai cluster DB che eseguono la versione. LTS Per alcune correzioni critiche, Amazon potrebbe eseguire un aggiornamento gestito a un livello di patch all'interno della stessa LTS versione. Tali aggiornamenti gestiti vengono eseguiti automaticamente nella finestra di manutenzione del cluster.

Ti consigliamo di eseguire l'aggiornamento alla versione più recente, anziché utilizzare la LTS versione, per la maggior parte dei tuoi cluster Aurora MySQL. In questo modo, puoi sfruttare Aurora anche come servizio gestito e puoi accedere alle ultime funzionalità e correzioni di bug. Le LTS versioni sono destinate ai cluster con le seguenti caratteristiche:

- Non puoi permetterti tempi di inattività della tua applicazione Aurora SQL My per gli aggiornamenti oltre a casi rari per le patch critiche.
- Il ciclo di test per il cluster e le applicazioni associate richiede molto tempo per ogni aggiornamento del motore di SQL database Aurora My.
- La versione del database per il tuo SQL cluster Aurora My include tutte le funzionalità del motore DB e le correzioni di bug di cui l'applicazione ha bisogno.

L'attuale LTS versione di Aurora My SQL è la seguente:

- Aurora La mia SQL versione 3.04.*. Per ulteriori dettagli sulla LTS versione, consulta Aggiornamenti del motore di database per Amazon Aurora My SQL versione 3 nelle note di rilascio per Aurora My. SQL

> **Note**
>
> Ti consigliamo di disabilitare gli aggiornamenti automatici delle versioni secondarie per le versioni. LTS Impostare il `AutoMinorVersionUpgrade` parametro su `false` o deselezionare la casella di controllo Abilita l'aggiornamento automatico della versione secondaria sul AWS Management Console.

Se non lo disabiliti, il cluster DB potrebbe essere aggiornato a una LTS versione diversa come 3.05.2.

Versioni beta di Aurora My SQL

Una SQL versione beta di Aurora My è una correzione di sicurezza anticipata, disponibile solo in un numero limitato di Regioni AWS. Queste correzioni vengono successivamente distribuite in modo più ampio in tutte le regioni con la prossima versione della patch.

La numerazione di una versione beta è simile a quella di una versione secondaria di Aurora SQL My, ma con una quarta cifra aggiuntiva, ad esempio 2.12.0.1 o 3.05.0.1.

Per ulteriori informazioni, consulta Aggiornamenti del motore di database per Amazon Aurora My SQL versione 2 e Aggiornamenti del motore di database per Amazon Aurora SQL My versione 3 nelle Note di rilascio per Aurora My. SQL

Preparazione per la fine del supporto standard di Amazon Aurora My SQL - Compatible Edition versione 2

La versione 2 di Amazon Aurora My SQL -Compatible Edition (con compatibilità con My SQL 5.7) dovrebbe raggiungere la fine del supporto standard il 31 ottobre 2024. Si consiglia di aggiornare tutti i cluster che eseguono Aurora SQL My versione 2 alla versione predefinita Aurora My 3 (con compatibilità con SQL My 8.0) o superiore prima che Aurora SQL SQL My versione 2 raggiunga la fine del periodo di supporto standard. Il 31 ottobre 2024, Amazon RDS registrerà automaticamente i tuoi database in Amazon Extended RDS Support. Se utilizzi Amazon Aurora My SQL versione 2 (con compatibilità con My SQL 5.7) in un Aurora Serverless cluster versione 1, questo non si applica a te. Se vuoi aggiornare il tuo Aurora Serverless la versione 1 si raggruppa in Aurora SQL La mia versione 3, vedi. Percorso di aggiornamento per Aurora Serverless v1 Cluster database

Puoi trovare end-of-support le date imminenti per le versioni SQL principali di Aurora My nel calendario delle versioni principali di Aurora My. SQL

Se disponi di cluster che eseguono Aurora SQL My versione 2, riceverai avvisi periodici con le informazioni più recenti su come eseguire un aggiornamento man mano che ci avviciniamo alla fine della data di supporto standard. Aggiorneremo periodicamente questa pagina con le informazioni più recenti.

Fine della tempistica di assistenza standard

1. Da oggi fino al 31 ottobre 2024: puoi aggiornare i cluster da Aurora My SQL versione 2 (con compatibilità My SQL 5.7) ad Aurora My SQL versione 3 (con compatibilità My 8.0). SQL
2. 31 ottobre 2024 — In questa data, Aurora SQL My versione 2 raggiungerà la fine del supporto standard e Amazon registrerà automaticamente i tuoi cluster in RDS Amazon Extended Support. RDS

Ti iscriveremo automaticamente all'RDSExtended Support. Per ulteriori informazioni, consulta Supporto esteso di Amazon RDS con .

Individuazione dei cluster interessati da questo processo end-of-life

Per trovare i cluster interessati da questo end-of-life processo, utilizzare le seguenti procedure.

> **Important**
>
> Assicurati di eseguire queste istruzioni in ogni Regione AWS luogo in Account AWS cui si trovano le tue risorse.

Console

Per trovare un cluster Aurora My SQL versione 2

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Nella casella Filtra per database, inserisci 5.7.
4. Cerca Aurora My SQL nella colonna del motore.

AWS CLI

Per trovare i cluster interessati da questo end-of-life processo utilizzando il AWS CLI, chiamate il describe-db-clusterscomando. È possibile utilizzare il seguente script di esempio.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?(Engine==`aurora-
mysql` && contains(EngineVersion,`5.7.mysql_aurora`))].{EngineVersion:EngineVersion,
 DBClusterIdentifier:DBClusterIdentifier, EngineMode:EngineMode}' --output table
 --region us-east-1

       +-----------------------------------------------------------------+
       |                       DescribeDBClusters                        |
       +---------------------+---------------+-------------------------+
       |         DBCI        |      EM       |          EV             |
       +---------------------+---------------+-------------------------+
       |    aurora-mysql2    |  provisioned  | 5.7.mysql_aurora.2.11.3 |
       | aurora-serverlessv1 |   serverless  | 5.7.mysql_aurora.2.11.3 |
       +---------------------+---------------+-------------------------+
```

RDS API

Per trovare i cluster Aurora My SQL DB che eseguono Aurora My SQL versione 2, usa l'escribeDBClustersAPIoperazione RDS D con i seguenti parametri obbligatori:

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nome
      - engine
    - Values.Value.N
      - ['aurora']

Amazon RDS Extended Support

Puoi utilizzare Amazon RDS Extended Support tramite la community My SQL 5.7 gratuitamente fino alla data di fine del supporto, il 31 ottobre 2024. Il 31 ottobre 2024, Amazon registra RDS automaticamente i tuoi database in Extended RDS Support for Aurora My versione 2. SQL RDS Extended Support for Aurora è un servizio a pagamento che fornisce fino a 28 mesi aggiuntivi di supporto per Aurora My SQL version 2 fino alla fine dell'Extended RDS Support nel febbraio 2027. RDS L'Extended Support sarà offerto solo per le versioni SQL secondarie di Aurora My 2.11 e 2.12. Per utilizzare Amazon Aurora My SQL versione 2 dopo la fine del supporto standard, pianifica di eseguire i database su una di queste versioni minori prima del 31 ottobre 2024.

Per ulteriori informazioni su RDS Extended Support, ad esempio costi e altre considerazioni, vedereSupporto esteso di Amazon RDS con .

Esecuzione di un aggiornamento

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole. Consideriamo l'aggiornamento come un processo in tre fasi, con attività da svolgere prima, per e dopo l'aggiornamento.

Prima dell'aggiornamento:

Prima dell'aggiornamento, si consiglia di verificare la compatibilità, le prestazioni, le procedure di manutenzione delle applicazioni nel cluster aggiornato e altre considerazioni simili, al fine di confermare che, dopo l'aggiornamento, le applicazioni funzionino come previsto. I cinque consigli seguenti ti aiuteranno a ottenere un'esperienza di aggiornamento migliore.

- Innanzitutto, è fondamentale comprendereCome funziona l'aggiornamento della versione principale di Aurora MySQL in loco.
- Successivamente, esplora le tecniche di aggiornamento disponibili quandoAggiornamento da Aurora MySQL versione 2 alla versione 3.
- Per aiutarti a decidere il momento e l'approccio giusti per l'aggiornamento, puoi scoprire le differenze tra Aurora My SQL versione 3 e il tuo ambiente attuale con. Confronto tra Aurora My SQL versione 2 e Aurora My versione 3 SQL
- Dopo aver deciso l'opzione più comoda e più efficace, prova un aggiornamento simulato in loco su un cluster clonato, utilizzando. Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL
- Rivedere le Controlli preliminari di aggiornamento delle versioni principali per Aurora My SQL. L'upgrade prechecker può essere eseguito e determinare se il database può essere aggiornato correttamente e se ci sono problemi di incompatibilità delle applicazioni dopo l'aggiornamento, nonché considerazioni relative alle prestazioni, alle procedure di manutenzione e simili.
- Non tutti i tipi o versioni di Aurora My SQL cluster possono utilizzare il meccanismo di aggiornamento sul posto. Per ulteriori informazioni, consulta Procedure di aggiornamento a una versione principale di Aurora MySQL.

In caso di domande o dubbi, il team di AWS supporto è disponibile sui forum della community e su Premium Support.

Esecuzione dell'aggiornamento:

È possibile utilizzare una delle seguenti tecniche di aggiornamento. La quantità di tempi di inattività del sistema dipende dalla tecnica scelta.

- Implementazioni blu/verdi: per le situazioni in cui la massima priorità è ridurre i tempi di inattività delle applicazioni, puoi utilizzare Amazon RDS Blue/Green Deployments per eseguire l'aggiornamento della versione principale nei cluster Amazon Aurora DB forniti. Un'implementazione blu/verde crea un ambiente di gestione temporanea che copia l'ambiente di produzione. È possibile apportare alcune modifiche al cluster di database Aurora nell'ambiente verde (gestione temporanea) senza influire sui carichi di lavoro di produzione. Lo switchover richiede in genere meno di un minuto e non comporta perdita di dati. Per ulteriori informazioni, consulta Panoramica delle implementazioni di Amazon Aurora Blue/Green. Questa tecnica riduce al minimo i tempi di inattività, ma comporta l'utilizzo di risorse aggiuntive durante l'esecuzione dell'aggiornamento.
- Aggiornamenti sul posto: puoi eseguire un aggiornamento sul posto in cui Aurora esegue automaticamente un processo di precontrollo per te, mette offline il cluster, esegue il backup del cluster, esegue l'aggiornamento e riporta il cluster online. Un aggiornamento della versione principale sul posto può essere eseguito in pochi clic e non comporta altri coordinamenti o failover con altri cluster, ma comporta tempi di inattività. Per ulteriori informazioni, consulta Come eseguire un aggiornamento in loco
- Ripristino istantanee: è possibile aggiornare il cluster Aurora SQL My versione 2 eseguendo il ripristino da uno snapshot di Aurora My versione 2 in un cluster Aurora SQL My versione 3. SQL A tale scopo, devi seguire la procedura di acquisizione di uno snapshot ed eseguire il ripristino a partire da esso. Questo processo comporta l'interruzione del database perché si esegue il ripristino da uno snapshot.

Dopo l'aggiornamento:

Dopo l'aggiornamento, è necessario monitorare attentamente il sistema (applicazione e database) e apportare modifiche di precisione, se necessario. Seguendo attentamente le fasi di pre-aggiornamento, si ridurranno al minimo le modifiche necessarie. Per ulteriori informazioni, consulta Risoluzione dei problemi relativi alle prestazioni del database Amazon Aurora My SQL.

Per ulteriori informazioni sui metodi, la pianificazione, i test e la risoluzione dei problemi degli aggiornamenti della versione SQL principale di Aurora My, assicurati di leggere Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL attentamente, tra cui.

Risoluzione dei problemi relativi all'aggiornamento in-place di Aurora My SQL Inoltre, tieni presente che alcuni tipi di istanza non sono supportati per Aurora My SQL versione 3. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.

Percorso di aggiornamento per Aurora Serverless v1 Cluster database

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole. Consideriamo l'aggiornamento come un processo in tre fasi, con attività da svolgere prima, per e dopo l'aggiornamento.

Aurora My SQL versione 2 (con compatibilità con My SQL 5.7) continuerà a ricevere il supporto standard per Aurora Serverless v1 i cluster.

Se desideri eseguire l'aggiornamento ad Amazon Aurora My SQL 3 (con compatibilità con My SQL 8.0) e continuare a funzionare Aurora Serverless, puoi usare Amazon Aurora Serverless v2. Per comprendere le differenze tra Aurora Serverless v1 e Aurora Serverless v2, consulta Confronto di Aurora Serverless v2 e Aurora Serverless v1.

Esegui l'upgrade a Aurora Serverless v2: È possibile aggiornare un Aurora Serverless v1 raggruppare a Aurora Serverless v2. Per ulteriori informazioni, vedereAggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2.

Preparazione alla fine del ciclo di vita di Amazon Aurora My SQL - Compatible Edition versione 1

La versione 1 di Amazon Aurora My SQL -Compatible Edition (con compatibilità con My SQL 5.6) dovrebbe raggiungere la fine del ciclo di vita il 28 febbraio 2023. Amazon consiglia di aggiornare tutti i cluster (forniti e) che Aurora Serverless eseguono Aurora My versione SQL 1 ad Aurora My versione 2 (con compatibilità My 5.7) o Aurora My SQL versione 3 (con SQL compatibilità My 8.0). SQL SQL Esegui questa operazione prima che Aurora My SQL version 1 raggiunga la fine del periodo di supporto.

Per i cluster DB con provisioning Aurora, è possibile completare gli aggiornamenti da Aurora My SQL versione 1 ad Aurora My versione 2 con diversi metodi. SQL Le istruzioni per il meccanismo di aggiornamento in loco sono disponibili in Come eseguire un aggiornamento in loco. Un altro modo per completare l'aggiornamento consiste nello scattare un'istantanea di un cluster Aurora SQL My versione 1 e ripristinare l'istantanea in un cluster SQL Aurora My versione 2. Oppure è possibile seguire un processo in più fasi che esegue i cluster vecchi e nuovi affiancati. Per ulteriori dettagli su

ciascun metodo, consulta Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL.

Per i cluster Aurora Serverless v1 DB, è possibile eseguire un aggiornamento sul posto da Aurora My versione SQL 1 ad Aurora My versione 2. SQL Per informazioni dettagliate su questo metodo, consulta Modificare un Aurora Serverless v1 cluster di database

Per i cluster DB con provisioning Aurora, è possibile completare gli aggiornamenti da Aurora My SQL versione 1 ad SQL Aurora My versione 3 utilizzando un processo di aggiornamento in due fasi:

1. Esegui l'aggiornamento da Aurora My SQL versione 1 ad Aurora My SQL versione 2 utilizzando i metodi descritti in precedenza.
2. Esegui l'aggiornamento da Aurora My SQL versione 2 ad Aurora My SQL versione 3 utilizzando gli stessi metodi utilizzati per l'aggiornamento dalla versione 1 alla versione 2. Per ulteriori dettagli, consulta Aggiornamento da Aurora MySQL versione 2 alla versione 3. Prendi nota di Differenze di funzionalità tra Aurora My SQL versione 2 e 3.

Puoi trovare end-of-life le prossime date per le versioni principali di Aurora in. Versioni di Amazon Aurora Amazon aggiorna automaticamente tutti i cluster che non hai aggiornato tu stesso prima della end-of-life data. Dopo la end-of-life data, questi aggiornamenti automatici alla versione principale successiva avvengono durante una finestra di manutenzione programmata per i cluster.

Di seguito sono riportate altre tappe fondamentali per l'aggiornamento dei cluster Aurora SQL My versione 1 (forniti Aurora Serverless e) che stanno raggiungendo la fine del ciclo di vita. Per ciascuno di essi, l'ora di inizio è 00:00 Universal Coordinated Time (). UTC

1. Da ora fino al 28 febbraio 2023: puoi avviare in qualsiasi momento gli aggiornamenti dei cluster Aurora SQL My versione 1 (con compatibilità SQL My 5.6) ad Aurora My versione 2 (con compatibilità SQL My 5.7). SQL Da Aurora My SQL versione 2, puoi eseguire un ulteriore aggiornamento ad Aurora My SQL versione 3 (con SQL compatibilità My 8.0) per i cluster DB forniti da Aurora.
2. 16 gennaio 2023 — Dopo questo periodo, non è possibile creare nuovi cluster o istanze Aurora My SQL versione 1 da o da (). AWS Management Console AWS Command Line Interface AWS CLI Inoltre, non è possibile aggiungere nuove regioni secondarie a un database globale Aurora. Ciò potrebbe influire sulla capacità di recupero da un'interruzione non pianificata, come descritto in Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata, perché dopo questo periodo non è possibile completare i passaggi 5 e 6. Inoltre, non sarà possibile creare una

nuova replica di lettura interregionale con Aurora My versione 1. SQL Puoi ancora fare quanto segue per i cluster Aurora My SQL versione 1 esistenti fino al 28 febbraio 2023:

- Ripristina un'istantanea scattata di un cluster Aurora SQL My versione 1 alla stessa versione del cluster di snapshot originale.
- Aggiunta di repliche di lettura (non applicabile ai cluster di database Aurora Serverless).
- Modificare la configurazione dell'istanza.
- Eseguire il ripristino. point-in-time
- Creare cloni di cluster della versione 1 esistenti.
- Crea una nuova replica di lettura interregionale con Aurora My SQL versione 2 o successiva.

3. 28 febbraio 2023 — Trascorso questo periodo, prevediamo di aggiornare automaticamente i cluster Aurora My SQL versione 1 alla versione predefinita di Aurora My SQL versione 2 entro una finestra di manutenzione pianificata che segue. Il ripristino delle istantanee DB di Aurora SQL My versione 1 comporta un aggiornamento automatico del cluster ripristinato alla versione predefinita di Aurora SQL My versione 2 in quel momento.

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole.

Per le situazioni in cui la massima priorità è ridurre i tempi di inattività, puoi utilizzare le implementazioni blu/verdi per eseguire l'aggiornamento della versione principale nei cluster database Amazon Aurora allocati. Un'implementazione blu/verde crea un ambiente di gestione temporanea che copia l'ambiente di produzione. È possibile apportare modifiche al cluster database Aurora nell'ambiente verde (gestione temporanea) senza influire sui carichi di lavoro di produzione. Lo switchover richiede in genere meno di un minuto senza perdita di dati e senza la necessità di modificare l'applicazione. Per ulteriori informazioni, consulta Panoramica delle implementazioni di Amazon Aurora Blue/Green.

Al termine dell'aggiornamento, è possibile che sia necessario compiere ulteriori operazioni.
Ad esempio, potrebbe essere necessario eseguire il follow-up a causa delle differenze di SQL compatibilità, del funzionamento di alcune funzionalità SQL relative a My o delle impostazioni dei parametri tra la vecchia e la nuova versione.

Per saperne di più sui metodi, la pianificazione, i test e la risoluzione dei problemi degli aggiornamenti della versione SQL principale di Aurora My, assicurati di leggere attentamente. Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL

Individuazione dei cluster interessati da questo processo end-of-life

Per trovare i cluster interessati da questo end-of-life processo, utilizzare le seguenti procedure.

> **Important**
>
> Assicurati di eseguire queste istruzioni in ogni Regione AWS luogo in Account AWS cui si trovano le tue risorse.

Console

Per trovare un cluster Aurora My SQL versione 1

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Nella casella Filter by databases (Filtra per database), inserisci 5.6.
4. Cerca Aurora My SQL nella colonna del motore.

AWS CLI

Per trovare i cluster interessati da questo end-of-life processo utilizzando il AWS CLI, chiamate il describe-db-clusterscomando. È possibile utilizzare il seguente script di esempio.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?Engine==`aurora`].
{EV:EngineVersion, DBCI:DBClusterIdentifier, EM:EngineMode}' --output table --region
 us-east-1

        +-----------------------------------------+
        |            DescribeDBClusters           |
        +---------------+-------------+-----------+
        |     DBCI      |     EM      |    EV     |
        +---------------+-------------+-----------+
        |  my-database-1|  serverless |  5.6.10a  |
        +---------------+-------------+-----------+
```

RDS API

Per trovare i cluster Aurora My SQL DB che eseguono Aurora My SQL versione 1, usa l'escribeDBClustersAPIoperazione RDS D con i seguenti parametri obbligatori:

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nome
      - engine
    - Values.Value.N
      - ['aurora']

Aggiornamento dei cluster Amazon Aurora My DB SQL

È possibile aggiornare un cluster Aurora My SQL DB per ottenere correzioni di bug, nuove SQL funzionalità di Aurora My o passare a una versione completamente nuova del motore di database sottostante. Nelle sezioni seguenti viene illustrato come.

> **Note**
>
> Il tipo di aggiornamento che esegui dipende dalla quantità di tempo di inattività che puoi permetterti per il tuo cluster, dalla quantità di test di verifica che intendi fare, dall'importanza delle correzioni di bug specifiche o delle nuove funzionalità per il tuo caso d'uso e dalla previsione di eseguire frequenti piccoli aggiornamenti o aggiornamenti occasionali che saltano diversi versioni intermedie. Per ogni aggiornamento, è possibile modificare la versione principale, la versione secondaria e il livello di patch per il cluster. Se non conosci la distinzione tra le versioni principali, le versioni secondarie e i livelli di patch di Aurora SQL My, puoi leggere le informazioni di base all'indirizzo. Verifica dei numeri di versione di Aurora My SQL

> **Tip**
>
> È possibile ridurre al minimo i tempi di inattività necessari per l'aggiornamento di un cluster di database utilizzando un'implementazione blu/verde. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Argomenti

- Aggiornamento della versione secondaria o del livello di patch di un cluster Aurora My DB SQL
- Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL

Aggiornamento della versione secondaria o del livello di patch di un cluster Aurora My DB SQL

Puoi utilizzare i seguenti metodi per aggiornare la versione secondaria di un cluster di database o per applicare le patch a un cluster di database:

- Aggiornamento di Aurora MySQL modificando la versione del motore(per Aurora My SQL versione 2 e 3)
- Abilitazione degli aggiornamenti automatici tra versioni minori di Aurora My SQL

Per informazioni su come l'applicazione di patch senza tempi di inattività può ridurre le interruzioni durante il processo di aggiornamento, consulta Utilizzo dell'applicazione di patch senza tempi di inattività.

Per informazioni sull'esecuzione di un aggiornamento di versione minore per il cluster Aurora My SQL DB, consulta i seguenti argomenti.

Argomenti

- Prima di eseguire un aggiornamento di una versione secondaria
- Controlli preliminari di aggiornamento delle versioni minori per Aurora My SQL
- Aggiornamento di Aurora MySQL modificando la versione del motore
- Abilitazione degli aggiornamenti automatici tra versioni minori di Aurora My SQL
- Utilizzo dell'applicazione di patch senza tempi di inattività
- Tecnica alternativa di aggiornamento blu/verde

Prima di eseguire un aggiornamento di una versione secondaria

Si consiglia di eseguire le seguenti azioni per ridurre i tempi di inattività durante l'aggiornamento di una versione secondaria:

- La manutenzione del cluster Aurora DB deve essere eseguita durante un periodo di traffico ridotto. Usa Performance Insights per identificare questi periodi di tempo al fine di configurare

correttamente le finestre di manutenzione. Per ulteriori informazioni su Performance Insights, consulta Monitoraggio del carico del DB con Performance Insights su Amazon RDS. Per ulteriori informazioni sulla finestra di manutenzione del cluster DB,Impostazione della finestra di manutenzione preferita del cluster database.

- Utilizzo AWS SDKsche supportano il backoff e il jitter esponenziali come best practice. Per ulteriori informazioni, consulta Exponential Backoff And Jitter.

Controlli preliminari di aggiornamento delle versioni minori per Aurora My SQL

Quando avvii un aggiornamento di una versione minore, Amazon Aurora esegue automaticamente i controlli preliminari.

Questi controlli preliminari sono obbligatori. Non puoi scegliere di saltarli. I controlli preliminari offrono i seguenti vantaggi:

- Ti consentono di evitare tempi di inattività non pianificati durante l'aggiornamento.
- In caso di incompatibilità, Amazon Aurora impedisce l'aggiornamento e ti fornisce un registro per conoscerle. Puoi quindi utilizzare il log per preparare il database per l'aggiornamento riducendo le incompatibilità. Per informazioni dettagliate sulla rimozione delle incompatibilità, vedere Preparazione dell'installazione per l'aggiornamento nella documentazione personale. SQL

I controlli preliminari vengono eseguiti prima dell'arresto dell'istanza database per l'aggiornamento, il che significa che non generano alcun tempo di inattività durante l'esecuzione. Se i precontrolli rilevano un'incompatibilità, Aurora annulla automaticamente l'aggiornamento prima che l'istanza DB venga interrotta. Aurora genera anche un evento per l'incompatibilità. Per ulteriori informazioni sugli eventi di Amazon Aurora, consulta. Utilizzo delle notifiche di RDS eventi di Amazon

Aurora registra informazioni dettagliate su ogni incompatibilità nel file di registro. `PrePatchCompatibility.log` Nella maggior parte dei casi, la voce di registro include un collegamento alla SQL documentazione My per correggere l'incompatibilità. Per ulteriori informazioni sulla visualizzazione dei file di log, consultare Visualizzazione ed elenco dei file di log del database.

A causa della natura dei controlli preliminari, questi analizzano gli oggetti nel database. Questa analisi comporta il consumo di risorse e incrementa il tempo di completamento dell'aggiornamento.

Aggiornamento di Aurora MySQL modificando la versione del motore

L'aggiornamento della versione secondaria di un cluster Aurora MySQL DB applica correzioni aggiuntive e nuove funzionalità a un cluster esistente.

Questo tipo di aggiornamento si applica ai cluster Aurora MySQL in cui la versione originale e la versione aggiornata hanno entrambe la stessa versione principale di Aurora MySQL, 2 o 3. Il processo è semplice e veloce perché non comporta alcuna conversione dei metadati di Aurora MySQL o la riorganizzazione dei dati della tabella.

È possibile eseguire questo tipo di aggiornamento modificando la versione del motore del cluster DB utilizzando, o l'API RDS. AWS Management Console AWS CLI Ad esempio, se sul cluster è in esecuzione Aurora MySQL 3.x, scegli una versione 3.x superiore.

Se stai eseguendo un aggiornamento minore su un Aurora Global Database, aggiorna tutti i cluster secondari prima di aggiornare il cluster primario.

> **Note**
>
> Per eseguire un aggiornamento di una versione secondaria a Aurora MySQL versione 3.04.* o successiva o versione 2.12.*, utilizza la seguente procedura:
>
> 1. Rimuovi tutte le regioni secondarie dal cluster globale. Seguire la procedura riportata in Rimozione di un cluster da un database globale Amazon Aurora.
> 2. Aggiorna la versione del motore della regione principale alla versione 3.04.* o successiva o alla versione 2.12.*, a seconda dei casi. Seguire la procedura riportata in To modify the engine version of a DB cluster.
> 3. Aggiungi le regioni secondarie al cluster globale. Seguire la procedura riportata in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Per modificare la versione del motore di un cluster di database

- Utilizzando la console – Modificare le proprietà del cluster. Nella finestra Modify DB cluster (Modifica il cluster di database) cambia la versione del motore Aurora MySQL nella casella DB engine version (Versione motore database). Se non hai familiarità con la procedura generale per la modifica di un cluster, segui le istruzioni riportate in Modifica del cluster di database tramite la console, la CLI e l'API.
- Utilizzando AWS CLI — Richiamate il modify-db-cluster AWS CLI comando e specificate il nome del cluster DB per l'`--db-cluster-identifier`opzione e la versione del motore per l'opzione. `--engine-version`

Ad esempio, per eseguire l'aggiornamento alla versione 3.04.1 di Aurora MySQL, imposta l'opzione su. `--engine-version 8.0.mysql_aurora.3.04.1` Specifica l'opzione `--apply-immediately` per aggiornare immediatamente la versione del motore per il cluster di database.

- Utilizzando l'API RDS: richiama l'operazione Modify DBCluster API e specifica il nome del cluster DB per il `DBClusterIdentifier` parametro e la versione del motore per il parametro. `EngineVersion` Imposta il parametro `ApplyImmediately` su `true` per aggiornare immediatamente la versione del motore per il cluster di database.

Abilitazione degli aggiornamenti automatici tra versioni minori di Aurora My SQL

Per un cluster Amazon Aurora My SQL DB, puoi specificare che Aurora aggiorni automaticamente il cluster DB a nuove versioni minori. Puoi farlo impostando la `AutoMinorVersionUpgrade` proprietà (Aggiornamento automatico della versione secondaria) nel AWS Management Console) del cluster DB.

Gli aggiornamenti automatici vengono eseguiti durante le finestre di manutenzione. Se le singole istanze database nel cluster database hanno finestre di manutenzione diverse dalla finestra di manutenzione del cluster, la finestra di manutenzione del cluster ha la precedenza.

L'aggiornamento automatico della versione secondaria non si applica ai seguenti tipi di cluster Aurora MySQL:

- Cluster che fanno parte di un database globale Aurora
- Cluster con repliche tra regioni

La durata dell'interruzione varia a seconda del carico di lavoro, della dimensione del cluster, della quantità di dati di registro binario e se Aurora può utilizzare la funzionalità zero-downtime patching (). ZDP Aurora riavvia il cluster di database, pertanto potrebbe verificarsi un breve periodo di indisponibilità prima di riprendere l'utilizzo del cluster. In particolare, la quantità di dati dei log binari influisce sui tempi di ripristino. L'istanza database elabora i dati del log binario durante il ripristino. Pertanto un volume elevato di dati dei log binari aumenta il tempo di recupero.

> **Note**
>
> Aurora esegue gli aggiornamenti automatici solo se tutte le istanze DB nel cluster DB hanno l'impostazione abilitata. `AutoMinorVersionUpgrade` Per informazioni su come

impostarla e su come funziona quando viene applicata a livello di cluster e istanza, consulta. Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora
Quindi, se esiste un percorso di aggiornamento per le istanze del cluster DB a una versione secondaria del motore DB `AutoUpgrade` impostata su true, l'aggiornamento avrà luogo. L'`AutoUpgrade`impostazione è dinamica ed è impostata daRDS.
Gli aggiornamenti automatici delle versioni secondarie vengono eseguiti sulla versione secondaria predefinita.

Puoi usare un CLI comando come il seguente per controllare lo stato dell'`AutoMinorVersionUpgrade`impostazione per tutte le istanze DB nei tuoi cluster Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Questo comando genera un output simile al seguente:

```
[
  {
      "DBInstanceIdentifier": "db-t2-medium-instance",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": true
  },
  {
      "DBInstanceIdentifier": "db-t2-small-original-size",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "instance-2020-05-01-2332",
      "DBClusterIdentifier": "cluster-57-2020-05-01-4615",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

In questo esempio, l'impostazione Abilita aggiornamento automatico versione secondaria è disattivata per il cluster di database `cluster-57-2020-06-03-6411`, perché è disattivata per una delle istanze database del cluster.

Utilizzo dell'applicazione di patch senza tempi di inattività

L'esecuzione degli aggiornamenti per i cluster Aurora SQL My DB comporta la possibilità di un'interruzione quando il database viene chiuso e durante l'aggiornamento. Per impostazione predefinita, se si avvia l'aggiornamento mentre il database è occupato, si perdono tutte le connessioni e le transazioni elaborate dal cluster di database. Se si aspetta che il database sia inattivo per eseguire l'aggiornamento, potrebbe essere necessario attendere a lungo.

La funzionalità zero-downtime patching (ZDP) tenta, nel migliore dei modi, di preservare le connessioni client tramite un aggiornamento Aurora My. SQL Se viene ZDP completata correttamente, le sessioni dell'applicazione vengono conservate e il motore di database si riavvia mentre è in corso l'aggiornamento. Il riavvio del motore del database può causare un calo del throughput della durata di alcuni secondi fino a circa un minuto.

ZDPnon si applica a quanto segue:

- Patch e aggiornamenti del sistema operativo
- Aggiornamenti di una versione principale

ZDPè disponibile per tutte le SQL versioni di Aurora My supportate e le classi di istanze DB.

ZDPnon è supportato per Aurora Serverless v1 o database globali Aurora.

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Utilizzo delle classi di istanza T per lo sviluppo e i test.

Puoi visualizzare le metriche degli attributi importanti ZDP nella sezione Il mio log degli SQL errori. È inoltre possibile visualizzare informazioni su quando Aurora My SQL utilizza ZDP o sceglie di non utilizzare nella ZDP pagina Eventi di. AWS Management Console

In Aurora My, Aurora può eseguire una patch SQL senza downtime indipendentemente dal fatto che la replica dei log binari sia abilitata o meno. Se la replica dei log binari è abilitata, Aurora SQL My interrompe automaticamente la connessione alla destinazione binlog durante un'operazione. ZDP Aurora My si riconnette SQL automaticamente alla destinazione binlog e riprende la replica al termine del riavvio.

ZDPfunziona anche in combinazione con i miglioramenti al riavvio di Aurora My. SQL L'applicazione di patch all'istanza database scrittore applica automaticamente le patch ai lettori contemporaneamente. Dopo aver applicato la patch, Aurora ripristina le connessioni sia sulle istanze DB dello scrittore che del lettore.

ZDPpotrebbe non essere completato correttamente nelle seguenti condizioni:

- Durante l'esecuzione di query o transazioni di lunga durata. Se Aurora è ZDP in grado di eseguire questa operazione, tutte le transazioni aperte vengono annullate ma le relative connessioni vengono mantenute.
- Vengono utilizzate tabelle temporanee, blocchi utente o blocchi tabellari, ad esempio durante l'esecuzione delle istruzioni Data Definition Language (). DDL Aurora interrompe queste connessioni.
- Esistono modifiche ai parametri in attesa.

Se non è ZDP disponibile una finestra temporale adeguata per l'esecuzione a causa di una o più di queste condizioni, l'applicazione delle patch torna al comportamento standard.

Sebbene le connessioni rimangano intatte dopo un'ZDPoperazione riuscita, alcune variabili e funzionalità vengono reinizializzate. I seguenti tipi di informazioni non vengono conservati tramite un riavvio causato da zero-downtime patching:

- Variabili globali. Aurora ripristina le variabili di sessione, ma non ripristina le variabili globali dopo il riavvio.
- Variabili di stato. In particolare, il valore di operatività riportato dallo stato del motore viene ripristinato dopo un riavvio che utilizza i meccanismi or. ZDR ZDP
- `LAST_INSERT_ID`.
- Stato `auto_increment` in memoria per le tabelle. Lo stato di incremento automatico in memoria viene reinizializzato. Per ulteriori informazioni sui valori di incremento automatico, vedere My SQL Reference Manual.
- Informazioni diagnostiche dalle tabelle `INFORMATION_SCHEMA` e `PERFORMANCE_SCHEMA`. Queste informazioni diagnostiche vengono visualizzate anche nell'output di comandi come `SHOW PROFILE` e `SHOW PROFILES`.

Le seguenti attività relative a zero-downtime restart sono riportate nella pagina Eventi:

- Tentativo di aggiornare il database senza tempi di inattività.
- Tentativo di aggiornare il database senza tempi di inattività terminato. L'evento riporta quanto tempo è durato il processo. L'evento segnala inoltre quante connessioni sono state mantenute durante il riavvio e quante connessioni sono state interrotte. È possibile consultare il registro degli errori del database per visualizzare ulteriori dettagli su ciò che è successo durante il riavvio.

Tecnica alternativa di aggiornamento blu/verde

In alcune situazioni, la priorità principale è eseguire il passaggio immediato dal cluster precedente a quello aggiornato. In tali situazioni, è possibile utilizzare un processo in più fasi che esegue i cluster vecchi e nuovi. side-by-side Qui, i dati vengono replicati dal cluster precedente a quello nuovo fino a quando il nuovo cluster non prende il controllo. Per informazioni dettagliate, consultare Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Aggiornamento della versione principale di un cluster di database Amazon Aurora MySQL

In un numero di versione di Aurora MySQL come 3.04.1, il 3 rappresenta la versione principale. Aurora MySQL versione 2 è compatibile con MySQL 5.7. Aurora MySQL versione 3 è compatibile con MySQL 8.0.

L'aggiornamento tra le versioni principali richiede una pianificazione e test più estesi rispetto all'aggiornamento a una versione secondaria. Il processo può richiedere un tempo considerevole. Al termine dell'aggiornamento, è possibile che sia necessario compiere ulteriori operazioni. Ad esempio, ciò potrebbe verificarsi a causa di differenze nella compatibilità SQL o della modalità di funzionamento di alcune caratteristiche correlate a MySQL. Oppure potrebbe verificarsi a causa delle diverse impostazioni dei parametri tra la vecchia e la nuova versione.

Indice

- Aggiornamento da Aurora MySQL versione 2 alla versione 3
- Procedure di aggiornamento a una versione principale di Aurora MySQL
- Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco
- Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL
  - Simulazione dell'aggiornamento mediante clonazione del cluster DB
  - Distribuzioni blu/verdi
- Controlli preliminari di aggiornamento delle versioni principali per Aurora My SQL

- Procedura di precontrollo per Aurora My SQL
- Formato di registro di Precheck per Aurora My SQL
- Esempi di output del log di Precheck per Aurora My SQL
- Verifica preliminare le prestazioni di Aurora My SQL
- Riepilogo dei controlli preliminari di Community My upgrade SQL
- Riepilogo dei controlli preliminari di Aurora My upgrade SQL
- Riferimento alle descrizioni di precontrollo per Aurora MySQL
  - Errori
    - Precontrolli MySQL che segnalano errori
    - Precontrolli Aurora MySQL che segnalano errori
  - Avvisi
    - Precontrolli MySQL che segnalano avvisi
    - Aurora MySQL precontrolli che segnalano gli avvisi
  - Note
  - Errori, avvertenze o avvisi
- Come eseguire un aggiornamento in loco
  - Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster
  - Modifiche delle proprietà del cluster tra versioni di Aurora MySQL
  - Principali aggiornamenti in loco per database globali
  - Aggiornamenti sul posto per cluster DB con repliche di lettura interregionali
- Aurora Il mio tutorial di aggiornamento SQL sul posto
- Individuazione dei motivi degli errori di aggiornamento di Aurora My SQL major version
- Risoluzione dei problemi relativi all'aggiornamento in-place di Aurora My SQL
- Pulizia post-aggiornamento per Aurora My versione 3 SQL
  - Indici spaziali

Aggiornamento da Aurora MySQL versione 2 alla versione 3

Se si dispone di un cluster compatibile con MySQL 5.7 e si desidera aggiornarlo a un cluster compatibile con MySQL 8.0, è possibile eseguire un processo di aggiornamento sul cluster stesso. Questo tipo di aggiornamento è un aggiornamento in loco, a differenza degli aggiornamenti che si eseguono creando un nuovo cluster. Questa tecnica mantiene lo stesso endpoint e altre

caratteristiche del cluster. L'aggiornamento è relativamente veloce perché non richiede la copia di tutti i dati in un nuovo volume cluster. Questa stabilità consente di ridurre al minimo eventuali modifiche di configurazione nelle applicazioni. Inoltre, consente di ridurre la quantità di test per il cluster aggiornato. Questo perché il numero di istanze database e le relative classi di istanza rimangono identiche.

Il meccanismo di aggiornamento utilizzato comporta l'arresto del cluster di database durante l'operazione. Aurora esegue un arresto pulito e completa le operazioni in sospeso, come il rollback delle transazioni e l'eliminazione degli annullamenti. Per ulteriori informazioni, consulta Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco.

Il metodo di aggiornamento locale è conveniente, in quanto è semplice da eseguire e riduce al minimo le modifiche alla configurazione delle applicazioni associate. Ad esempio, un aggiornamento in loco conserva gli endpoint e il set di istanze database per il cluster. Tuttavia, il tempo necessario per un aggiornamento in loco può variare a seconda delle proprietà dello schema e dell'attività del cluster. Pertanto, a seconda delle esigenze del cluster, è possibile scegliere tra le tecniche di aggiornamento:

- Aggiornamento locale
- Implementazione blu/verde
- Ripristino delle istantanee

> **Note**
>
> Se si utilizza l'API AWS CLI o RDS per il metodo di aggiornamento del ripristino delle istantanee, è necessario eseguire un'operazione successiva per creare un'istanza DB di scrittura nel cluster DB ripristinato.

Per informazioni generali su Aurora MySQL versione 3 e sulle nuove funzionalità, consultare Aurora My SQL versione 3 compatibile con My 8.0 SQL.

Per informazioni dettagliate sulla pianificazione di un aggiornamento, consultare Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL e Come eseguire un aggiornamento in loco.

Procedure di aggiornamento a una versione principale di Aurora MySQL

Non tutti i tipi o le versioni di cluster Aurora MySQL possono utilizzare il meccanismo di aggiornamento in loco. È possibile conoscere la procedura di aggiornamento appropriata per ogni cluster Aurora MySQL consultando la tabella seguente.

Tipo di cluster di database Aurora MySQL	Può utilizzare l'aggiornamento in loco?	Azione
Cluster con provisioning MySQL Aurora, versione 2	Sì	L'aggiornamento sul posto è supportato per i cluster Aurora MySQL compatibili con MySQL 5.7. Per informazioni sull'aggiornamento ad Aurora MySQL versione 3, consulta Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL e Come eseguire un aggiornamento in loco.
Cluster con provisioning MySQL Aurora, versione 3	Non applicabile	Utilizza una procedura di aggiornamento della versione secondaria per eseguire l'aggiornamento tra le versioni di Aurora MySQL versione 3.
Aurora Serverless v1 cluster	Non applicabile	Aurora Serverless v1 è supportato per Aurora MySQL solo nella versione 2.
Aurora Serverless v2 cluster	Non applicabile	Aurora Serverless v2 è supportato per Aurora MySQL solo nella versione 3.
Cluster in un database globale Aurora	Sì	Per eseguire l'aggiornamento di Aurora MySQL versione 2 alla versione 3, segui la procedura per eseguire un aggiornamento locale per i cluster in un database globale Aurora. Esegui l'aggiornamento sul cluster globale. Aurora esegue l'aggiornamento del cluster primario e di

Tipo di cluster di database Aurora MySQL	Può utilizzare l'aggiornamento in loco?	Azione
		tutti i cluster secondari nel database globale nello stesso momento. Se utilizzi l'API AWS CLI o RDS, chiama il `modify-global-cluster` comando o l'`ModifyGlobalCluster` operazione anziché o. `modify-db-cluster` `ModifyDBCluster` È possibile eseguire un aggiornamento sul posto da Aurora MySQL versione 2 alla versione 3 solo `lower_case_table_names` se il parametro è impostato come predefinito e si riavvia il database globale. Per ulteriori informazioni, consulta Aggiornamenti di una versione principale.
Cluster di query parallele	Sì	È possibile eseguire l'aggiornamento locale.
Cluster che costituisce la destinazione della replica del log binario	Probabile	Se la replica del log binario proviene da un cluster Aurora MySQL, è possibile eseguire un aggiornamento locale. Non è possibile eseguire l'aggiornamento se la replica del log binario proviene da un'istanza MySQL RDS o da un'istanza database MySQL on-premise. In tal caso, è possibile eseguire l'aggiornamento utilizzando il meccanismo di ripristino degli snapshot.

Tipo di cluster di database Aurora MySQL	Può utilizzare l'aggiornamento in loco?	Azione
Cluster con zero istanze database	No	Utilizzando AWS CLI o l'API RDS, è possibile creare un cluster Aurora MySQL senza istanze DB collegate. Allo stesso modo, è inoltre possibile rimuovere tutte le istanze database da un cluster Aurora MySQL lasciando intatti i dati nel volume del cluster. Sebbene il cluster non abbia istanze database collegate, non è possibile eseguire l'aggiornamento in loco. Il meccanismo di aggiornamento richiede un'istanza di scrittura nel cluster per eseguire conversioni sulle tabelle di sistema, sui file di dati e così via. In questo caso, utilizza l'API AWS CLI o l'API RDS per creare un'istanz a writer per il cluster. Dopodiché è possibile eseguire l'aggiornamento in loco.
Cluster con backtrack abilitato	Sì	È possibile eseguire un aggiornamento sul posto per un cluster Aurora MySQL che utilizza la funzional ità Backtrack. Tuttavia, dopo l'aggiornamento, non è possibile eseguire il backtrack del cluster a un momento precedente l'aggiornamento.

Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco

Aurora MySQL esegue l'aggiornamento della versione principale come processo in più fasi. È possibile controllare lo stato corrente di un aggiornamento. Alcuni passaggi dell'aggiornamento forniscono anche informazioni sullo stato di avanzamento. All'inizio di ciascuna fase, Aurora MySQL registra un evento. È possibile esaminare gli eventi man mano che si verificano nella pagina Events (Eventi) della console RDS. Per maggiori informazioni sull'utilizzo degli eventi, consulta Utilizzo delle notifiche di RDS eventi di Amazon.

> **Important**
>
> Una volta avviato, il processo viene eseguito fino a quando l'aggiornamento va a buon fine o non riesce. Non è possibile annullare l'aggiornamento mentre è in corso. Se l'aggiornamento non riesce, Aurora esegue il rollback di tutte le modifiche e il cluster mantiene la versione del motore, i metadati e così via precedenti.

Il processo di aggiornamento è costituito da tre fasi:

1. Aurora esegue una serie di controlli preliminari prima di iniziare il processo di aggiornamento. Il cluster continua a funzionare mentre Aurora esegue questi controlli. Ad esempio, il cluster non può avere transazioni XA nello stato preparato né avere istruzioni DDL (Data Definition Language) in corso di elaborazione. Ad esempio, potrebbe essere necessario arrestare le applicazioni che inviano determinati tipi di istruzioni SQL. Oppure si può semplicemente aspettare fino a quando alcune istruzioni di lunga durata non siano state completate. Dopodiché, si può riprovare a eseguire l'aggiornamento. Alcuni controlli verificano le condizioni che, pur non impedendo l'aggiornamento, potrebbero far sì che richieda molto tempo.

   Se Aurora rileva che le condizioni richieste non sono soddisfatte, modifica le condizioni identificate nei dettagli dell'evento. Segui le indicazioni riportate in Risoluzione dei problemi relativi all'aggiornamento in-place di Aurora My SQL. Se Aurora rileva condizioni che potrebbero rallentare l'aggiornamento, pianifica il monitoraggio dell'aggiornamento per un periodo prolungato.
2. Aurora mette il cluster offline. Quindi Aurora esegue un insieme di test analoghi a quelli della fase precedente, per confermare che non si sono presentati nuovi problemi durante il processo di arresto. Se, a questo punto, Aurora rileva condizioni che impediscono l'aggiornamento, Aurora annulla l'aggiornamento e riporta il cluster in linea. In questo caso, verifica quando le condizioni non sono più applicabili e riavvia l'aggiornamento.
3. Aurora crea uno snapshot del volume del cluster. Si supponga di scoprire problemi di compatibilità o di altro tipo di al termine dell'aggiornamento. Oppure si supponga di volere eseguire test utilizzando sia i cluster originali sia quelli aggiornati. In questi casi, è possibile eseguire il ripristino da questa snapshot per creare un nuovo cluster con la versione originale del motore e i dati originali.

> **Tip**
>
> Questo snapshot è di tipo manuale. Tuttavia, Aurora può crearlo e continuare con il processo di aggiornamento anche se è stata raggiunta la quota per gli snapshot manuali. Questa istantanea rimane permanente (se necessario) finché non viene eliminata. Al termine di tutti i test successivi all'aggiornamento, è possibile eliminare questo snapshot per ridurre al minimo i costi di storage.

4. Aurora clona il volume del cluster. La clonazione è un'operazione veloce che non comporta la copia dei dati effettivi della tabella. Se Aurora riscontra un problema durante l'aggiornamento, ripristina i dati originali dal volume del cluster clonato e riporta il cluster in linea. Il volume temporaneo clonato durante l'aggiornamento non è soggetto al normale limite del numero di cloni per un singolo volume cluster.
5. Aurora esegue un arresto pulito per l'istanza database di scrittura. Durante l'arresto pulito, gli eventi di avanzamento vengono registrati ogni 15 minuti per le operazioni seguenti. È possibile esaminare gli eventi man mano che si verificano nella pagina Events (Eventi) della console RDS.
   - Aurora elimina i record di annullamento per le versioni precedenti delle righe.
   - Aurora esegue il rollback di tutte le transazioni non salvate.
6. Aurora aggiorna la versione del motore sull'istanza database di scrittura:
   - Aurora installa il binario per la nuova versione del motore nell'istanza database di scrittura.
   - Aurora utilizza l'istanza database di scrittura per aggiornare i dati in formato compatibile con MySQL 5.7. Durante questa fase, Aurora modifica le tabelle di sistema ed esegue altre conversioni che influiscono sui dati nel volume del cluster. In particolare, Aurora aggiorna i metadati delle partizioni nelle tabelle di sistema affinché siano compatibili con il formato di partizione MySQL 5.7. Questa fase può richiedere molto tempo se le tabelle del cluster dispongono di un numero elevato di partizioni.

     Se durante questa fase si verificano degli errori, è possibile trovare i dettagli nei log degli errori di MySQL. Dopo l'avvio di questa fase, se il processo di aggiornamento non riesce per qualsiasi motivo, Aurora ripristina i dati originali dal volume cluster clonato.
7. Aurora aggiorna la versione del motore sulle istanze database di lettura.
8. Il processo di aggiornamento è completato. Aurora registra un evento finale per indicare che il processo di aggiornamento è stato completato correttamente. Ora il cluster di database sta eseguendo la nuova versione principale.

Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL

Per aiutarti a decidere il momento e l'approccio giusti per l'aggiornamento, puoi scoprire le differenze tra Aurora MySQL versione 3 e il tuo ambiente attuale:

- Se stai eseguendo la conversione da RDS for MySQL 8.0 o MySQL 8.0 Community Edition, vedi. Confronto tra Aurora MySQL versione 3 e MySQL 8.0 Community Edition
- Se stai eseguendo l'aggiornamento da Aurora MySQL versione 2, RDS for MySQL 5.7 o dalla community MySQL 5.7, consulta. Confronto tra Aurora My SQL versione 2 e Aurora My versione 3 SQL
- Creare nuove versioni compatibili con MySQL 8.0 di qualsiasi gruppo di parametri personalizzati. Applicare i valori dei parametri personalizzati necessari ai nuovi gruppi di parametri. Consultare Modifiche ai parametri per Aurora My versione 3 SQL per conoscere le modifiche ai parametri.
- Esaminare lo schema del database Aurora MySQL versione 2 e le definizioni di oggetto per l'utilizzo di nuove parole chiave riservate introdotte in MySQL 8.0 Community Edition. Eseguire questa operazione prima di effettuare l'aggiornamento. Per ulteriori informazioni, consulta MySQL 8.0 New Keywords and Reserved Words nella documentazione di MySQL.

Puoi inoltre trovare ulteriori suggerimenti e considerazioni sull'aggiornamento specifici di MySQL in Changes in MySQL 8.0 (Modifiche in MySQL 8.0)nel Manuale di riferimento MySQL. Per esempio, è possibile utilizzare il comando `mysqlcheck --check-upgrade` per analizzare i database Aurora MySQL esistenti e identificare potenziali problemi di aggiornamento.

> **Note**
>
> Ti consigliamo di utilizzare classi di istanza database più grandi durante l'aggiornamento ad Aurora MySQL versione 3 utilizzando la tecnica di aggiornamento locale o di ripristino da snapshot. Esempi sono db.r5.24xlarge e db.r6g.16xlarge. Ciò consente di completare il processo di aggiornamento più rapidamente utilizzando la maggior parte della capacità della CPU disponibile sull'istanza database. Al termine dell'aggiornamento della versione principale, puoi passare alla classe di istanza database desiderata.

Dopo aver terminato l'aggiornamento stesso, è possibile seguire le procedure di post-aggiornamento in Pulizia post-aggiornamento per Aurora My versione 3 SQL. Infine, testare la funzionalità e le prestazioni dell'applicazione.

Se stai eseguendo la conversione da RDS da MySQL o dalla community MySQL, segui la procedura di migrazione spiegata in. Migrazione dei dati verso un cluster Amazon Aurora My DB SQL In alcuni casi, è possibile utilizzare la replica del log binario per sincronizzare i dati con un cluster Aurora MySQL versione 3 come parte della migrazione. In tal caso, il sistema di origine deve eseguire una versione compatibile con il cluster DB di destinazione.

Per assicurarsi che le applicazioni e le procedure di amministrazione funzionino correttamente dopo l'aggiornamento di un cluster tra versioni principali, eseguire parte della pianificazione e della preparazione in anticipo. Per vedere quali tipi di codice di gestione aggiornare per AWS CLI gli script o le applicazioni basate sull'API RDS, consulta. Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster Consultare anche Modifiche delle proprietà del cluster tra versioni di Aurora MySQL.

Per informazioni sui problemi che potresti riscontrare durante l'aggiornamento, consulta. Risoluzione dei problemi relativi all'aggiornamento in-place di Aurora My SQL In caso di problemi che potrebbero richiedere un lungo periodo di tempo per completare l'aggiornamento, è possibile verificare tali condizioni in anticipo e correggerle.

> **Note**
>
> Un aggiornamento sul posto comporta la chiusura del cluster DB durante l'operazione. Aurora MySQL esegue un arresto pulito e completa operazioni eccezionali come l'annullamento della cancellazione. Un aggiornamento potrebbe richiedere molto tempo se ci sono molti record di annullamento da eliminare. Si consiglia di eseguire l'aggiornamento solo dopo che la lunghezza dell'elenco della cronologia (HLL) è bassa. Un valore generalmente accettabile per l'HLL è pari o inferiore a 100.000. Per ulteriori informazioni, consulta questo post del blog.

Simulazione dell'aggiornamento mediante clonazione del cluster DB

È possibile verificare la compatibilità dell'applicazione, le prestazioni, le procedure di manutenzione e considerazioni simili per il cluster aggiornato. A questo scopo, è possibile eseguire una simulazione dell'aggiornamento prima di eseguire l'aggiornamento reale. Questa tecnica può essere particolarmente utile per i cluster di produzione. In questo caso, è importante ridurre al minimo i tempi di inattività e disporre del cluster aggiornato pronto all'uso non appena l'aggiornamento è terminato.

Utilizza le fasi seguenti:

1. Crea un clone del cluster originale. Segui la procedura riportata in Clonazione di un volume per un cluster di database Amazon Aurora.
2. Imposta un insieme di istanze database di scrittura e di lettura analogo a quello del cluster originale.
3. Esegui un aggiornamento in loco del cluster clonato. Segui la procedura riportata in Come eseguire un aggiornamento in loco.

   Avvia l'aggiornamento immediatamente dopo aver creato il clone. In questo modo, il volume del cluster è ancora identico allo stato del cluster originale. Se il clone rimane inattivo prima di eseguire l'aggiornamento, Aurora esegue i processi di pulizia del database in background. In tal caso, l'aggiornamento del clone non è una simulazione accurata dell'aggiornamento del cluster originale.
4. Verifica la compatibilità e le prestazioni delle applicazioni, le procedure di amministrazione e così via, utilizzando il cluster clonato.
5. Se riscontri problemi, modifica i tuoi piani di aggiornamento per tenerne conto. Ad esempio, adatta qualsiasi codice dell'applicazione affinché sia compatibile con il set di caratteristiche della versione superiore. Stima il tempo necessario per l'aggiornamento in base alla quantità di dati nel cluster. È inoltre possibile scegliere di pianificare l'aggiornamento per un momento in cui il cluster non è occupato.
6. Dopo avere verificato che le applicazioni e il carico di lavoro funzionano correttamente con il cluster di test, puoi eseguire l'aggiornamento locale per il cluster di produzione.
7. Agisci per ridurre al minimo il tempo di inattività totale del cluster durante un aggiornamento della versione principale. A questo scopo, assicurati che il carico di lavoro sul cluster sia basso o zero al momento dell'aggiornamento. In particolare, assicurati che non vi siano transazioni di lunga durata in corso all'avvio dell'aggiornamento.

Distribuzioni blu/verdi

In alcune situazioni, la priorità principale è eseguire il passaggio immediato dal cluster precedente a quello aggiornato. In tali situazioni, è possibile utilizzare un processo in più fasi che esegue i cluster vecchi e nuovi. side-by-side Qui, i dati vengono replicati dal cluster precedente a quello nuovo fino a quando il nuovo cluster non prende il controllo. Per informazioni dettagliate, consultare Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Controlli preliminari di aggiornamento delle versioni principali per Aurora My SQL

L'aggiornamento di My SQL da una versione principale a un'altra, ad esempio il passaggio da My SQL 5.7 a My SQL 8.0, comporta alcune modifiche architettoniche significative che richiedono un'attenta pianificazione e preparazione. A differenza degli aggiornamenti di versioni minori, in cui l'attenzione si concentra principalmente sull'aggiornamento del software del motore di database e in alcuni casi delle tabelle di sistema, gli SQL aggiornamenti principali di My spesso introducono modifiche fondamentali al modo in cui il database archivia e gestisce i metadati.

Per aiutarvi a identificare tali incompatibilità, durante l'aggiornamento da Aurora SQL My versione 2 alla versione 3, Aurora esegue automaticamente controlli di compatibilità degli aggiornamenti (precontrolli) per esaminare gli oggetti nel cluster di database e identificare le incompatibilità note che possono impedire il proseguimento dell'aggiornamento. Per informazioni dettagliate sui SQL precheck di Aurora My, consulta. Riferimento alle descrizioni di precontrollo per Aurora MySQL
I precontrolli Aurora vengono eseguiti in aggiunta a quelli eseguiti dall'utilità Community My SQL upgrade checker.

Questi controlli preliminari sono obbligatori. Non puoi scegliere di saltarli. I controlli preliminari offrono i seguenti vantaggi:

- Possono ridurre la possibilità di incorrere in errori di aggiornamento che possono portare a tempi di inattività prolungati.
- In caso di incompatibilità, Amazon Aurora impedisce l'esecuzione dell'aggiornamento e ti fornisce un registro per conoscerle. Puoi quindi utilizzare il log per preparare il database per l'aggiornamento alla versione 3 risolvendo le incompatibilità. Per informazioni dettagliate sulla risoluzione delle incompatibilità, vedere Preparazione dell'installazione per l'aggiornamento nella documentazione personale e Aggiornamento a My 8.0? SQL SQL Ecco cosa devi sapere... sul blog My SQL Server.

  Per ulteriori informazioni sull'aggiornamento a My SQL 8.0, consulta Upgrading My nella documentazione My SQL. SQL

I precontrolli vengono eseguiti prima che il cluster DB venga messo offline per l'aggiornamento della versione principale. Se i precontrolli rilevano un'incompatibilità, Aurora annulla automaticamente l'aggiornamento prima che l'istanza DB venga interrotta. Aurora genera anche un evento per l'incompatibilità. Per ulteriori informazioni sugli eventi di Amazon Aurora, consulta. Utilizzo delle notifiche di RDS eventi di Amazon

Una volta completati i precontrolli, Aurora registra informazioni dettagliate su ogni incompatibilità nel file. `upgrade-prechecks.log` Nella maggior parte dei casi, la voce di registro include un collegamento alla SQL documentazione My per correggere l'incompatibilità. Per ulteriori informazioni sulla visualizzazione dei file di log, consultare Visualizzazione ed elenco dei file di log del database.

> **Note**
>
> A causa della natura dei controlli preliminari, questi analizzano gli oggetti nel database. Questa analisi comporta il consumo di risorse e incrementa il tempo di completamento dell'aggiornamento. Per ulteriori informazioni sulle considerazioni relative alle prestazioni del controllo preliminare, vedere. Procedura di precontrollo per Aurora My SQL

Indice

- Procedura di precontrollo per Aurora My SQL
- Formato di registro di Precheck per Aurora My SQL
- Esempi di output del log di Precheck per Aurora My SQL
- Verifica preliminare le prestazioni di Aurora My SQL
- Riepilogo dei controlli preliminari di Community My upgrade SQL
- Riepilogo dei controlli preliminari di Aurora My upgrade SQL
- Riferimento alle descrizioni di precontrollo per Aurora MySQL
  - Errori
    - Precontrolli MySQL che segnalano errori
    - Precontrolli Aurora MySQL che segnalano errori
  - Avvisi
    - Precontrolli MySQL che segnalano avvisi
    - Aurora MySQL precontrolli che segnalano gli avvisi
  - Note
  - Errori, avvertenze o avvisi

Procedura di precontrollo per Aurora My SQL

Come descritto in precedenza, il processo di SQL aggiornamento di Aurora My prevede l'esecuzione di controlli di compatibilità (precontrolli) sul database prima di procedere con l'aggiornamento della versione principale.

Per gli aggiornamenti in loco, i precontrolli vengono eseguiti sull'istanza Writer DB mentre è online. Se il precontrollo ha esito positivo, l'aggiornamento procede. Se vengono rilevati errori, questi vengono registrati nel `upgrade-prechecks.log` file e l'aggiornamento viene annullato. Prima di tentare nuovamente l'aggiornamento, risolvete gli eventuali errori restituiti nel file. `upgrade-prechecks.log`

Per gli aggiornamenti con ripristino tramite snapshot, il precontrollo viene eseguito durante il processo di ripristino. In caso di successo, il database verrà aggiornato alla nuova versione di Aurora MySQL. Se vengono rilevati errori, questi vengono registrati nel `upgrade-prechecks.log` file e l'aggiornamento viene annullato. Prima di tentare nuovamente l'aggiornamento, risolvete gli eventuali errori restituiti nel file. `upgrade-prechecks.log`

Per ulteriori informazioni, consulta Individuazione dei motivi degli errori di aggiornamento di Aurora My SQL major version e Riferimento alle descrizioni di precontrollo per Aurora MySQL.

Per monitorare lo stato del precontrollo, è possibile visualizzare i seguenti eventi sul cluster DB.

Precontrolla lo stato	Messaggio di evento	Azione
Avviato	Preparazione dell'aggi ornamento in corso: avvio dei controlli preliminari di aggiornamento online.	Nessuno
Non riuscito	Il cluster di database si trova in uno stato che non può essere aggiornato: i controlli preliminari di aggiornam ento non sono riusciti. Per ulteriori dettagli, consulta il file upgrade-prechecks.log. Per ulteriori informazioni sulla risoluzione della causa	Verifica la `upgrade-p rechecks.log` presenza di errori. Correggere gli errori. Riprova a eseguire l'aggiorn amento.

Precontrolla lo stato	Messaggio di evento	Azione
	dell'errore di aggiornamento, vedere https://docs.aws.amazon.com/AmazonRDS/più recente//AuroraUserGuide.upgrading.Troubleshooting.html AuroraMySQL	
Riuscito	Preparazione dell'aggiornamento in corso: precontrolli preliminari di aggiornamento online completati.	Il precontrollo è riuscito e non sono stati restituiti errori. Verifica `upgrade-prechecks.log` la presenza di avvisi e avvisi.

Per ulteriori informazioni sulla visualizzazione degli eventi, vedere. Visualizzazione degli RDS eventi Amazon

Formato di registro di Precheck per Aurora My SQL

Una volta completati i controlli di compatibilità dell'aggiornamento (controlli preliminari), puoi esaminare il file. `upgrade-prechecks.log` Il file di registro contiene i risultati, gli oggetti interessati e le informazioni di riparazione per ogni precontrollo.

Gli errori bloccano l'aggiornamento. È necessario risolverli prima di riprovare l'aggiornamento.

Gli avvisi e gli avvisi sono meno importanti, ma consigliamo comunque di esaminarli attentamente per assicurarsi che non vi siano problemi di compatibilità con il carico di lavoro dell'applicazione. Risolvete presto eventuali problemi identificati.

Il file di registro ha il seguente formato:

- `targetVersion`— La versione SQL compatibile con My dell'aggiornamento Aurora SQL My.
- `auroraServerVersion`— La SQL versione Aurora My su cui è stato eseguito il precheck.
- `auroraTargetVersion`— La SQL versione Aurora My a cui stai effettuando l'aggiornamento.
- `checksPerformed`— Contiene l'elenco dei precontrolli eseguiti.

- `id`— Il nome del precontrollo in esecuzione.
- `title`— Una descrizione del precontrollo in corso.
- `status`— Ciò non indica se il precontrollo è riuscito o meno, ma mostra lo stato della richiesta di precontrollo:
  - `OK`— La query di precontrollo è stata eseguita e completata correttamente.
  - `ERROR`— La query di precontrollo non è stata eseguita. Ciò può verificarsi a causa di problemi quali vincoli di risorse, riavvii imprevisti dell'istanza o interruzione della query di precontrollo della compatibilità.

    Per ulteriori informazioni, consulta questo esempio.
- `description`— Una descrizione generale dell'incompatibilità e di come risolvere il problema.
- `documentationLink`— Ove applicabile, qui è riportato un collegamento alla SQL documentazione pertinente di Aurora My SQL o My. Per ulteriori informazioni, consulta Riferimento alle descrizioni di precontrollo per Aurora MySQL.
- `detectedProblems`— Se il precontrollo restituisce un errore, un avviso o un avviso, mostra i dettagli dell'incompatibilità e, ove applicabile, degli oggetti incompatibili:
  - `level`— Il livello di incompatibilità rilevato dal precontrollo. I livelli validi sono i seguenti:
    - `Error`— L'aggiornamento non può procedere finché non viene risolta l'incompatibilità.
    - `Warning`— L'aggiornamento può continuare, ma è stato rilevato un oggetto, una sintassi o una configurazione obsoleti. Esamina attentamente gli avvisi e risolvili al più presto per evitare problemi nelle versioni future.
    - `Notice`— L'aggiornamento può continuare, ma è stato rilevato un oggetto, una sintassi o una configurazione obsoleti. Esamina attentamente gli avvisi e risolvili al più presto per evitare problemi nelle versioni future.
  - `dbObject`— Il nome dell'oggetto del database in cui è stata rilevata l'incompatibilità.
  - `description`— Una descrizione dettagliata dell'incompatibilità e di come risolvere il problema.
- `errorCount`— Il numero di errori di incompatibilità rilevati. Questi bloccano l'aggiornamento.
- `warningCount`— Il numero di avvisi di incompatibilità rilevati. Questi non bloccano l'aggiornamento, ma li risolvono presto per evitare problemi nelle versioni future.
- `noticeCount`— Il numero di avvisi di incompatibilità rilevati. Questi problemi non bloccano l'aggiornamento, ma risolvili al più presto per evitare problemi nelle versioni future.
- `Summary`— Un riepilogo del conteggio degli errori di compatibilità, degli avvisi e degli avvisi di precontrollo.

Esempi di output del log di Precheck per Aurora My SQL

Gli esempi seguenti mostrano l'output del registro di precontrollo che potresti vedere. Per i dettagli sui precontrolli eseguiti, vedere. Riferimento alle descrizioni di precontrollo per Aurora MySQL

Stato di precontrollo OK, non è stata rilevata alcuna incompatibilità

L'interrogazione di precontrollo è stata completata correttamente. Non sono state rilevate incompatibilità.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinytext",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny text columns",
  "status": "OK",
  "detectedProblems": []
},
```

Stato di precontrollo OK, errore rilevato

L'interrogazione di precontrollo è stata completata correttamente. È stato rilevato un errore.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexes",
  "status": "OK",
  "description": "Consider dropping the prefix indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test25.sbtest1",
        "description": "Table `test25`.`sbtest1` has an index `idx_t1` on geometry
 column/s. Mysql 8.0 does not support this type of index on a geometry column
 https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run 'DROP
 INDEX `idx_t1` ON `test25`.`sbtest1`;"
      },
 }
```

Stato di precontrollo OK, avviso rilevato

Gli avvisi possono essere restituiti quando un precontrollo ha esito positivo o negativo.

Qui la richiesta di precontrollo è stata completata con successo. Sono stati rilevati due avvisi.


```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
    ]
}
```


Verifica preliminare lo statoERROR, nessuna incompatibilità segnalata

La query di precontrollo non è riuscita con un errore, quindi non è stato possibile verificare le incompatibilità.


```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "ERROR",
  "description": "Can't connect to MySQL server on 'localhost:3306' (111) at
 13/08/2024 12:22:20 UTC. This failure can occur due to low memory available on the
 instance for executing upgrade prechecks. Please check 'FreeableMemory' Cloudwatch
 metric to verify the available memory on the instance while executing prechecks. If
```

```
 instance ran out of memory, we recommend to retry the upgrade on a higher instance
 class."
}
```

Questo errore può verificarsi a causa di un riavvio imprevisto dell'istanza o dell'interruzione di una query di precontrollo della compatibilità sul database durante l'esecuzione. Ad esempio, su classi di istanze DB più piccole, questo problema potrebbe verificarsi quando la memoria disponibile sull'istanza si esaurisce.

Puoi utilizzare il CloudWatch parametro `FreeableMemory` Amazon per verificare la memoria disponibile sull'istanza durante l'esecuzione dei precontrolli. Se l'istanza ha esaurito la memoria, consigliamo di riprovare l'aggiornamento su una classe di istanza DB più grande. In alcuni casi, è possibile utilizzare una distribuzione Blue/Green. Ciò consente l'esecuzione di precontrolli e aggiornamenti sul cluster DB «verde» indipendentemente dal carico di lavoro di produzione, che consuma anche risorse di sistema.

Per ulteriori informazioni, consulta Risoluzione dei problemi di utilizzo della memoria per i database Aurora MySQL.

Riepilogo del precontrollo, sono stati rilevati un errore e tre avvisi

I controlli preliminari di compatibilità contengono anche informazioni sulle versioni di Aurora SQL My di origine e di destinazione e un riepilogo del conteggio degli errori, degli avvisi e degli avvisi alla fine dell'output del precontrollo.

Ad esempio, l'output seguente mostra che è stato effettuato un tentativo di aggiornamento da Aurora My SQL 2.11.6 ad Aurora My 3.07.1. SQL L'aggiornamento ha restituito un errore, tre avvisi e nessun avviso. Poiché gli aggiornamenti non possono procedere quando viene restituito un errore, è necessario risolvere il problema di routineSyntaxCheckcompatibilità e riprovare l'aggiornamento.

```
{
  "serverAddress": "/tmp%2Fmysql.sock",
  "serverVersion": "5.7.12 - MySQL Community Server (GPL)",
  "targetVersion": "8.0.36",
  "auroraServerVersion": "2.11.6",
  "auroraTargetVersion": "3.07.1",
  "outfilePath": "/rdsdbdata/tmp/PreChecker.log",
  "checksPerformed": [{
      ... output for each individual precheck ...
      .
      .
```

```
      {
        "id": "oldTemporalCheck",
        "title": "Usage of old temporal type",
        "status": "OK",
          "detectedProblems": []
      },
      {
        "id": "routinesSyntaxCheck",
        "title": "MySQL 8.0 syntax check for routine-like objects",
        "status": "OK",
        "description": "The following objects did not pass a syntax check with
 the latest MySQL 8.0 grammar. A common reason is that they reference names that
 conflict with new reserved keywords. You must update these routine definitions and
 `quote` any such references before upgrading.",
        "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
        "detectedProblems": [{
            "level": "Error",
            "dbObject": "test.select_res_word",
            "description": "at line 2,18: unexpected token 'except'"
        }]
      },
      .
      .
      .
      {
        "id": "zeroDatesCheck",
        "title": "Zero Date, Datetime, and Timestamp values",
        "status": "OK",
        "description": "Warning: By default zero date/datetime/timestamp values
 are no longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are
 included in SQL_MODE by default. These modes should be used with strict mode as
 they will be merged with strict mode in a future release. If you do not include
 these modes in your SQL_MODE setting, you are able to insert date/datetime/
timestamp values that contain zeros. It is strongly advised to replace zero values
 with valid ones, as they may not work correctly in the future.",
        "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
        "detectedProblems": [{
            "level": "Warning",
            "dbObject": "global.sql_mode",
            "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
            },
            {
            "level": "Warning",
```

```
            "dbObject": "session.sql_mode",
            "description": " of 8 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
            }
          ]
      },
      .
      .
      .
  }],
  "errorCount": 1,
  "warningCount": 3,
  "noticeCount": 0,
  "Summary": "1 errors were found. Please correct these issues before upgrading to
 avoid compatibility issues."
}
```

Verifica preliminare le prestazioni di Aurora My SQL

I precontrolli di compatibilità vengono eseguiti prima che l'istanza DB venga messa offline per l'aggiornamento, quindi in circostanze normali non causano tempi di inattività dell'istanza DB durante l'esecuzione. Tuttavia, possono influire sul carico di lavoro dell'applicazione in esecuzione sull'istanza Writer DB. I precontrolli accedono al dizionario dei dati tramite le tabelle information_schema, che possono essere lente se ci sono molti oggetti del database. Considerate i seguenti fattori:

- La durata del precontrollo varia in base al numero di oggetti del database come tabelle, colonne, routine e vincoli. L'esecuzione dei cluster DB con un numero elevato di oggetti può richiedere più tempo.

  Ad esempio, removedFunctionsCheckpossono richiedere più tempo e utilizzare più risorse in base al numero di oggetti archiviati.
- Per gli aggiornamenti sul posto, l'utilizzo di una classe di istanza DB più grande (ad esempio, db.r5.24xlarge o db.r6g.16xlarge) può aiutare a completare l'aggiornamento più rapidamente utilizzandone di più. CPU È possibile ridurre le dimensioni dopo l'aggiornamento.
- Le interrogazioni `information_schema` su più database possono essere lente, specialmente con molti oggetti e su istanze DB più piccole. In questi casi, prendi in considerazione l'utilizzo della clonazione, del ripristino delle istantanee o di una distribuzione Blue/Green per gli aggiornamenti.
- Il precontrollo dell'utilizzo delle risorse (e della memoria) può aumentare con più oggettiCPU, con conseguenti tempi di esecuzione più lunghi su istanze DB più piccole. In questi casi, prendi in

considerazione la possibilità di eseguire test utilizzando la clonazione, il ripristino delle istantanee o una distribuzione Blue/Green per gli aggiornamenti.

Se i precontrolli falliscono a causa della mancanza di risorse, è possibile rilevarlo nel registro di precontrollo utilizzando l'output di stato:

```
"status": "ERROR",
```

Per ulteriori informazioni, consulta Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco e Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL.

Riepilogo dei controlli preliminari di Community My upgrade SQL

Di seguito è riportato un elenco generale delle incompatibilità tra My SQL 5.7 e 8.0:

- Il cluster DB SQL compatibile con My 5.7 non deve utilizzare funzionalità non supportate in My 8.0. SQL

  Per ulteriori informazioni, consulta Funzionalità rimosse in My SQL 8.0 nella documentazione personale. SQL
- Non devono essere presenti violazioni di parole chiave o parole riservate. Alcune parole chiave potrebbero essere riservate in My SQL 8.0 che non lo erano in precedenza.

  Per ulteriori informazioni, consulta Parole chiave e parole riservate nella SQL documentazione My.
- Per supporto Unicode migliorato, valuta la conversione di oggetti che utilizzano il charset `utf8mb3` per utilizzare il charset `utf8mb4`. Il set di caratteri `utf8mb3` è obsoleto. Inoltre, valuta l'utilizzo di `utf8mb4` per i riferimenti al set di caratteri anziché `utf8`, perché attualmente `utf8` è un'alias per il charset `utf8mb3`.

  Per ulteriori informazioni, vedete Il set di caratteri utf8mb3 (codifica unicode a 3 byte UTF -8) nella mia documentazione. SQL
- Non devono esserci tabelle InnoDB con un formato di riga non predefinito.
- Non devono essere presenti attributi di tipo «no» `ZEROFILL` o di tipo «`display`lunghezza».
- Non devono essere presenti tabelle partizionate che utilizzano un motore di storage che non dispone di supporto di partizionamento nativo.

- Nel database di `mysql` sistema My SQL 5.7 non devono esserci tabelle con lo stesso nome di una tabella utilizzata dal dizionario di dati My SQL 8.0.
- Non devono essere presenti tabelle che utilizzano tipi di dati o funzioni obsolete.
- Non devono essere presenti nomi di vincoli della chiave più lunghi di 64 caratteri.
- Non devono essere definite SQL modalità obsolete nelle impostazioni delle variabili di `sql_mode` sistema.
- Non devono esserci tabelle o stored procedure con elementi singoli `ENUM` o di `SET` colonna che superino i 255 caratteri di lunghezza.
- Non devono esserci partizioni di tabella che risiedono in tablespace InnoDB condivisi.
- Non devono esserci riferimenti circolari nei percorsi dei file di dati del tablespace.
- Non devono essere presenti interrogazioni e definizioni di programmi memorizzate che utilizzano `ASC` o `DESC` qualificatori per le clausole. `GROUP BY`
- Non devono essere rimosse variabili di sistema e le variabili di sistema devono utilizzare i nuovi valori predefiniti per My 8.0. SQL
- Non devono essere presenti valori di data, datetime o timestamp pari a zero (`0`).
- Non devono esserci incongruenze nello schema dovute alla rimozione o al danneggiamento dei file.
- Non devono esserci nomi di tabella che contengano la stringa di `FTS` caratteri.
- Non devono esserci tabelle InnoDB che appartengano a un motore diverso.
- Non devono esserci nomi di tabelle o schemi non validi per My SQL 5.7.

Per i dettagli sui precontrolli eseguiti, vedere. Riferimento alle descrizioni di precontrollo per Aurora MySQL

Per ulteriori informazioni sull'aggiornamento a My SQL 8.0, consulta Upgrading My nella documentazione My. SQL SQL Per una descrizione generale delle modifiche apportate a My SQL 8.0, consulta Cosa c'è di nuovo in My 8.0 nella documentazione My SQL. SQL

Riepilogo dei controlli preliminari di Aurora My upgrade SQL

Aurora My SQL ha i propri requisiti specifici per l'aggiornamento dalla versione 2 alla versione 3, tra cui:

- Non deve essere presente alcuna SQL sintassi obsoleta, ad esempio, e, nelle viste`SQL_CACHE`, nelle `SQL_NO_CACHE` routine, nei `QUERY_CACHE` trigger e negli eventi.

- Nessuna `FTS_DOC_ID` colonna deve essere presente in nessuna tabella senza l'indice. `FTS`
- Non deve esserci alcuna discrepanza nella definizione delle colonne tra il dizionario dei dati InnoDB e la definizione effettiva della tabella.
- Tutti i nomi di database e tabelle devono essere in minuscolo quando il `lower_case_table_names` parametro è impostato su. `1`
- Gli eventi e i trigger non devono avere un definitore mancante o vuoto o un contesto di creazione non valido.
- Tutti i nomi dei trigger in un database devono essere univoci.
- DDLrecovery e Fast DDL non sono supportati in Aurora My SQL versione 3. Non devono esserci artefatti nei database relativi a queste funzionalità.
- Le tabelle con il formato di `COMPACT` riga `REDUNDANT` o non possono avere indici più grandi di 767 byte.
- La lunghezza del prefisso degli indici definiti nelle colonne di `tiny` testo non può superare i 255 byte. Con il set di `utf8mb4` caratteri, ciò limita la lunghezza del prefisso supportata a 63 caratteri.

  Una lunghezza del prefisso maggiore era consentita in My SQL 5.7 utilizzando il parametro. `innodb_large_prefix` Questo parametro è obsoleto in My 8.0. SQL
- Non deve esserci alcuna incoerenza dei metadati InnoDB nella tabella. `mysql.host`
- Non deve esserci alcuna discrepanza tra i tipi di dati delle colonne nelle tabelle di sistema.
- Non devono esserci transazioni XA nello `prepared` stato.
- I nomi delle colonne nelle viste non possono superare i 64 caratteri.
- I caratteri speciali nelle stored procedure non possono essere incoerenti.
- Le tabelle non possono presentare incoerenze nel percorso dei file di dati.

Per i dettagli sui precontrolli eseguiti, consulta. Riferimento alle descrizioni di precontrollo per Aurora MySQL

Riferimento alle descrizioni di precontrollo per Aurora MySQL

I precontrolli di aggiornamento per Aurora MySQL sono descritti qui in dettaglio.

Indice

- Errori
  - Precontrolli MySQL che segnalano errori

  - Precontrolli Aurora MySQL che segnalano errori
- Avvisi
  - Precontrolli MySQL che segnalano avvisi
  - Aurora MySQL precontrolli che segnalano gli avvisi
- Note
- Errori, avvertenze o avvisi

Errori

I seguenti precontrolli generano errori quando il precontrollo fallisce e l'aggiornamento non può procedere.

Argomenti

- Precontrolli MySQL che segnalano errori
- Precontrolli Aurora MySQL che segnalano errori

Precontrolli MySQL che segnalano errori

I seguenti controlli preliminari provengono da Community MySQL:

- checkTableMysqlSchema
- circularDirectoryCheck
- columnsWhichCannotHaveDefaultsCheck
- depreciatedSyntaxCheck
- engineMixupCheck
- enumSetElementLengthCheck
- foreignKeyLengthControlla
- getDuplicateTriggers
- getEventsWithNullDefiner
- getMismatchedMetadata
- getTriggersWithNullDefiner
- getValueOfVariableLower_case_table_names
- groupByAscSyntaxCheck

- mysqlEmptyDotTableSyntaxCheck
- mysqlIndexTooLargeCheck
- MySQL è valido 57 NamesCheck
- mysqlOrphanedRoutinesControlla
- mysqlSchemaCheck
- nonNativePartitioningControlla
- oldTemporalCheck
- partitionedTablesInCondiviso TablespaceCheck
- removedFunctionsCheck
- routineSyntaxCheck
- schemaInconsistencyCheck

checkTableMysqlSchema

Livello di precontrollo: errore

Problemi segnalati dal **`check table x for upgrade`** comando per lo schema **`mysql`**

Prima di iniziare l'aggiornamento ad Aurora MySQL versione 3, `check table for upgrade` viene eseguito su ogni tabella dello schema dell'`mysql`istanza DB. Il `check table for upgrade` comando esamina le tabelle per individuare eventuali problemi che potrebbero insorgere durante l'aggiornamento a una versione più recente di MySQL. L'esecuzione di questo comando prima di tentare un aggiornamento può aiutare a identificare e risolvere eventuali incompatibilità in anticipo, rendendo più agevole l'effettivo processo di aggiornamento.

Questo comando esegue diversi controlli su ogni tabella, come i seguenti:

- Verifica della compatibilità della struttura della tabella e dei metadati con la versione di MySQL di destinazione
- Verifica della presenza di eventuali funzionalità obsolete o rimosse utilizzate dalla tabella
- Garantire che la tabella possa essere aggiornata correttamente senza perdita di dati

Per ulteriori informazioni, vedere l'istruzione CHECK TABLE nella documentazione di MySQL.

Esempio di output:

```
{
```

```
  "id": "checkTableMysqlSchema",
  "title": "Issues reported by 'check table x for upgrade' command for mysql
 schema.",
  "status": "OK",
  "detectedProblems": []
}
```

L'output di questo precontrollo dipende dall'errore riscontrato e dal momento in cui si verifica, poiché `check table for upgrade` esegue più controlli.

Se riscontri errori con questo precontrollo, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

circularDirectoryCheck

Livello di precontrollo: errore

Riferimenti circolari alle directory nei percorsi dei file di dati del tablespace

A partire da MySQL 8.0.17, la clausola non consente più riferimenti `CREATE TABLESPACE ... ADD DATAFILE` circolari alle directory. Per evitare problemi di aggiornamento, rimuovi tutti i riferimenti circolari alle directory dai percorsi dei file di dati del tablespace prima di eseguire l'aggiornamento ad Aurora MySQL versione 3.

Esempio di output:

```
{
  "id": "circularDirectory",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "description": "Error: Following tablespaces contain circular directory references
 (e.g. the reference '/../') in data file paths which as of MySQL 8.0.17 are not
 permitted by the CREATE TABLESPACE ... ADD DATAFILE clause. An exception to the
 restriction exists on Linux, where a circular directory reference is permitted if
 the preceding directory is a symbolic link. To avoid upgrade issues, remove any
 circular directory references from tablespace data file paths before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-innodb-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "ts2",
```

```
        "description": "circular reference in datafile path: '/home/ec2-user/dbdata/
mysql_5_7_44/../ts2.ibd'",
        "dbObjectType": "Tablespace"
      }
  ]
}
```

Se ricevi questo errore, ricostruisci le tabelle utilizzando un file-per-table tablespace. Utilizza i percorsi di file predefiniti per tutte le tablespace e le definizioni delle tabelle.

> **Note**
>
> Aurora MySQL non supporta tablespace o comandi generici. `CREATE TABLESPACE` Prima di ricostruire i tablespace, consulta le operazioni DDL online nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

Dopo la ricostruzione, il controllo preliminare viene superato e l'aggiornamento può procedere.

```
{
  "id": "circularDirectoryCheck",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "detectedProblems": []
},
```

columnsWhichCannotHaveDefaultsCheck

Livello di precontrollo: errore

Colonne che non possono avere valori predefiniti

Prima di MySQL 8.0.13`BLOB`,, `TEXT``GEOMETRY`, `JSON` e le colonne non possono avere valori predefiniti. Rimuovi tutte le clausole predefinite su queste colonne prima di eseguire l'aggiornamento ad Aurora MySQL versione 3. Per ulteriori informazioni sulle modifiche alla gestione predefinita per questi tipi di dati, consulta i Valori predefiniti del tipo di dati nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "description": "Error: The following columns are defined as either BLOB, TEXT,
 GEOMETRY or JSON and have a default value set. These data types cannot have default
 values in MySQL versions prior to 8.0.13, while starting with 8.0.13, the default
 value must be specified as an expression. In order to fix this issue, please use
 the ALTER TABLE ... ALTER COLUMN ... DROP DEFAULT statement.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/data-type-
defaults.html#data-type-defaults-explicit",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test_blob_default.geo_col",
        "description": "geometry"
      }
  ]
},
```

Il precontrollo restituisce un errore perché la `geo_col` colonna della `test.test_blob_default` tabella utilizza un tipo di `JSON` dati `BLOB` `TEXT``GEOMETRY`,, o con un valore predefinito specificato.

Guardando la definizione della tabella, possiamo vedere che la `geo_col` colonna è definita come`geo_col geometry NOT NULL default ''`.

```
mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Rimozione di questa clausola predefinita per consentire il superamento del precontrollo.

> **Note**
>
> Prima di eseguire `ALTER TABLE` istruzioni o ricostruire tablespace, consulta le operazioni DDL online nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
mysql> ALTER TABLE test_blob_default modify COLUMN geo_col geometry NOT NULL;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

Il controllo preliminare viene superato e puoi riprovare a eseguire l'aggiornamento.

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "detectedProblems": []
},
```

depreciatedSyntaxCheck

Livello di precontrollo: errore

Utilizzo di parole chiave deprezzate nella definizione

MySQL 8.0 ha rimosso la cache delle query. Di conseguenza, è stata rimossa una parte della sintassi SQL specifica della cache delle query. Se uno qualsiasi degli oggetti del database contiene le SQL_NO_CACHE parole chiave, o QUERY CACHESQL_CACHE, viene restituito un errore di precontrollo. Per risolvere questo problema, ricreate questi oggetti, rimuovendo le parole chiave menzionate.

Esempio di output:

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "description": "Error: The following DB objects contain keywords like 'QUERY
 CACHE', 'SQL_CACHE', 'SQL_NO_CACHE' which are not supported in major version 8.0.
```

```
 It is recommended to drop these DB objects or rebuild without any of the above
 keywords before upgrade.",
  "detectedProblems": [
      {
"level": "Error",
"dbObject": "test.no_query_cache_check",
"description": "PROCEDURE uses depreciated words in definition"
      }
  ]
}
```

Il precontrollo riporta che la `test.no_query_cache_check` stored procedure utilizza una delle parole chiave rimosse. Guardando la definizione della procedura, possiamo vedere che utilizza`SQL_NO_CACHE`.

```
mysql> show create procedure test.no_query_cache_check\G
*************************** 1. row ***************************
           Procedure: no_query_cache_check
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `no_query_cache_check`()
BEGIN
    SELECT SQL_NO_CACHE k from sbtest1 where id > 10 and id < 20 group by k asc;
END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Rimuovi la parola chiave.

```
mysql> drop procedure test.no_query_cache_check;
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter //

mysql> CREATE DEFINER=`reinvent`@`%` PROCEDURE `no_query_cache_check`() BEGIN
 SELECT k from sbtest1 where id > 10 and id < 20 group by k asc; END//
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Dopo aver rimosso la parola chiave, il precontrollo viene completato correttamente.

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "detectedProblems": []
}
```

engineMixupCheck

Livello di precontrollo: errore

Tabelle riconosciute da InnoDB che appartengono a un motore diverso

Analogamente schemaInconsistencyCheck, questo precontrollo verifica che i metadati delle tabelle in MySQL siano coerenti prima di procedere con l'aggiornamento.

Se riscontri errori con questo precontrollo, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

Esempio di output:

```
{
  "id": "engineMixupCheck",
  "title": "Tables recognized by InnoDB that belong to a different engine",
  "status": "OK",
  "description": "Error: Following tables are recognized by InnoDB engine while the
 SQL layer believes they belong to a different engine. Such situation may happen
 when one removes InnoDB table files manually from the disk and creates e.g. a
 MyISAM table with the same name.\n\nA possible way to solve this situation is to
 e.g. in case of MyISAM table:\n\n1. Rename the MyISAM table to a temporary name
 (RENAME TABLE).\n2. Create some dummy InnoDB table (its definition does not need to
 match), then copy (copy, not move) and rename the dummy .frm and .ibd files to the
 orphan name using OS file commands.\n3. The orphan table can be then dropped (DROP
 TABLE), as well as the dummy table.\n4. Finally the MyISAM table can be renamed
 back to its original name.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.general_log_backup",
        "description": "recognized by the InnoDB engine but belongs to CSV"
```

```
      }
  ]
}
```

enumSetElementLengthCheck

Livello di precontrollo: errore

ENUMe definizioni di **SET** colonna contenenti elementi più lunghi di 255 caratteri

Le tabelle e le stored procedure non devono contenere elementi `ENUM` o `SET` colonne che superino i 255 caratteri o 1020 byte. Prima di MySQL 8.0, la lunghezza massima combinata era di 64K, ma 8.0 limita i singoli elementi a 255 caratteri o 1020 byte (supporto multibyte). Se si verifica un errore di precontrollo per`enumSetElementLengthCheck`, modifica tutti gli elementi che superano questi nuovi limiti prima di riprovare l'aggiornamento.

Esempio di output:

```
{
  "id": "enumSetElementLengthCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "description": "Error: The following columns are defined as either ENUM or SET and
 contain at least one element longer that 255 characters. They need to be altered so
 that all elements fit into the 255 characters limit.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/string-type-
overview.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.large_set.s",
        "description": "SET contains element longer than 255 characters"
      }
  ]
},
```

Il precontrollo riporta un errore perché la colonna `s` della `test.large_set` tabella contiene un `SET` elemento più grande di 255 caratteri.

Dopo aver ridotto le `SET` dimensioni di questa colonna, il precontrollo viene eseguito e l'aggiornamento può continuare.

```
{
  "id": "enumSetElementLenghtCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "detectedProblems": []
},
```

foreignKeyLengthControlla

Livello di precontrollo: errore

Nomi di vincoli di chiave esterna più lunghi di 64 caratteri

In MySQL, la lunghezza degli identificatori è limitata a 64 caratteri, come indicato nella documentazione di MySQL. A causa dei problemi identificati per cui la lunghezza delle chiavi esterne poteva essere uguale o superiore a questo valore, con conseguenti errori di aggiornamento, è stato implementato questo controllo preliminare. Se si verificano errori con questo precontrollo, è necessario modificare o rinominare il vincolo in modo che sia inferiore a 64 caratteri prima di riprovare l'aggiornamento.

Esempio di output:

```
{
  "id": "foreignKeyLength",
  "title": "Foreign key constraint names longer than 64 characters",
  "status": "OK",
  "detectedProblems": []
}
```

getDuplicateTriggers

Livello di precontrollo: errore

Tutti i nomi dei trigger in un database devono essere univoci.

A causa delle modifiche nell'implementazione del dizionario dei dati, MySQL 8.0 non supporta i trigger con distinzione tra maiuscole e minuscole all'interno di un database. Questo precontrollo verifica che il cluster DB non disponga di uno o più database contenenti trigger duplicati. Per ulteriori informazioni, consulta Identifier case sensitive nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "description": "Error: You have one or more database containing duplicate
 triggers. Mysql 8.0 does not support case sensitive triggers within a database
 https://dev.mysql.com/doc/refman/8.0/en/identifier-case-sensitivity.html. To
 upgrade to MySQL 8.0, drop the triggers with case-insensitive duplicate names and
 recreate with distinct names.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_product"
      },
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_PRODUCT"
      }
  ]
}
```

Il precontrollo riporta un errore indicante che il cluster di database ha due trigger con lo stesso nome, ma che utilizzano casi diversi: `test.before_insert_product` e. `test.before_insert_PRODUCT`

Prima dell'aggiornamento, rinomina i trigger o eliminali e ricreali con un nuovo nome.

Dopo la `test.before_insert_PRODUCT` ridenominazione in, il precontrollo ha esito positivo. `test.before_insert_product_2`

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "detectedProblems": []
}
```

getEventsWithNullDefiner

Livello di precontrollo: errore

La colonna definer per non **`mysql.event`** può essere nulla o vuota.

L'`DEFINER`attributo specifica l'account MySQL che possiede una definizione di oggetto memorizzato, come un trigger, una stored procedure o un evento. Questo attributo è particolarmente utile in situazioni in cui si desidera controllare il contesto di sicurezza in cui viene eseguito l'oggetto archiviato. Quando si crea un oggetto memorizzato, se `DEFINER` non è specificato un, l'impostazione predefinita è l'utente che ha creato l'oggetto.

Quando si esegue l'aggiornamento a MySQL 8.0, non è possibile avere oggetti memorizzati che abbiano un definitore o `null` vuoto nel dizionario dei dati MySQL. Se si dispone di tali oggetti memorizzati, viene generato un errore di precontrollo. È necessario correggerlo prima di procedere con l'aggiornamento.

Esempio di errore:

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "description": "Error: Set definer column in mysql.event to a valid non-null
 definer.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.get_version",
        "description": "Set definer for event get_version in Schema test"
      }
  ]
}
```

Il precontrollo restituisce un errore per l'`test.get_version`evento perché ha un `null` definitore.

Per risolvere questo problema puoi controllare la definizione dell'evento. Come puoi vedere, il definitore è `null` o vuoto.

```
mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+---------+
```

```
| db   | name        | definer |
+------+-------------+---------+
| test | get_version |         |
+------+-------------+---------+
1 row in set (0.00 sec)
```

Elimina o ricrea l'evento con un definitore valido.

> **Note**
>
> Prima di eliminare o ridefinire un`DEFINER`, esamina e verifica attentamente i requisiti dell'applicazione e dei privilegi. Per ulteriori informazioni, vedere Stored object access control nella documentazione di MySQL.

```
mysql> drop event test.get_version;
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> delimiter $$
mysql> CREATE EVENT get_version
    ->     ON SCHEDULE
    ->       EVERY 1 DAY
    ->     DO
    ->      ///DO SOMETHING //
    -> $$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+------------+
| db   | name        | definer    |
+------+-------------+------------+
| test | get_version | reinvent@% |
+------+-------------+------------+
1 row in set (0.00 sec)
```

Ora il precontrollo passa.

```
{
```

```
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "detectedProblems": []},
```

getMismatchedMetadata

Livello di precontrollo: errore

Mancata corrispondenza della definizione delle colonne tra il dizionario dei dati InnoDB e la definizione effettiva della tabella

Analogamente schemaInconsistencyCheck, questo precontrollo verifica che i metadati delle tabelle in MySQL siano coerenti prima di procedere con l'aggiornamento. In questo caso, il precontrollo verifica che le definizioni delle colonne corrispondano tra il dizionario dei dati InnoDB e la definizione della tabella MySQL. Se viene rilevata una mancata corrispondenza, l'aggiornamento non procede.

Se riscontri errori con questo precontrollo, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

Esempio di output:

```
{
  "id": "getMismatchedMetadata",
  "title": "Column definition mismatch between InnoDB Data Dictionary and actual
 table definition.",
  "status": "OK",
  "description": "Error: Your database has mismatched metadata. The upgrade to mysql
 8.0 will not succeed until this is fixed.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.mismatchTable",
        "description": "Table `test/mismatchTable` column names mismatch with InnoDb
 dictionary column names: iD <> id"
      }
  ]
}
```

Il precontrollo riporta una mancata corrispondenza nei metadati per la `id` colonna della tabella. `test.mismatchTable` In particolare, i metadati MySQL hanno il nome della colonna as`iD`, mentre InnoDB lo ha come. `id`

getTriggersWithNullDefiner

Livello di precontrollo: errore

La colonna del definitore per non **`information_schema.triggers`** può essere **`null`** o vuota.

Il precontrollo verifica che il database non abbia trigger definiti con `null` o definitori vuoti. Per ulteriori informazioni sui requisiti di definizione per gli oggetti archiviati, vedere. getEventsWithNullDefiner

Esempio di output:

```
{
  "id": "getTriggersWithNullDefiner",
  "title": "The definer column for information_schema.triggers cannot be null or
 blank.",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.example_trigger",
        "description": "Set definer for trigger example_trigger in Schema test"
      }
  ]
}
```

Il precontrollo restituisce un errore perché il `example_trigger` trigger nello `test` schema ha un `null` definitore. Per correggere questo problema, correggi il definitore ricreando il trigger con un utente valido oppure elimina il trigger. Per ulteriori informazioni, consulta l'esempio in. getEventsWithNullDefiner

> **Note**
>
> Prima di eliminare o ridefinire un`DEFINER`, esamina e controlla attentamente i requisiti dell'applicazione e dei privilegi. Per ulteriori informazioni, vedere Stored object access control nella documentazione di MySQL.

getValueOfVariabileLower_case_table_names

Livello di precontrollo: errore

Tutti i nomi di database o tabelle devono essere in minuscolo quando il **`lower_case_table_names`** parametro è impostato su. **`1`**

Prima di MySQL 8.0, i nomi dei database, i nomi delle tabelle e altri oggetti corrispondevano ai file nella directory dei dati, come i metadati basati su file (.frm). La variabile di sistema lower_case_table_names consente agli utenti di controllare il modo in cui il server gestisce la distinzione tra maiuscole e minuscole degli identificatori per gli oggetti del database e l'archiviazione di tali oggetti di metadati. Questo parametro può essere modificato su un server già inizializzato dopo un riavvio.

Tuttavia, in MySQL 8.0, sebbene questo parametro controlli ancora il modo in cui il server gestisce la distinzione tra maiuscole e minuscole degli identificatori, non può essere modificato dopo l'inizializzazione del dizionario dei dati. Quando si aggiorna o si crea un database MySQL 8.0, il valore impostato per `lower_case_table_names` la prima volta che il dizionario dei dati viene avviato su MySQL, viene utilizzato per tutta la durata del database. Questa restrizione è stata messa in atto come parte dell'implementazione dell'implementazione dell'Atomic Data Dictionary, in cui gli oggetti del database vengono migrati dai metadati basati su file alle tabelle InnoDB interne dello schema. `mysql`

Per ulteriori informazioni, consulta Modifiche al dizionario dei dati nella documentazione di MySQL.

Per evitare problemi durante l'aggiornamento dei metadati basati su file al nuovo Atomic Data Dictionary, questo precontrollo verifica che, quando`lower_case_table_names = 1`, tutte le tabelle vengano archiviate su disco in lettere minuscole. In caso contrario, viene restituito un errore di precontrollo ed è necessario correggere i metadati prima di procedere con l'aggiornamento.

Esempio di output:

```
{
  "id": "getValueOfVariablelower_case_table_names",
  "title": "MySQL pre-checks that all database or table names are lowercase when the
 lower_case_table_names parameter is set to 1.",
  "status": "OK",
  "description": "Error: You have one or more databases or tables with uppercase
 letters in the names, but the lower_case_table_names parameter is set to 1. To
```

```
 upgrade to MySQL 8.0, either change all database or table names to lowercase, or
 set the parameter to 0.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.TEST",
        "description": "Table test.TEST contains one or more capital letters in name
 while lower_case_table_names = 1"
      }
  ]
}
```

Viene restituito un errore perché la tabella `test.TEST` contiene lettere maiuscole, ma `lower_case_table_names` è impostata su. `1`

Per risolvere questo problema, è possibile rinominare la tabella in lettere minuscole o modificare il `lower_case_table_names` parametro sul cluster DB prima di iniziare l'aggiornamento.

> **Note**
>
> Verifica ed esamina attentamente la documentazione sulla distinzione tra maiuscole e minuscole in MySQL e su come tali modifiche potrebbero influire sulla tua applicazione. Consulta anche la documentazione di MySQL 8.0 su come lower_case_table_names vengono gestiti in modo diverso in MySQL 8.0.

groupByAscSyntaxCheck

Livello di precontrollo: errore

Utilizzo della sintassi rimossa **`GROUP BY ASC/DESC`**

A partire da MySQL 8.0.13, la sintassi o `ASC` obsoleta per le clausole è stata rimossa`DESC`. `GROUP BY` Le query che si basano sull'ordinamento potrebbero ora produrre risultati diversi. `GROUP BY` Per ottenere un ordinamento specifico, utilizzate invece una `ORDER BY` clausola. Se nel database sono presenti oggetti che utilizzano questa sintassi, è necessario ricrearli utilizzando una `ORDER BY` clausola prima di riprovare l'aggiornamento. Per ulteriori informazioni, vedere Modifiche SQL nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "groupbyAscSyntaxCheck",
  "title": "Usage of removed GROUP BY ASC/DESC syntax",
  "status": "OK",
  "description": "Error: The following DB objects use removed GROUP BY ASC/DESC
 syntax. They need to be altered so that ASC/DESC keyword is removed from GROUP BY
 clause and placed in appropriate ORDER BY clause.",
  "documentationLink": "https://dev.mysql.com/doc/relnotes/mysql/8.0/en/
news-8-0-13.html#mysqld-8-0-13-sql-syntax",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.groupbyasc",
        "description": "PROCEDURE uses removed GROUP BY ASC syntax",
        "dbObjectType": "Routine"
      }
  ]
}
```

mysqlEmptyDotTableSyntaxCheck

Livello di precontrollo: errore

Verifica la presenza di una **.<table>** sintassi obsoleta utilizzata nelle routine.

In MySQL 8.0, le routine non possono più contenere la sintassi dell'identificatore obsoleta (). \".<table>\" Se alcune routine o trigger memorizzati contengono tali identificatori, l'aggiornamento non riesce. Ad esempio, il seguente `.dot_table` riferimento non è più consentito:

```
mysql> show create procedure incorrect_procedure\G
*************************** 1. row ***************************
           Procedure: incorrect_procedure
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `incorrect_procedure`()
BEGIN delete FROM .dot_table; select * from .dot_table where 1=1; END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Dopo aver ricreato le routine e i trigger per utilizzare la sintassi dell'identificatore e l'escape corretti, il precontrollo viene superato e l'aggiornamento può procedere. Per ulteriori informazioni sugli identificatori, vedere Nomi degli oggetti dello schema nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "mysqlEmptyDotTableSyntaxCheck",
  "title": "Check for deprecated '.<table>' syntax used in routines.",
  "status": "OK",
  "description": "Error: The following routines contain identifiers in deprecated
 identifier syntax (\".<table>\"), and should be corrected before upgrade:\n",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.incorrect_procedure",
        "description": " routine body contains deprecated identifiers."
      }
  ]
}
```

Il precheck restituisce un errore per la `incorrect_procedure` routine nel `test` database perché contiene una sintassi obsoleta.

Dopo aver corretto la routine, il precontrollo ha esito positivo ed è possibile ripetere l'aggiornamento.

mysqlIndexTooLargeCheck

Livello di precontrollo: errore

Verifica la presenza di indici troppo grandi per funzionare su versioni di MySQL superiori alla 5.7

Per i formati di riga compatti o ridondanti, non dovrebbe essere possibile creare un indice con un prefisso superiore a 767 byte. Tuttavia, prima della versione 5.7.35 di MySQL questo era possibile. Per ulteriori informazioni, consulta le note di rilascio di MySQL 5.7.35.

Tutti gli indici interessati da questo bug diventeranno inaccessibili dopo l'aggiornamento a MySQL 8.0. Questo precontrollo identifica gli indici pericolosi che devono essere ricostruiti prima che l'aggiornamento possa procedere.

```
 {
  "id": "mysqlIndexTooLargeCheck",
```

```
  "title": "Check for indexes that are too large to work on higher versions of MySQL
 Server than 5.7",
  "status": "OK",
  "description": "Error: The following indexes ware made too large for their format
 in an older version of MySQL (older than 5.7.34). Normally those indexes within
 tables with compact or redundant row formats shouldn't be larger than 767 bytes. To
 fix this problem those indexes should be dropped before upgrading or those tables
 will be inaccessible.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.table_with_large_idx",
        "description": "IDX_2"
      }
  ]
}
```

Il precontrollo restituisce un errore perché la `test.table_with_large_idx` tabella contiene un indice su una tabella che utilizza un formato di riga compatto o ridondante di dimensioni superiori a 767 byte. Queste tabelle diventerebbero inaccessibili dopo l'aggiornamento a MySQL 8.0. Prima di procedere con l'aggiornamento, effettuate una delle seguenti operazioni:

- Eliminate l'indice indicato nel precontrollo.
- Aggiungi un indice menzionato nel precontrollo.
- Cambia il formato di riga usato dalla tabella.

Qui ricostruiamo la tabella per risolvere l'errore di precontrollo. Prima di ricostruire la tabella, assicurati che innodb_file_format sia impostato su e innodb_default_row_format sia impostato `Barracuda` su. `dynamic` Queste sono le impostazioni predefinite in MySQL 5.7. Per ulteriori informazioni, consulta i formati di riga InnoDB e la gestione dei formati di file InnoDB nella documentazione MySQL.

> **Note**
>
> Prima di ricostruire i tablespace, consulta le operazioni DDL online nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
mysql > select @@innodb_file_format,@@innodb_default_row_format;
```

```
+----------------------+-----------------------------+
| @@innodb_file_format | @@innodb_default_row_format |
+----------------------+-----------------------------+
| Barracuda            | dynamic                     |
+----------------------+-----------------------------+
1 row in set (0.00 sec)

mysql> optimize table table_with_large_idx;
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| Table                     | Op       | Msg_type | Msg_text
                              |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| test.table_with_large_idx | optimize | note     | Table does not support optimize,
 doing recreate + analyze instead |
| test.table_with_large_idx | optimize | status   | OK
                              |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.02 sec)

# Verify FILE_FORMAT and ROW_FORMAT
mysql>  select * from information_schema.innodb_sys_tables where name like 'test/
table_with_large_idx';
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
| TABLE_ID | NAME                      | FLAG | N_COLS | SPACE | FILE_FORMAT |
 ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
|       43 | test/table_with_large_idx |   33 |      4 |    26 | Barracuda   |
 Dynamic    |             0 | Single     |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
1 row in set (0.00 sec)
```

Dopo aver ricostruito la tabella, il precontrollo viene superato e l'aggiornamento può procedere.

```
{
  "id": "mysqlIndexTooLargeCheck",
  "title": "Check for indexes that are too large to work on higher versions of MySQL
 Server than 5.7",
```

```
  "status": "OK",
  "detectedProblems": []
},
```

MySQL è valido 57 NamesCheck

Livello di precontrollo: errore

Verifica la presenza di nomi di tabelle e schemi non validi utilizzati in MySQL 5.7

Durante la migrazione al nuovo dizionario dei dati in MySQL 8.0, l'istanza del database non può contenere schemi o tabelle con il prefisso. `#mysql50#` Se esistono oggetti di questo tipo, l'aggiornamento non riesce. Per risolvere questo problema, esegui mysqlcheck sugli schemi e sulle tabelle restituiti.

> **Note**
>
> Assicurati di utilizzare una versione `mysqlcheck` di MySQL 5.7, perché - fix-db-names - e - fix-table-names - sono state rimosse da MySQL 8.0.

Esempio di output:

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "description": "The following tables and/or schemas have invalid names. In order
 to fix them use the mysqlcheck utility as follows:\n\n  $ mysqlcheck --check-
upgrade --all-databases\n  $ mysqlcheck --fix-db-names --fix-table-names --all-
databases\n\nOR via mysql client, for eg:\n\n  ALTER DATABASE `#mysql50#lost+found`
 UPGRADE DATA DIRECTORY NAME;",
  "documentationLink": "https://dev.mysql.com/doc/refman/5.7/en/identifier-
mapping.html https://dev.mysql.com/doc/refman/5.7/en/alter-database.html https://
dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "#mysql50#fix_db_names",
        "description": "Schema name"
      }
  ]
```

```
}
```

Il precontrollo riporta che lo schema `#mysql50#fix_db_names` ha un nome non valido.

Dopo aver corretto il nome dello schema, il precontrollo viene eseguito e l'aggiornamento può procedere.

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlOrphanedRoutinesControlla

Livello di precontrollo: errore

Verifica la presenza di routine orfane in 5.7

Durante la migrazione al nuovo dizionario dei dati in MySQL 8.0, se nel database sono presenti procedure memorizzate in cui lo schema non esiste più, l'aggiornamento non riesce. Questo precontrollo verifica che tutti gli schemi a cui si fa riferimento nelle stored procedure sull'istanza DB esistano ancora. Per consentire il proseguimento dell'aggiornamento, elimina queste stored procedure.

Esempio di output:

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "description": "Error: The following routines have been orphaned. Schemas that
 they are referencing no longer exists.\nThey have to be cleaned up or the upgrade
 will fail.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "dropped_db.get_version",
        "description": "is orphaned"
      }
  ]
```

```
},
```

Il precontrollo riporta che la `get_version` stored procedure nel `dropped_db` database è orfana.

Per ripulire questa procedura, è possibile ricreare lo schema mancante.

```
mysql> create database dropped_db;
Query OK, 1 row affected (0.01 sec)
```

Dopo aver ricreato lo schema, è possibile interrompere la procedura per consentire il proseguimento dell'aggiornamento.

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlSchemaCheck

Livello di precontrollo: errore

I nomi delle tabelle nello **`mysql`** schema sono in conflitto con le nuove tabelle in MySQL 8.0

Il nuovo Atomic Data Dictionary introdotto in MySQL 8.0 memorizza tutti i metadati in un set di tabelle InnoDB interne nello schema. `mysql` Durante l'aggiornamento, le nuove tabelle interne del dizionario dei dati vengono create nello schema. `mysql` Per evitare collisioni di denominazione, che potrebbero causare errori di aggiornamento, il controllo preliminare esamina tutti i nomi delle tabelle dello `mysql` schema per assicurarsi che nessuno dei nuovi nomi di tabella sia già in uso. In caso affermativo, viene restituito un errore e l'aggiornamento non può continuare.

Esempio di output:

```
{
  "id": "mysqlSchema",
  "title": "Table names in the mysql schema conflicting with new tables in the
 latest MySQL.",
  "status": "OK",
  "description": "Error: The following tables in mysql schema have names that will
 conflict with the ones introduced in the latest version. They must be renamed or
```

```
 removed before upgrading (use RENAME TABLE command). This may also entail changes
 to applications that use the affected tables.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrade-before-you-
begin.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.tablespaces",
        "description": "Table name used in mysql schema.",
        "dbObjectType": "Table"
      }
  ]
}
```

Viene restituito un errore perché `mysql` nello schema è presente una tabella denominata`tablespaces`. Questo è uno dei nuovi nomi di tabella del dizionario dati interno di MySQL 8.0. È necessario rinominare o rimuovere tali tabelle prima dell'aggiornamento, utilizzando il comando. `RENAME TABLE`

nonNativePartitioningControlla

Livello di precontrollo: errore

Tabelle partizionate utilizzando motori con partizionamento non nativo

Secondo la documentazione di MySQL 8.0, due motori di archiviazione attualmente forniscono supporto nativo per il partizionamento: InnoDB e NDB. Di questi, solo InnoDB è supportato in Aurora MySQL versione 3, compatibile con MySQL 8.0. Qualsiasi tentativo di creare tabelle partizionate in MySQL 8.0 utilizzando qualsiasi altro motore di archiviazione fallisce. Questo precontrollo cerca le tabelle nel cluster DB che utilizzano il partizionamento non nativo. Se ne vengono restituiti alcuni, è necessario rimuovere il partizionamento o convertire il motore di archiviazione in InnoDB.

Esempio di output:

```
{
  "id": "nonNativePartitioning",
  "title": "Partitioned tables using engines with non native partitioning",
  "status": "OK",
  "description": "Error: In the latest MySQL storage engine is responsible for
 providing its own partitioning handler, and the MySQL server no longer provides
 generic partitioning support. InnoDB and NDB are the only storage engines that
```

```
 provide a native partitioning handler that is supported in the latest MySQL. A
 partitioned table using any other storage engine must be altered—either to convert
 it to InnoDB or NDB, or to remove its partitioning—before upgrading the server,
 else it cannot be used afterwards.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-configuration-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partMyisamTable",
         "description": "MyISAM engine does not support native partitioning",
         "dbObjectType": "Table"
      }
  ]
}
```

Qui una tabella MyISAM utilizza il partizionamento, che richiede un'azione prima che l'aggiornamento possa procedere.

oldTemporalCheck

Livello di precontrollo: errore

Utilizzo del vecchio tipo temporale

I «vecchi temporali» si riferiscono alle colonne di tipo temporale (come `TIMESTAMP` e`DATETIME`) create nelle versioni di MySQL 5.5 e precedenti. In MySQL 8.0, il supporto per questi vecchi tipi di dati temporali viene rimosso, il che significa che gli aggiornamenti in loco da MySQL 5.7 a 8.0 non sono possibili se il database contiene questi vecchi tipi temporali. Per risolvere questo problema, è necessario ricostruire tutte le tabelle contenenti questi vecchi tipi di date temporali prima di procedere con l'aggiornamento.

Per ulteriori informazioni sulla deprecazione dei vecchi tipi di dati temporali in MySQL 5.7, consulta questo blog. Per ulteriori informazioni sulla rimozione dei vecchi tipi di dati temporali in MySQL 8.0, consulta questo blog.

> **Note**
>
> Prima di ricostruire i tablespace, consulta le operazioni DDL online nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

Esempio di output:

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "description": "Error: Following table columns use a deprecated and no longer
 supported temporal disk storage format. They must be converted to the new format
 before upgrading. It can by done by rebuilding the table using 'ALTER TABLE
 <table_name> FORCE' command",
  "documentationLink": "https://dev.mysql.com/blog-archive/mysql-8-0-removing-
support-for-old-temporal-datatypes/",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.55_temporal_table.timestamp_column",
        "description": "timestamp /* 5.5 binary format */",
        "dbObjectType": "Column"
      }
  ]
},
```

Viene segnalato un errore per la colonna `timestamp_column` della tabella`test.55_temporal_table`, poiché utilizza un vecchio formato di archiviazione su disco temporale che non è più supportato.

Per risolvere questo problema e consentire il proseguimento dell'aggiornamento, ricostruisci la tabella per convertire il vecchio formato di archiviazione su disco temporale in quello nuovo introdotto in MySQL 5.6. Per ulteriori informazioni e prerequisiti prima di procedere, consulta Conversione tra set di caratteri Unicode a 3 e 4 byte nella documentazione di MySQL.

L'esecuzione del comando seguente per ricostruire le tabelle menzionate in questo precontrollo converte i vecchi tipi temporali nel formato più recente con una precisione di frazioni di secondo.

```
ALTER TABLE ... ENGINE=InnoDB;
```

Per ulteriori informazioni sulla ricostruzione delle tabelle, vedere l'istruzione ALTER TABLE nella documentazione di MySQL.

Dopo aver ricostruito la tabella in questione e riavviato l'aggiornamento, il controllo di compatibilità viene superato e l'aggiornamento può procedere.

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "detectedProblems": []
}
```

partitionedTablesInCondiviso TablespaceCheck

Livello di precontrollo: errore

Utilizzo di tabelle partizionate in tablespace condivisi

A partire da MySQL 8.0.13, il supporto per l'inserimento di partizioni di tabelle in tablespace condivisi è stato rimosso. Prima dell'aggiornamento, sposta tali tabelle dai tablespace condivisi ai tablespace. file-per-table

> **Note**
>
> Prima di ricostruire i tablespace, consulta Operazioni di partizionamento nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

Esempio di output:

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "description": "Error: The following tables have partitions in shared tablespaces.
 They need to be moved to file-per-table tablespace before upgrading. You can do
 this by running query like 'ALTER TABLE table_name REORGANIZE PARTITION X INTO
 (PARTITION X VALUES LESS THAN (30) TABLESPACE=innodb_file_per_table);'",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partInnoDBTable",
        "description": "Partition p1 is in shared tablespace innodb",
```

```
        "dbObjectType": "Table"
      }
  ]
}
```

Il precontrollo fallisce perché la partizione della `p1` tabella `test.partInnoDBTable` si trova nel tablespace del sistema.

Per risolvere questo problema, ricostruite la `test.partInnodbTable` tabella inserendo la partizione che causa il problema in una tablespace. `p1` file-per-table

```
mysql > ALTER TABLE partInnodbTable REORGANIZE PARTITION p1
    ->   INTO (PARTITION p1 VALUES LESS THAN ('2014-01-01')
 TABLESPACE=innodb_file_per_table);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Dopo averlo fatto, il precontrollo passa.

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "detectedProblems": []
}
```

removedFunctionsCheck

Livello di precontrollo: errore

Utilizzo di funzioni rimosse dall'ultima versione di MySQL

In MySQL 8.0, sono state rimosse diverse funzioni integrate. Questo controllo preliminare esamina il database alla ricerca di oggetti che potrebbero utilizzare queste funzioni. Se vengono trovati, viene restituito un errore. È necessario risolvere i problemi prima di riprovare l'aggiornamento.

La maggior parte delle funzioni rimosse sono funzioni spaziali, che sono state sostituite con funzioni equivalentiST_*. In questi casi, si modificano gli oggetti del database per utilizzare la nuova denominazione delle procedure. Per ulteriori informazioni, consulta Features Removed in MySQL 8.0 nella documentazione MySQL.

Esempio di output:

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "description": "Error: The following DB objects use functions that were removed
 in the latest MySQL version. Please make sure to update them to use supported
 alternatives before upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.GetLocationsInPolygon",
        "description": "PROCEDURE uses removed function POLYGONFROMTEXT (consider
 using ST_POLYGONFROMTEXT instead)",
        "dbObjectType": "Routine"
      },
      {
        "level": "Error",
        "dbObject": "test.InsertLocation",
        "description": "PROCEDURE uses removed function POINTFROMTEXT (consider
 using ST_POINTFROMTEXT instead)",
        "dbObjectType": "Routine"
      }
  ]
},
```

Il precontrollo riporta che la `test.GetLocationsInPolygon` stored procedure utilizza due funzioni rimosse: POLYGONFROMTEXT e POINTFROMTEXT. Suggerisce inoltre di utilizzare i nuovi ST_POLYGONFROMTEXT e ST_POINTFROMTEXT come sostituti. Dopo aver ricreato la procedura utilizzando i suggerimenti, il precontrollo viene completato correttamente.

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "detectedProblems": []
},
```

> **Note**
>
> Sebbene nella maggior parte dei casi le funzioni obsolete siano sostituite direttamente, assicuratevi di testare l'applicazione e di consultare la documentazione per eventuali modifiche di comportamento dovute alla modifica.

routineSyntaxCheck

Livello di precontrollo: errore

Controllo della sintassi MySQL per oggetti simili a routine

MySQL 8.0 ha introdotto parole chiave riservate che non erano riservate in precedenza. Il prechecker di aggiornamento valuta l'utilizzo di parole chiave riservate nei nomi degli oggetti del database e nelle relative definizioni e nel corpo. Se rileva l'uso di parole chiave riservate negli oggetti del database, ad esempio stored procedure, funzioni, eventi e trigger, l'aggiornamento non riesce e viene pubblicato un errore nel file. `upgrade-prechecks.log` Per risolvere il problema, è necessario aggiornare le definizioni degli oggetti e racchiudere tali riferimenti tra virgolette singole (') prima dell'aggiornamento. Per ulteriori informazioni sull'escape delle parole riservate in MySQL, consulta String literals nella documentazione di MySQL.

In alternativa, è possibile modificare il nome con un nome diverso, il che potrebbe richiedere modifiche all'applicazione.

Esempio di output:

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "description": "The following objects did not pass a syntax check with the latest
 MySQL grammar. A common reason is that they reference names that conflict with new
 reserved keywords. You must update these routine definitions and `quote` any such
 references before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
         "level": "Error",
         "dbObject": "test.select_res_word",
```

```
         "description": "at line 2,18: unexpected token 'except'",
         "dbObjectType": "Routine"
      }
  ]
}
```

Per risolvere questo problema, controlla la definizione della routine.

```
SHOW CREATE PROCEDURE test.select_res_word\G

*************************** 1. row ***************************
           Procedure: select_res_word
            sql_mode:
 ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_Z
    Create Procedure: CREATE PROCEDURE 'select_res_word'()
BEGIN
    SELECT * FROM except;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

La procedura utilizza una tabella denominata`except`, che è una nuova parola chiave in MySQL 8.0. Ricrea la procedura evitando il valore letterale della stringa.

```
> drop procedure if exists select_res_word;
Query OK, 0 rows affected (0.00 sec)

> DELIMITER $$
 > CREATE PROCEDURE select_res_word()
    -> BEGIN
    ->     SELECT * FROM 'except';
    -> END$$
Query OK, 0 rows affected (0.00 sec)

 > DELIMITER ;
```

Il precontrollo ora passa.

```
{
```

```
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "detectedProblems": []
}
```

schemaInconsistencyCheck

Livello di precontrollo: errore

Incoerenze dello schema derivanti dalla rimozione o dal danneggiamento dei file

Come descritto in precedenza, MySQL 8.0 ha introdotto l'Atomic Data Dictionary, che memorizza tutti i metadati in un set di tabelle InnoDB interne nello schema. `mysql` Questa nuova architettura offre un modo transazionale e conforme agli standard ACID per gestire i metadati del database, risolvendo il problema della «DDL atomica» derivante dal vecchio approccio basato su file. Prima di MySQL 8.0, era possibile che gli oggetti dello schema diventassero orfani se un'operazione DDL veniva interrotta inaspettatamente. La migrazione dei metadati basati su file nelle nuove tabelle Atomic Data Dictionary durante l'aggiornamento garantisce che non vi siano tali oggetti dello schema orfani nell'istanza DB. Se vengono rilevati oggetti orfani, l'aggiornamento non riesce.

Per aiutare a rilevare questi oggetti orfani prima di iniziare l'aggiornamento, viene eseguito il `schemaInconsistencyCheck` precontrollo per garantire che tutti gli oggetti di metadati del dizionario di dati siano sincronizzati. Se vengono rilevati oggetti di metadati orfani, l'aggiornamento non procede. Per procedere con l'aggiornamento, pulite questi oggetti di metadati orfani.

Se riscontri errori con questo precontrollo, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

Esempio di output:

```
{
  "id": "schemaInconsistencyCheck",
  "title": "Schema inconsistencies resulting from file removal or corruption",
  "status": "OK",
  "description": "Error: Following tables show signs that either table datadir
 directory or frm file was removed/corrupted. Please check server logs, examine
 datadir to detect the issue and fix it before upgrade",
```

```
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.schemaInconsistencyCheck_failure",
        "description": "present in INFORMATION_SCHEMA's INNODB_SYS_TABLES table but
 missing from TABLES table"
      }
  ]
}
```

Il precontrollo riporta che la `test.schemaInconsistencyCheck_failure` tabella contiene metadati incoerenti. In questo caso, la tabella esiste nei metadati del motore di archiviazione InnoDB (`information_schema.INNODB_SYS_TABLES`), ma non nei metadati MySQL (). `information_schema.TABLES`

Precontrolli Aurora MySQL che segnalano errori

I seguenti controlli preliminari sono specifici di Aurora MySQL:

- Aurora Check DDLRecovery
- auroraCheckRdsUpgradePrechecksTable
- Aurora Controlla FODUpgrade
- auroraGetDanglingFulltextIndex
- auroraUpgradeCheckForDatafilePathInconsistency
- auroraUpgradeCheckForFtsSpaceIdZero
- auroraUpgradeCheckForIncompleteXATransactions
- auroraUpgradeCheckForInstanceLimit
- auroraUpgradeCheckForInternalUsers
- auroraUpgradeCheckForMasterUser
- auroraUpgradeCheckForPrefixIndexOnGeometryColumns
- auroraUpgradeCheckForSpecialCharactersInProcedures
- auroraUpgradeCheckForSysSchemaObjectTypeMismatch
- auroraUpgradeCheckForViewColumnNameLength
- auroraUpgradeCheckIndexLengthLimitOnTinyColumns

- auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable
- auroraUpgradeCheckForInvalidUtf8 mb3 ColumnComments

Controllo Aurora DDLRecovery

Livello di precontrollo: errore

Verifica la presenza di artefatti relativi alla funzione di ripristino Aurora DDL

Come parte dell'implementazione di ripristino DDL (Data Definition Language) in Aurora MySQL, i metadati sulle istruzioni DDL in volo vengono mantenuti nelle tabelle e nello schema. `ddl_log_md_table` `ddl_log_table` `mysql` L'implementazione di Aurora del ripristino DDL non è supportata dalla versione 3 in poi, poiché la funzionalità fa parte della nuova implementazione di Atomic Data Dictionary in MySQL 8.0. Se durante i controlli di compatibilità sono in esecuzione delle istruzioni DDL, questo precontrollo potrebbe non riuscire. Si consiglia di provare l'aggiornamento mentre non è in esecuzione alcuna istruzione DDL.

Se questo precontrollo fallisce senza alcuna istruzione DDL in esecuzione, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

Se sono in esecuzione delle istruzioni DDL, l'output del precheck stampa il seguente messaggio:

```
“There are DDL statements in process. Please allow DDL statements to finish before
 upgrading.”
```

Esempio di output:

```
{
  "id": "auroraCheckDDLRecovery",
  "title": "Check for artifacts related to Aurora DDL recovery feature",
  "status": "OK",
  "description": "Aurora implementation of DDL recovery is not supported from 3.x
 onwards. This check verifies that the database do not have artifacts realted to the
 feature",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_md_table",
```

```
        "description": "Table mysql.ddl_log_md_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_table",
        "description": "Table mysql.ddl_log_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "information_schema.processlist",
        "description": "There are DDL statements in process. Please allow DDL
 statements to finish before upgrading."
      }
  ]
}
```

Il precontrollo ha restituito un errore dovuto a un DDL in volo eseguito contemporaneamente ai controlli di compatibilità. Si consiglia di riprovare l'aggiornamento senza che venga eseguito. DDLs

auroraCheckRdsUpgradePrechecksTable

Livello di precontrollo: errore

Verifica l'esistenza della tabella **`mysql.rds_upgrade_prechecks`**

Si tratta di un controllo preliminare esclusivamente interno eseguito dal servizio RDS. Eventuali errori verranno gestiti automaticamente durante l'aggiornamento e possono essere tranquillamente ignorati.

Se riscontri errori con questo precontrollo, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

```
{
  "id": "auroraCheckRdsUpgradePrechecksTable",
  "title": "Check existence of mysql.rds_upgrade_prechecks table",
  "status": "OK",
  "detectedProblems": []
}
```

aurora Check FODUpgrade

Livello di precontrollo: errore

Verifica la presenza di artefatti relativi alla funzione Aurora fast DDL

L'ottimizzazione Fast DDL è stata introdotta in modalità lab su Aurora MySQL versione 2 per migliorare l'efficienza di alcune operazioni DDL. In Aurora MySQL versione 3, la modalità lab è stata rimossa e l'implementazione Fast DDL è stata sostituita dalla funzionalità MySQL 8.0 denominata Instant DDL.

Prima dell'aggiornamento ad Aurora MySQL versione 3, tutte le tabelle che utilizzano Fast DDL in modalità lab dovranno essere ricostruite eseguendo il comando `OPTIMIZE TABLE` o per garantire la compatibilità con Aurora MySQL versione 3. `ALTER TABLE ... ENGINE=InnoDB`

Questo precontrollo restituisce un elenco di tali tabelle. Dopo che le tabelle restituite sono state ricostruite, è possibile riprovare a eseguire l'aggiornamento.

Esempio di output:

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "description": "Aurora fast DDL is not supported from 3.x onwards. This check
 verifies that the database does not have artifacts related to the feature",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/AuroraMySQL.Managing.FastDDL.html#AuroraMySQL.Managing.FastDDL-v2",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test",
        "description": "Your table has pending Aurora fast DDL operations. Run
 'OPTIMIZE TABLE <table name>' for the table to apply all the pending DDL updates.
 Then try the upgrade again."
      }
  ]
}
```

Il precontrollo riporta che nella tabella sono presenti operazioni `test.test` Fast DDL in sospeso.

Per consentire il proseguimento dell'aggiornamento, è possibile ricostruire la tabella, quindi ritentare l'aggiornamento.

> **Note**
>
> Prima di ricostruire i tablespace, consulta le operazioni DDL online nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
mysql> optimize table test.test;
+-----------+----------+----------
+-------------------------------------------------------------------+
| Table     | Op       | Msg_type | Msg_text
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
| test.test | optimize | note     | Table does not support optimize, doing recreate
 + analyze instead |
| test.test | optimize | status   | OK
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.04 sec)
```

Dopo aver ricostruito la tabella, il precontrollo ha esito positivo.

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "detectedProblems": []
}
```

auroraGetDanglingFulltextIndex

Livello di precontrollo: errore

Tabelle con riferimento all'indice pendente **FULLTEXT**

Prima di MySQL 5.6.26, era possibile che, dopo aver eliminato un indice di ricerca completo, le colonne nascoste e quelle nascoste diventassero orfane. `FTS_DOC_ID` `FTS_DOC_ID_INDEX` Per ulteriori informazioni, vedere Bug #76012.

Se sono presenti tabelle create su versioni precedenti di MySQL in cui ciò si è verificato, l'aggiornamento ad Aurora MySQL versione 3 può fallire. Questo precontrollo verifica che nel cluster DB non esistano tali indici full-text orfani o «sospesi» prima dell'aggiornamento a MySQL 8.0. Se questo precontrollo fallisce, ricostruisci tutte le tabelle che contengono tali indici full-text sospesi.

Esempio di output:

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "description": "Error: The following tables contain dangling FULLTEXT index which
 is not supported. It is recommended to rebuild the table before upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.table_with_fts_index",
        "description": "Table `test.table_with_fts_index` contains dangling FULLTEXT
 index. Kindly recreate the table before upgrade."
      }
  ]
},
```

Il precontrollo riporta un errore per la `test.table_with_fts_index` tabella perché contiene un indice di testo completo sospeso. Per consentire il proseguimento dell'aggiornamento, ricostruite la tabella per ripulire le tabelle ausiliarie con indice full-text. Utilizza `OPTIMIZE TABLE test.table_with_fts_index` o `ALTER TABLE test.table_with_fts_index, ENGINE=INNODB`.

Dopo aver ricostruito la tabella, il precontrollo viene superato.

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForDatafilePathInconsistency

Livello di precontrollo: errore

Verifica l'eventuale presenza di incongruenze relative al **`ibd`** percorso del file

Questo controllo preliminare si applica solo a Aurora MySQL versione 3.03.0 e precedenti. Se si verifica un errore con questo precontrollo, esegui l'aggiornamento a Aurora MySQL versione 3.04 o successiva.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForFtsSpaceIdZero

Livello di precontrollo: errore

Verifica la presenza di un indice di testo completo con ID di spazio pari a zero

In MySQL, quando si aggiunge un indice full-text a una tabella InnoDB, vengono creati numerosi tablespace di indici ausiliari. A causa di un bug nelle versioni precedenti di MySQL, corretto nella versione 5.6.20, era possibile che queste tabelle di indici ausiliarie fossero state create nel tablespace di sistema, anziché nel proprio tablespace InnoDB.

Se esistono tablespace ausiliarie di questo tipo, l'aggiornamento avrà esito negativo. Ricreate gli indici full-text menzionati nell'errore di precheck, quindi riprovate a eseguire l'aggiornamento.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForFtsSpaceIdZero",
  "title": "Check for fulltext index with space id as zero",
  "status": "OK",
  "description": "The auxiliary tables of FTS indexes on the table are created
 in system table-space. Due to this DDL queries executed on MySQL8.0 shall cause
 database unavailability. To avoid that, drop and recreate all the FTS indexes on
 the table or rebuild the table using ALTER TABLE query before the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
```

```
        "dbObject": "test.fts_space_zero_check",
        "description": " The auxiliary tables of FTS indexes on the table
 'test.fts_space_zero_check' are created in system table-space due to https://
bugs.mysql.com/bug.php?id=72132. In MySQL8.0, DDL queries executed on this table
 shall cause database unavailability. To avoid that, drop and recreate all the
 FTS indexes on the table or rebuild the table using ALTER TABLE query before the
 upgrade."
      }
  ]
},
```

Il precontrollo riporta un errore per la `test.fts_space_zero_check` tabella, poiché nel tablespace di sistema sono presenti tabelle ausiliarie di ricerca full-text (FTS).

Dopo aver eliminato e ricreato gli indici FTS associati a questa tabella, il precontrollo ha esito positivo.

> **Note**
>
> Prima di ricostruire i tablespace, consulta Operazioni di partizionamento nella documentazione di MySQL per comprendere gli effetti del blocco e dello spostamento dei dati sulle transazioni in primo piano.

```
{
 "id": "auroraUpgradeCheckForFtsSpaceIdZero",
 "title": "Check for fulltext index with space id as zero",
 "status": "OK",
 "detectedProblems": []
}
```

auroraUpgradeCheckForIncompleteXATransactions

Livello di precontrollo: errore

Verifica la presenza di transazioni XA in stato di preparazione

Durante l'esecuzione del processo di aggiornamento della versione principale, è essenziale che l'istanza DB Aurora MySQL versione 2 sia sottoposta a un arresto completo. Ciò garantisce che tutte le transazioni vengano eseguite o ripristinate e che InnoDB abbia eliminato tutti i record del registro di annullamento. Poiché il rollback delle transazioni è necessario, se il database contiene

transazioni XA in uno stato preparato, può impedire il proseguimento del clean shutdown. Per questo motivo, se vengono rilevate transazioni XA preparate, l'aggiornamento non potrà procedere finché non si interverrà per confermarle o ripristinarle.

Per ulteriori informazioni sulla ricerca di transazioni XA in uno stato preparato utilizzando`XA RECOVER`, consulta le istruzioni SQL delle transazioni XA nella documentazione di MySQL. Per ulteriori informazioni sugli stati delle transazioni XA, consulta gli stati delle transazioni XA nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "description": "Your cluster currently has XA transactions in the prepared state.
 To proceed with the upgrade, commit or rollback these transactions.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your cluster currently has XA transactions in the prepared
 state. To proceed with the upgrade, commit or rollback these transactions."
      }
  ]
}
```

Questo precontrollo riporta un errore perché ci sono transazioni in uno stato preparato che devono essere eseguite o ripristinate.

Dopo aver effettuato l'accesso al database, puoi controllare la tabella information_schema.innodb_trx e l'output per ulteriori informazioni. `XA RECOVER`

> **Important**
>
> Prima di eseguire il commit o il rollback di una transazione, ti consigliamo di consultare la documentazione di MySQL e i requisiti dell'applicazione.

```
mysql> select trx_started,
    trx_mysql_thread_id,
```

```
    trx_id,trx_state,
    trx_operation_state,
    trx_rows_modified,
    trx_rows_locked
from
    information_schema.innodb_trx;
+---------------------+--------------------+---------+-----------
+---------------------+-------------------+-----------------+
| trx_started         | trx_mysql_thread_id | trx_id  | trx_state |
 trx_operation_state | trx_rows_modified | trx_rows_locked |
+---------------------+--------------------+---------+-----------
+---------------------+-------------------+-----------------+
| 2024-08-12 01:09:39 |                   0 | 2849470 | RUNNING   | NULL
    |                 1 |              0 |
+---------------------+--------------------+---------+-----------
+---------------------+-------------------+-----------------+
1 row in set (0.00 sec)

mysql> xa recover;
+----------+--------------+--------------+--------+
| formatID | gtrid_length | bqual_length | data   |
+----------+--------------+--------------+--------+
|        1 |            6 |            0 | xatest |
+----------+--------------+--------------+--------+
1 row in set (0.00 sec)
```

In questo caso, ripristiniamo la transazione preparata.

```
mysql> XA ROLLBACK 'xatest';
Query OK, 0 rows affected (0.00 sec)
v
mysql> xa recover;
Empty set (0.00 sec)
```

Dopo il rollback della transazione XA, il precontrollo ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInstanceLimit

Livello di precontrollo: errore

Verifica se l'aggiornamento è supportato nella classe di istanza corrente

L'esecuzione di un aggiornamento sul posto da Aurora MySQL versione 2.12.0 o 2.12.1, in cui la classe di istanza Writer DB è db.r6i.32xlarge, attualmente non è supportata. In questo caso, il precontrollo restituisce un errore. Per consentire il proseguimento dell'aggiornamento, puoi modificare la classe dell'istanza DB in db.r6i.24xlarge o inferiore. Oppure puoi eseguire l'aggiornamento ad Aurora MySQL versione 2.12.2 o successiva, dove l'aggiornamento in loco ad Aurora MySQL versione 3 è supportato su db.r6i.32xlarge.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForInstanceLimit",
  "title": "Checks if upgrade is supported on the current instance class",
  "status": "OK",
  "description": "Upgrade from Aurora Version 2.12.0 and 2.12.1 may fail for 32.xl
 and above instance class.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Upgrade is not supported on this instance size for Aurora
 MySql Version 2.12.1. Before upgrading to Aurora MySql 3, please consider either:
 1. Changing the instance class to 24.xl or lower. -or- 2. Upgrading to patch
 version 2.12.2 or higher."
      }
  ]
},
```

Il precontrollo restituisce un errore perché l'istanza Writer DB utilizza la classe di istanza db.r6i.32xlarge ed è in esecuzione su Aurora MySQL versione 2.12.1.

auroraUpgradeCheckForInternalUsers

Livello di precontrollo: errore

Verifica la presenza di utenti interni 8.0

Questo controllo preliminare si applica solo a Aurora MySQL versione 3.03.0 e precedenti. Se si verifica un errore con questo precontrollo, esegui l'aggiornamento a Aurora MySQL versione 3.04 o successiva.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForInternalUsers",
  "title": "Check for 8.0 internal users.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForMasterUser

Livello di precontrollo: errore

Verifica se esiste un utente principale RDS

MySQL 8 ha aggiunto un nuovo modello di privilegi con supporto per i ruoli e i privilegi dinamici per rendere la gestione dei privilegi più semplice e dettagliata. Come parte di questa modifica, Aurora MySQL ha introdotto il nuovo`rds_superuser_role`, che viene automaticamente concesso all'utente principale del database al momento dell'aggiornamento da Aurora MySQL versione 2 alla versione 3.

Per ulteriori informazioni sui ruoli e i privilegi assegnati all'utente master in Aurora MySQL, vedere. Privilegi dell'account utente master Per ulteriori informazioni sul modello di privilegi basato sui ruoli in Aurora MySQL versione 3, vedere. Privilegio basato sui ruoli

Questo precontrollo verifica che l'utente principale esista nel database. Se l'utente principale non esiste, il precontrollo ha esito negativo. Per consentire il proseguimento dell'aggiornamento, ricreate l'utente master reimpostando la password dell'utente principale o creando manualmente l'utente. Quindi riprova a eseguire l'aggiornamento. Per ulteriori informazioni sulla reimpostazione della password dell'utente principale, vedere. Modifica della password per l'utente master del database

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
```

```
  "description": "Throws error if master user has been dropped!",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/UsingWithRDS.MasterAccounts.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your Master User on host '%' has been dropped. To proceed
 with the upgrade, recreate the master user `reinvent` on default host '%'"
      }
  ]
}
```

Dopo aver reimpostato la password dell'utente principale, il precontrollo verrà superato e potrai riprovare a eseguire l'aggiornamento.

L'esempio seguente utilizza il AWS CLI per reimpostare la password. Le modifiche alla password vengono applicate immediatamente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --master-user-password my-new-password
```

Quindi il precontrollo ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForPrefixIndexOnGeometryColumns

Livello di precontrollo: errore

Verifica la presenza di colonne geometriche negli indici dei prefissi

A partire da MySQL 8.0.12, non è più possibile creare un indice con prefisso su una colonna utilizzando il tipo di dati GEOMETRY. Per ulteriori informazioni, vedere WL #11808.

Se esistono indici di questo tipo, l'aggiornamento avrà esito negativo. Per risolvere il problema, elimina gli `GEOMETRY` indici con prefisso nelle tabelle menzionate nell'errore di precontrollo.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "description": "Consider dropping the prefix Indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.geom_index_prefix",
        "description": "Table `test`.`geom_index_prefix` has an index `LatLon` on
 geometry column/s. Mysql 8.0 does not support this type of index on a geometry
 column https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run
 'DROP INDEX `LatLon` ON `test`.`geom_index_prefix`;"
      }
  ]
}
```

Il precontrollo riporta un errore perché la `test.geom_index_prefix` tabella contiene un indice con un prefisso su una `GEOMETRY` colonna.

Dopo aver eliminato questo indice, il precontrollo ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForSpecialCharactersInProcedures

Livello di precontrollo: errore

Verificare l'eventuale presenza di incongruenze relative ai caratteri speciali nelle procedure memorizzate

Prima di MySQL 8.0, i nomi dei database, i nomi delle tabelle e altri oggetti corrispondevano ai file nella directory dei dati, ovvero ai metadati basati su file. Come parte dell'aggiornamento a MySQL

8.0, tutti gli oggetti del database vengono migrati nelle nuove tabelle interne del dizionario dei dati che vengono archiviate nello schema per supportare `mysql` l'Atomic Data Dictionary appena implementato. Come parte della migrazione delle stored procedure, la definizione e il corpo della procedura per ciascuna procedura vengono convalidati man mano che vengono inseriti nel nuovo dizionario di dati.

Prima di MySQL 8, in alcuni casi era possibile creare routine memorizzate, o inserire direttamente nella tabella, procedure che `mysql.proc` contenevano caratteri speciali. Ad esempio, era possibile creare una stored procedure che contenesse un commento con il carattere di spazio non conforme e non divisibile. `\xa0` Se viene rilevata una di queste procedure, l'aggiornamento non riesce.

Questo precontrollo verifica che i corpi e le definizioni delle stored procedure non contengano tali caratteri. Per consentire il proseguimento dell'aggiornamento, ricreate queste stored procedure senza caratteri nascosti o speciali.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "description": "Following procedure(s) has special characters inconsistency.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "information_schema.routines",
        "description": "Data Dictionary Metadata is inconsistent for the procedure
 `get_version_proc` in the database `test` due to usage of special characters in
 procedure body. To avoid that, drop and recreate the procedure without any special
 characters before proceeding with the Upgrade."
      }
  ]
}
```

Il precheck riporta che il cluster DB contiene una procedura chiamata `get_version_proc` nel `test` database che contiene caratteri speciali nel corpo della procedura.

Dopo aver eliminato e ricreato la stored procedure, il precontrollo ha esito positivo e consente di procedere con l'aggiornamento.

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForSysSchemaObjectTypeMismatch

Livello di precontrollo: errore

Verifica che il tipo di oggetto non corrisponda allo schema **`sys`**

Lo schema sys è un insieme di oggetti e viste in un database MySQL che aiuta gli utenti a risolvere i problemi, ottimizzare e monitorare le proprie istanze DB. Quando si esegue un aggiornamento della versione principale da Aurora MySQL versione 2 alla versione 3, le viste `sys` dello schema vengono ricreate e aggiornate alle nuove definizioni di Aurora MySQL versione 3.

Come parte dell'aggiornamento, se alcuni oggetti `sys` dello schema vengono definiti utilizzando motori di archiviazione (`sys_config/BASE TABLE`in INFORMATION_SCHEMA.TABLES), anziché viste, l'aggiornamento avrà esito negativo. Tali tabelle sono disponibili nella tabella. `information_schema.tables` Questo non è un comportamento previsto, ma in alcuni casi può verificarsi a causa di un errore dell'utente.

Questo precontrollo convalida tutti gli oggetti `sys` dello schema per garantire che utilizzino le definizioni di tabella corrette e che le viste non vengano erroneamente definite come tabelle InnoDB o MyISAM. Per risolvere il problema, correggi manualmente gli oggetti restituiti rinominandoli o eliminandoli. Quindi riprova a eseguire l'aggiornamento.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "description": "Database contains objects with type mismatch for sys schema.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "sys.waits_global_by_latency",
```

```
        "description": "Your object sys.waits_global_by_latency has a type mismatch.
 To fix the inconsistency we recommend to rename or remove the object before
 upgrading (use RENAME TABLE command). "
      }
  ]
}
```

Il precheck riporta che la vista sys.waits_global_by_latency `sys` nello schema presenta una mancata corrispondenza di tipo che impedisce il proseguimento dell'aggiornamento.

Dopo aver effettuato l'accesso all'istanza DB, puoi vedere che questo oggetto è definito come una tabella InnoDB, quando dovrebbe essere una vista.

```
mysql> show create table sys.waits_global_by_latency\G
*************************** 1. row ***************************
       Table: waits_global_by_latency
Create Table: CREATE TABLE `waits_global_by_latency` (
  `events` varchar(128) DEFAULT NULL,
  `total` bigint(20) unsigned DEFAULT NULL,
  `total_latency` text,
  `avg_latency` text,
  `max_latency` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Per risolvere questo problema possiamo eliminare e ricreare la vista con la definizione corretta o rinominarla. Durante il processo di aggiornamento, verrà creata automaticamente con la definizione di tabella corretta.

```
mysql> RENAME TABLE sys.waits_global_by_latency to sys.waits_global_by_latency_old;
Query OK, 0 rows affected (0.01 sec)
```

Dopo aver fatto ciò, il precontrollo passa.

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForViewColumnNameLength

Livello di precontrollo: errore

Controlla il limite superiore per il nome della colonna visualizzata

La lunghezza massima consentita del nome di colonna in MySQL è di 64 caratteri. Prima di MySQL 8.0, in alcuni casi era possibile creare una vista con un nome di colonna più grande di 64 caratteri. Se sull'istanza del database sono presenti viste di questo tipo, viene restituito un errore di precontrollo e l'aggiornamento avrà esito negativo. Per consentire il proseguimento dell'aggiornamento, è necessario ricreare le viste in questione, assicurandosi che la lunghezza delle colonne sia inferiore a 64 caratteri. Quindi riprova a eseguire l'aggiornamento.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "description": "Following view(s) has column(s) with length greater than 64.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject":
 "test.colname_view_test.col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pa
        "description": "View `test`.`colname_view_test`has column
 `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` with
 invalid column name length. To avoid Upgrade errors, view should be altered by
 renaming the column name so that its length is not 0 and doesn't exceed 64."
      }
  ]
}
```

Il precontrollo riporta che la `test.colname_view_test` vista contiene una colonna `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` che supera la lunghezza massima consentita della colonna di 64 caratteri.

Guardando la definizione della vista, possiamo vedere la colonna offensiva.

```
mysql> desc `test`.`colname_view_test`;
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
```

```
| Field                                                                 | Type        |
 Null | Key | Default | Extra |
+-----------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| col1                                                                  | varchar(20) |
 YES  |     | NULL    |       |
| col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad | int(11)     |
 YES  |     | NULL    |       |
+-----------------------------------------------------------------------+-------------
+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Per consentire l'aggiornamento di procedere, ricrea la vista, assicurandoti che la lunghezza della colonna non superi i 64 caratteri.

```
mysql> drop view `test`.`colname_view_test`;
Query OK, 0 rows affected (0.01 sec)

mysql> create view `test`.`colname_view_test`(col1, col2_nopad) as select inf,
 fodcol from test;
Query OK, 0 rows affected (0.01 sec)

mysql> desc `test`.`colname_view_test`;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| col1       | varchar(20) | YES  |     | NULL    |       |
| col2_nopad | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Dopo aver eseguito questa operazione, il precontrollo ha esito positivo.

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckIndexLengthLimitOnTinyColumns

Livello di precontrollo: errore

Verifica la presenza di tabelle con indici definiti con una lunghezza del prefisso superiore a 255 byte su colonne minuscole

Quando si crea un indice su una colonna utilizzando un tipo di dati binario in MySQL, è necessario aggiungere una lunghezza del prefisso nella definizione dell'indice. Prima di MySQL 8.0, in alcuni casi era possibile specificare una lunghezza del prefisso maggiore della dimensione massima consentita per tali tipi di dati. Un esempio è `TINYTEXT` and `TINYBL0B` columns, in cui la dimensione massima consentita dei dati è di 255 byte, ma erano consentiti prefissi di indice più grandi di questa. Per ulteriori informazioni, consulta i limiti di InnoDB nella documentazione di MySQL.

Se questo precontrollo fallisce, elimina l'indice incriminato o riduci la lunghezza del prefisso `TINYTEXT` e delle `TINYBL0B` colonne dell'indice a meno di 255 byte. Quindi riprova a eseguire l'aggiornamento.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "description": "Prefix length of the indexes defined on tiny columns cannot
 exceed 255 bytes. With utf8mb4 char set, this limits the prefix length supported
 upto 63 characters only. A larger prefix length was allowed in MySQL5.7 using
 innodb_large_prefix parameter. This parameter is deprecated in MySQL 8.0.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/innodb-limits.html,
 https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.tintxt_prefixed_index.col1",
        "description": "Index 'PRIMARY' on tinytext/tinyblob column `col1` of table
 `test.tintxt_prefixed_index` is defined with prefix length exceeding 255 bytes.
 Reduce the prefix length to <=255 bytes depending on character set used. For
 utf8mb4, it should be <=63."
      }
  ]
}
```

Il precontrollo riporta un errore per la `test.tintxt_prefixed_index` tabella, poiché ha un indice `PRIMARY` con un prefisso maggiore di 255 byte su una colonna TINYTEXT o TINYBLOB.

Guardando la definizione di questa tabella, possiamo vedere che la chiave primaria ha un prefisso 65 sulla colonna. TINYTEXT `col1` Poiché la tabella è definita utilizzando il set di `utf8mb4` caratteri, che memorizza 4 byte per carattere, il prefisso supera il limite di 255 byte.

```
mysql> show create table `test`.`tintxt_prefixed_index`\G
*************************** 1. row ***************************
       Table: tintxt_prefixed_index
Create Table: CREATE TABLE `tintxt_prefixed_index` (
  `col1` tinytext NOT NULL,
  `col2` tinytext,
  `col_id` tinytext,
  PRIMARY KEY (`col1`(65))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=DYNAMIC
1 row in set (0.00 sec)
```

La modifica del prefisso dell'indice su 63 durante l'utilizzo del set di `utf8mb4` caratteri consentirà di procedere con l'aggiornamento.

```
mysql> alter table `test`.`tintxt_prefixed_index` drop PRIMARY KEY, ADD  PRIMARY KEY
 (`col1`(63));
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Dopo aver eseguito questa operazione, il precontrollo ha esito positivo.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

Livello di precontrollo: errore

Verifica l'incoerenza dei metadati InnoDB mancanti per la tabella **`mysql.host`**

Si tratta di un controllo preliminare solo interno eseguito dal servizio RDS. Eventuali errori verranno gestiti automaticamente durante l'aggiornamento e possono essere tranquillamente ignorati.

Se riscontri errori con questo precontrollo, apri un caso con AWS Support per richiedere che l'incoerenza dei metadati venga risolta. In alternativa, puoi riprovare l'aggiornamento eseguendo un dump logico, quindi eseguendo il ripristino su un nuovo cluster DB Aurora MySQL versione 3.

auroraUpgradeCheckForInvalidUtf8 mb3 ColumnComments

Livello di precontrollo: errore

Verifica la presenza di commenti non validi nella colonna utf8mb3

Questo precontrollo identifica le tabelle che contengono commenti di colonna con codifica dei caratteri utf8mb3 non valida. In MySQL 8.0, viene applicata una convalida più rigorosa alla codifica dei caratteri nei metadati, inclusi i commenti delle colonne. Se un commento di colonna contiene caratteri che non sono validi nel set di caratteri utf8mb3, l'aggiornamento avrà esito negativo.

La causa più comune di questo problema è la presenza di caratteri non BMP (Basic Multilingual Plane) nei commenti delle colonne. I caratteri non BMP richiedono la codifica UTF-8 a 4 byte (utf8mb4), ma utf8mb3 supporta solo sequenze a 3 byte, che non possono rappresentare questi caratteri.

Per risolvere questo problema, è necessario modificare i commenti delle colonne per rimuovere o sostituire eventuali caratteri non BMP prima di tentare l'aggiornamento. È possibile utilizzare l'`ALTER TABLE`istruzione per aggiornare i commenti della colonna.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "description": "Following table(s) has/have invalid utf8mb3 comments on the
 column/columns.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.t2",
        "description": "Table test.t2 has invalid utf8mb3 comments in it's column/
columns. This is due to non-BMP characters in the comment field. To fix the
 inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
      }
```

```
  ]
}
```

Il precontrollo riporta che la `test.t2` tabella contiene caratteri utf8mb3 non validi in uno o più commenti di colonna, in particolare a causa della presenza di caratteri non BMP.

Per risolvere questo problema, è possibile identificare le colonne problematiche e aggiornarne i commenti. Innanzitutto, esamina la struttura della tabella per identificare le colonne con commenti:

```
mysql> SHOW CREATE TABLE test.t2\G
```

Dopo aver identificato le colonne con commenti problematici, aggiornale utilizzando l'`ALTER TABLE`istruzione. Per esempio:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT 'Updated
 comment without non-BMP characters';
```

In alternativa, puoi rimuovere completamente il commento:

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT '';
```

Dopo aver aggiornato tutti i commenti delle colonne problematiche, il precontrollo verrà superato e l'aggiornamento potrà procedere:

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "detectedProblems": []
}
```

> **Note**
>
> Prima di modificare i commenti delle colonne, assicuratevi che qualsiasi codice dell'applicazione o documentazione che si basa su questi commenti venga aggiornato di conseguenza. Prendete in considerazione la migrazione al set di caratteri utf8mb4 per un migliore supporto Unicode se l'applicazione richiede caratteri non BMP.

Avvisi

I seguenti precontrolli generano avvisi quando il precontrollo fallisce, ma l'aggiornamento può continuare.

Argomenti

- Precontrolli MySQL che segnalano avvisi
- Aurora MySQL precontrolli che segnalano gli avvisi

Precontrolli MySQL che segnalano avvisi

I seguenti controlli preliminari provengono da Community MySQL:

- defaultAuthenticationPlugin
- maxdbFlagCheck
- mysqlDollarSignNameCheck
- reservedKeywordsCheck
- Controllo UTF8 MB3
- zeroDatesCheck

defaultAuthenticationPlugin

Livello di precontrollo: avviso

Nuove considerazioni sul plug-in di autenticazione predefinito

In MySQL 8.0, è stato introdotto `caching_sha2_password` il plug-in di autenticazione, che fornisce una crittografia delle password più sicura e prestazioni migliori rispetto al plug-in obsoleto. `mysql_native_password` Per Aurora MySQL versione 3, il plug-in di autenticazione predefinito utilizzato per gli utenti del database è il plug-in. `mysql_native_password`

Questo precontrollo avverte che questo plugin verrà rimosso e l'impostazione predefinita verrà modificata in una futura versione principale. Valuta la possibilità di valutare la compatibilità dei client e degli utenti delle applicazioni prima di questa modifica.

Per ulteriori informazioni, consulta i problemi e le soluzioni di compatibilità con caching_sha2_password nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "defaultAuthenticationPlugin",
  "title": "New default authentication plugin considerations",
  "description": "Warning: The new default authentication plugin
 'caching_sha2_password' offers more secure password hashing than previously
 used 'mysql_native_password' (and consequent improved client connection
 authentication). However, it also has compatibility implications that
 may affect existing MySQL installations. If your MySQL installation
 must serve pre-8.0 clients and you encounter compatibility issues after
 upgrading, the simplest way to address those issues is to reconfigure
 the server to revert to the previous default authentication plugin
 (mysql_native_password). For example, use these lines in the server option file:
\n\n[mysqld]\ndefault_authentication_plugin=mysql_native_password\n\nHowever, the
 setting should be viewed as temporary, not as a long term or permanent solution,
 because it causes new accounts created with the setting in effect to forego the
 improved authentication security.\nIf you are using replication please take time to
 understand how the authentication plugin changes may impact you.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-caching-sha2-password-compatibility-issues\nhttps://
dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-
sha2-password-replication"
},
```

maxdbFlagCheck

Livello di precontrollo: avviso

Utilizzo di una bandiera obsoleta **`MAXDB` `sql_mode`**

In MySQL 8.0, sono state rimosse diverse opzioni di variabili di sistema sql_mode obsolete, una delle quali era. `MAXDB` Questo controllo preliminare esamina tutte le sessioni attualmente connesse, insieme alle routine e ai trigger, per garantire che nessuna sia impostata su alcuna combinazione che includa. `sql_mode` `MAXDB`

Esempio di output:

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
```

```
  "description": "Warning: The following DB objects have the obsolete MAXDB
 option persisted for sql_mode, which will be cleared during the upgrade. It can
 potentially change the datatype DATETIME into TIMESTAMP if it is used inside
 object's definition, and this in turn can change the behavior in case of dates
 earlier than 1970 or later than 2037. If this is a concern, please redefine these
 objects so that they do not rely on the MAXDB flag before running the upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.maxdb_stored_routine",
        "description": "PROCEDURE uses obsolete MAXDB sql_mode",
        "dbObjectType": "Routine"
      }
  ]
}
```

Il precheck riporta che la `test.maxdb_stored_routine` routine contiene un'opzione non supportata`sql_mode`.

Dopo aver effettuato l'accesso al database, è possibile visualizzare la definizione di routine che contiene. `sql_mode` MAXDB

```
 > SHOW CREATE PROCEDURE test.maxdb_stored_routine\G

*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode:
 PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,MAXDB,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPT
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Per risolvere il problema, ricrea la procedura dopo aver impostato quella corretta `sql_mode` sul client.

> **Note**
>
> Secondo la documentazione di MySQL, MySQL memorizza le impostazioni in vigore quando una `sql_mode` routine viene creata o modificata. Esegue sempre la routine con questa impostazione, indipendentemente dall'impostazione al momento dell'avvio della `sql_mode` routine.
> Prima di modificare`sql_mode`, consulta le modalità SQL del server nella documentazione di MySQL. Valuta attentamente qualsiasi potenziale impatto di questa operazione sulla tua applicazione.

Ricrea la procedura senza elementi non `sql_mode` supportati.

```
mysql > set sql_mode='PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql > DROP PROCEDURE test.maxdb_stored_routine\G
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER $$
mysql >
mysql > CREATE PROCEDURE test.maxdb_stored_routine()
    -> SQL SECURITY DEFINER
    -> BEGIN
    ->     SELECT * FROM test;
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER ;
mysql > show create procedure test.maxdb_stored_routine\G
*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
```

```
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Il precontrollo ha esito positivo.

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "detectedProblems": []
}
```

mysqlDollarSignNameCheck

Livello di precontrollo: avviso

Verifica l'uso obsoleto dei simboli del singolo dollaro nei nomi degli oggetti

A partire da MySQL 8.0.32, l'uso del simbolo del dollaro $ () come primo carattere di un identificatore non quotato è obsoleto. Se sono presenti schemi, tabelle, viste, colonne o routine che contengono un come primo carattere, questo precontrollo restituisce un avviso. $ Sebbene questo avviso non impedisca il proseguimento dell'aggiornamento, ti consigliamo di agire subito per risolvere il problema. Da MySQL 8.4 qualsiasi identificatore di questo tipo restituirà un errore di sintassi anziché un avviso.

Esempio di output:

```
{
  "id": "mysqlDollarSignNameCheck",
  "title": "Check for deprecated usage of single dollar signs in object names",
  "status": "OK",
  "description": "Warning: The following objects have names with deprecated usage
 of dollar sign ($) at the begining of the identifier. To correct this warning,
 ensure, that names starting with dollar sign, also end with it, similary to quotes
 ($example$). ",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.$deprecated_syntx",
        "description": " name starts with $ sign."
      }
  ]
```

```
},
```

Il precontrollo riporta un avviso perché la `$deprecated_syntx` tabella nello `test` schema contiene un $ come primo carattere.

reservedKeywordsCheck

Livello di precontrollo: avviso

Utilizzo di oggetti di database con nomi in conflitto con nuove parole chiave riservate

Questo controllo è simile al routineSyntaxCheck, in quanto verifica l'utilizzo di oggetti di database con nomi in conflitto con nuove parole chiave riservate. Sebbene non blocchino gli aggiornamenti, è necessario valutare attentamente gli avvisi.

Esempio di output:

Utilizzando l'esempio precedente con la tabella denominata`except`, il precontrollo restituisce un avviso:

```
{
  "id": "reservedKeywordsCheck",
  "title": "Usage of db objects with names conflicting with new reserved keywords",
  "status": "OK",
  "description": "Warning: The following objects have names that conflict with
 new reserved keywords. Ensure queries sent by your applications use `quotes` when
 referring to them or they will result in errors.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.except",
        "description": "Table name",
        "dbObjectType": "Table"
      }
  ]
}
```

Questo avviso segnala che potrebbero esserci alcune domande relative all'applicazione da esaminare. Se le query dell'applicazione non escono correttamente dalle stringhe letterali, potrebbero verificarsi degli errori dopo l'aggiornamento a MySQL 8.0. Esamina le tue applicazioni per confermarle, testale con un clone o un'istantanea del cluster Aurora MySQL DB in esecuzione sulla versione 3.

Esempio di una query applicativa senza escape che avrà esito negativo dopo l'aggiornamento:

```
SELECT * FROM escape;
```

Esempio di una query applicativa con escape corretto che avrà esito positivo dopo l'aggiornamento:

```
SELECT * FROM 'escape';
```

Controllo UTF8 MB3

Livello di precontrollo: avviso

Utilizzo del set di **`utf8mb3`** caratteri

In MySQL 8.0 `utf8mb3` il set di caratteri è obsoleto e verrà rimosso in una futura versione principale di MySQL. Questo precontrollo è implementato per generare un avviso se vengono rilevati oggetti di database che utilizzano questo set di caratteri. Anche se ciò non impedirà il proseguimento dell'aggiornamento, consigliamo vivamente di prendere in considerazione la migrazione delle tabelle al set di `utf8mb4` caratteri, che è l'impostazione predefinita a partire da MySQL 8.0. Per ulteriori informazioni su utf8mb3 e utf8mb4, vedere Conversione tra set di caratteri Unicode a 3 e 4 byte nella documentazione di MySQL.

Esempio di output:

```
{
  "id": "utf8mb3",
  "title": "Usage of utf8mb3 charset",
  "status": "OK",
  "description": "Warning: The following objects use the deprecated utf8mb3
 character set. It is recommended to convert them to use utf8mb4 instead, for
 improved Unicode support. The utf8mb3 character is subject to removal in the
 future.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/charset-unicode-
utf8mb3.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.t1.col1",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      },
```

```
      {
        "level": "Warning",
        "dbObject": "test.t1.col2",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      }
  ]
}
```

Per risolvere questo problema, ricostruite gli oggetti e le tabelle a cui si fa riferimento. Per ulteriori informazioni e prerequisiti prima di procedere, consulta Conversione tra set di caratteri Unicode a 3 e 4 byte nella documentazione di MySQL.

zeroDatesCheck

Livello di precontrollo: avviso

Zero valori di data, datetime e timestamp

MySQL ora applica regole più rigide per quanto riguarda l'uso di valori zero nelle colonne date, datetime e timestamp. Si consiglia di utilizzare le `NO_ZERO_DATE` SQL modalità `NO_ZERO_IN_DATE` and insieme a `strict` mode, poiché verranno unite a `strict` mode in una futura versione di MySQL.

Se l'`sql_mode`impostazione per una qualsiasi delle connessioni al database, al momento dell'esecuzione del precontrollo, non include queste modalità, viene generato un avviso nel precontrollo. Gli utenti potrebbero comunque essere in grado di inserire valori di data, datetime e timestamp contenenti valori zero. Tuttavia, consigliamo vivamente di sostituire i valori zero con valori validi, poiché il loro comportamento potrebbe cambiare in futuro e potrebbero non funzionare correttamente. Poiché si tratta di un avviso, non bloccherà gli aggiornamenti, ma ti consigliamo di iniziare a pianificare le azioni necessarie.

Esempio di output:

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
 longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
 in SQL_MODE by default. These modes should be used with strict mode as they will be
 merged with strict mode in a future release. If you do not include these modes in
 your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
```

```
 contain zeros. It is strongly advised to replace zero values with valid ones, as
 they may not work correctly in the future.",
  "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "global.sql_mode",
        "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
      },
      {
        "level": "Warning",
        "dbObject": "session.sql_mode",
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
  ]
}
```

Aurora MySQL precontrolli che segnalano gli avvisi

I seguenti controlli preliminari sono specifici di Aurora MySQL:

- auroraUpgradeCheckForRollbackSegmentHistoryLength
- auroraUpgradeCheckForUncommittedRowModifications

auroraUpgradeCheckForRollbackSegmentHistoryLength

Livello di precontrollo: avviso

Verifica se la lunghezza dell'elenco della cronologia dei segmenti di rollback per il cluster è elevata

Come indicato in auroraUpgradeCheckForIncompleteXATransactions, durante l'esecuzione del processo di aggiornamento della versione principale, è essenziale che l'istanza DB Aurora MySQL versione 2 venga arrestata da zero. Ciò garantisce che tutte le transazioni vengano eseguite o ripristinate e che InnoDB abbia eliminato tutti i record del registro di annullamento.

Se il tuo cluster DB ha un'elevata lunghezza dell'elenco della cronologia dei segmenti di rollback (HLL), può prolungare il tempo impiegato da InnoDB per completare l'eliminazione dei record di registro degli annullamenti, con conseguenti tempi di inattività prolungati durante il processo

di aggiornamento della versione principale. Se il precontrollo rileva che l'HLL sul cluster DB è elevato, genera un avviso. Sebbene ciò non impedisca il proseguimento dell'aggiornamento, ti consigliamo di monitorare attentamente l'HLL sul tuo cluster DB. Mantenerlo a livelli bassi riduce i tempi di inattività necessari durante l'aggiornamento di una versione principale. Per ulteriori informazioni sul monitoraggio di HLL, vedere. La lunghezza dell'elenco della cronologia di InnoDB è aumentata in modo significativo

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForRollbackSegmentHistoryLength",
  "title": "Checks if the rollback segment history length for the cluster is high",
  "status": "OK",
  "description": "Rollback Segment History length is greater than 1M. Upgrade may
 take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_metrics",
        "description": "The InnoDB undo history list length('trx_rseg_history_len')
 is 82989114. Upgrade may take longer due to purging of undo information for old row
 versions."
      }
  ]
}
```

Il precontrollo restituisce un avviso perché ha rilevato che l'HLL di annullamento di InnoDB era alto nel cluster di database (82989114). Sebbene l'aggiornamento proceda, a seconda della quantità di annullamenti da eliminare, può prolungare i tempi di inattività necessari durante il processo di aggiornamento.

Ti consigliamo di esaminare le transazioni aperte sul tuo cluster DB prima di eseguire l'aggiornamento in produzione, per assicurarti che l'HLL mantenga una dimensione gestibile.

auroraUpgradeCheckForUncommittedRowModifications

Livello di precontrollo: avviso

Verifica se ci sono molte modifiche non confermate alle righe

Come indicato in auroraUpgradeCheckForIncompleteXATransactions, durante l'esecuzione del processo di aggiornamento della versione principale, è essenziale che l'istanza DB Aurora MySQL

versione 2 venga arrestata da zero. Ciò garantisce che tutte le transazioni vengano eseguite o ripristinate e che InnoDB abbia eliminato tutti i record del registro di annullamento.

Se il tuo cluster DB ha transazioni che hanno modificato un gran numero di righe, può prolungare il tempo impiegato da InnoDB per completare un rollback di questa transazione come parte del processo di chiusura pulita. Se il precontrollo rileva transazioni di lunga durata, con un gran numero di righe modificate sul cluster DB, genera un avviso. Sebbene ciò non impedisca il proseguimento dell'aggiornamento, ti consigliamo di monitorare attentamente la dimensione delle transazioni attive sul tuo cluster DB. Mantenerlo a livelli bassi riduce i tempi di inattività necessari durante l'aggiornamento di una versione principale.

Esempio di output:

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "description": "Database contains uncommitted row changes greater than 10M.
 Upgrade may take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_trx",
        "description": "The database contains 11000000 uncommitted row change(s) in
 1 transaction(s). Upgrade may take longer due to transaction rollback."
      }
  ]
},
```

Il precheck riporta che il cluster DB contiene una transazione con 11.000.000 di modifiche di riga senza commit che dovranno essere ripristinate durante il processo di chiusura completa. L'aggiornamento continuerà, ma per ridurre i tempi di inattività durante il processo di aggiornamento, si consiglia di monitorarlo e analizzarlo prima di eseguire l'aggiornamento sui cluster di produzione.

Per visualizzare le transazioni attive sulla tua istanza Writer DB, puoi utilizzare la tabella information_schema.innodb_trx. La seguente query sull'istanza Writer DB mostra le transazioni correnti, il tempo di esecuzione, lo stato e le righe modificate per il cluster DB.

```
# Example of uncommitted transaction
```

```
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+------------------------------
+--------------------------------+----------+-----------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state | rows_modified |
+---------------------+------------------------------
+--------------------------------+----------+-----------+---------------+
| 2024-08-12 18:32:52 |                           1592 |
 20041 | 52866130 | RUNNING   |      11000000 |
+---------------------+------------------------------
+--------------------------------+----------+-----------+---------------+
1 row in set (0.01 sec)

# Example of transaction rolling back
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+------------------------------
+--------------------------------+----------+--------------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state    | rows_modified |
+---------------------+------------------------------
+--------------------------------+----------+--------------+---------------+
| 2024-08-12 18:32:52 |                           1719 |
 20041 | 52866130 | ROLLING BACK |      10680479 |
+---------------------+------------------------------
+--------------------------------+----------+--------------+---------------+
1 row in set (0.01 sec)
```

Dopo il commit o il rollback della transazione, il precontrollo non restituisce più un avviso. Consulta la documentazione di MySQL e il tuo team applicativo prima di eseguire il rollback di transazioni di grandi dimensioni, poiché il completamento del rollback può richiedere del tempo, a seconda delle dimensioni della transazione.

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "detectedProblems": []
},
```

Per ulteriori informazioni sull'ottimizzazione della gestione delle transazioni InnoDB e sul potenziale impatto dell'esecuzione e del rollback di transazioni di grandi dimensioni sulle istanze del database MySQL, vedere Ottimizzazione della gestione delle transazioni InnoDB nella documentazione MySQL.

Note

Il seguente precontrollo genera un avviso quando il precontrollo fallisce, ma l'aggiornamento può procedere.

sqlModeFlagVerificare

Livello di precontrollo: Avviso

Utilizzo di bandiere obsolete **`sql_mode`**

InoltreMAXDB, sono state rimosse `sql_mode` diverse altre opzioni:DB2,,, `MSSQLMYSQL323`,`MYSQL40`,`ORACLE`, `POSTGRESQL` `NO_FIELD_OPTIONSNO_KEY_OPTIONS`, e. `NO_TABLE_OPTIONS` A partire da MySQL 8.0, nessuno di questi valori può essere assegnato alla variabile di sistema. `sql_mode` Se questo precontrollo rileva sessioni aperte che utilizzano queste `sql_mode` impostazioni, assicurati che l'istanza DB e i gruppi di parametri del cluster DB, nonché le applicazioni e le configurazioni client, siano aggiornati per disabilitarle.- Per ulteriori informazioni, consulta la documentazione di MySQL.

Esempio di output:

```
{
  "id": "sqlModeFlagCheck",
  "title": "Usage of obsolete sql_mode flags",
  "status": "OK",
  "detectedProblems": []
}
```

Per risolvere uno di questi errori di precontrollo, vedere. maxdbFlagCheck

Errori, avvertenze o avvisi

Il seguente precontrollo può restituire un errore, un avviso o un avviso a seconda dell'output del precontrollo.

checkTableOutput

Livello di precontrollo: errore, avviso o avviso

Problemi segnalati dal comando **`check table x for upgrade`**

Prima di iniziare l'aggiornamento ad Aurora MySQL versione 3, `check table for upgrade` viene eseguito su ogni tabella degli schemi utente del cluster DB. Questo precontrollo non è lo stesso di Schema. checkTableMysql

Il `check table for upgrade` comando esamina le tabelle per individuare eventuali problemi che potrebbero insorgere durante l'aggiornamento a una versione più recente di MySQL. L'esecuzione di questo comando prima di tentare un aggiornamento può aiutare a identificare e risolvere eventuali incompatibilità in anticipo, rendendo più agevole l'effettivo processo di aggiornamento.

Questo comando esegue diversi controlli su ogni tabella, come i seguenti:

- Verifica della compatibilità della struttura della tabella e dei metadati con la versione di MySQL di destinazione
- Verifica della presenza di eventuali funzionalità obsolete o rimosse utilizzate dalla tabella
- Garantire che la tabella possa essere aggiornata correttamente senza perdita di dati

A differenza di altri controlli preliminari, può restituire un errore, un avviso o un avviso a seconda dell'`check table`output. Se questo precontrollo restituisce delle tabelle, esaminale attentamente, insieme al codice restituito e al messaggio prima di iniziare l'aggiornamento. Per ulteriori informazioni, vedere l'istruzione CHECK TABLE nella documentazione di MySQL.

Qui forniamo un esempio di errore e un esempio di avviso.

Esempio di errore:

```
{
  "id": "checkTableOutput",
```

```
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.parent",
        "description": "Table 'test.parent' doesn't exist"
      }
  ]
},
```

Il precontrollo riporta un errore indicante che la `test.parent` tabella non esiste.

Il `mysql-error.log` file per l'istanza Writer DB mostra che c'è un errore di chiave esterna.

```
2024-08-13T15:32:10.676893Z 62 [Warning] InnoDB: Load table `test`.`parent` failed,
 the table has missing foreign key indexes. Turn off 'foreign_key_checks' and try
 again.
2024-08-13T15:32:10.676905Z 62 [Warning] InnoDB: Cannot open table test/parent from
 the internal data dictionary of InnoDB though the .frm file for the table exists.
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-troubleshooting.html
 for how to resolve the issue.
```

Accedi all'istanza Writer DB ed `show engine innodb status\G` eseguila per ottenere maggiori informazioni sull'errore della chiave esterna.

```
mysql> show engine innodb status\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2024-08-13 15:33:33 0x14ef7b8a1700 INNODB MONITOR OUTPUT
=====================================
.
.
.
------------------------
LATEST FOREIGN KEY ERROR
------------------------
2024-08-13 15:32:10 0x14ef6dbbb700 Error in foreign key constraint of table test/
child:
there is no index in referenced table which would contain
```

```
the columns as the first columns, or the data types in the
referenced table do not match the ones in table. Constraint:
,
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
The index in the foreign key in table is p_name_idx
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-foreign-key-
constraints.html for correct foreign key definition.
.
.
```

Il `LATEST FOREIGN KEY ERROR` messaggio segnala che il vincolo di chiave `fk_pname` esterna nella `test.child` tabella, che fa riferimento alla `test.parent` tabella, presenta un indice mancante o una mancata corrispondenza del tipo di dati. La documentazione MySQL sui vincoli di chiave esterna afferma che le colonne a cui si fa riferimento in una chiave esterna devono avere un indice associato e le colonne padre/figlio devono utilizzare lo stesso tipo di dati.

Per verificare se ciò è correlato a una mancata corrispondenza dell'indice o del tipo di dati, accedi al database e controlla le definizioni della tabella disabilitando temporaneamente la variabile di sessione foreign_key_checks. Dopo averlo fatto, possiamo vedere che il vincolo secondario in question () `fk_pname` utilizza per fare riferimento alla tabella principale. `p_name varchar(20) CHARACTER SET latin1 DEFAULT NULL` `name varchar(20) NOT NULL` La tabella principale utilizza`DEFAULT CHARSET=utf8`, ma la `p_name` colonna della tabella secondaria utilizza`latin1`, quindi viene generato l'errore di mancata corrispondenza del tipo di dati.

```
mysql> show create table parent\G
ERROR 1146 (42S02): Table 'test.parent' doesn't exist

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> set foreign_key_checks=0;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show create table parent\G
*************************** 1. row ***************************
       Table: parent
Create Table: CREATE TABLE `parent` (
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Per risolvere questo problema, possiamo modificare la tabella figlio in modo che utilizzi lo stesso set di caratteri della tabella principale oppure modificare la tabella principale per utilizzare lo stesso set di caratteri della tabella figlio. Qui, poiché la tabella secondaria utilizza esplicitamente `latin1` nella definizione della `p_name` colonna, `ALTER TABLE` eseguiamo la modifica del set di caratteri in`utf8`.

```
mysql> alter table child modify p_name varchar(20) character set utf8 DEFAULT NULL;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> flush tables;
Query OK, 0 rows affected (0.01 sec)
```

Dopo averlo fatto, il precontrollo viene superato e l'aggiornamento può procedere.

Esempio di avviso:

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
```

```
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.orders",
        "description": "Trigger test.orders.delete_audit_trigg does not have CREATED
 attribute."
      }
  ]
}
```

Il precontrollo riporta un avviso per il `delete_audit_trigg` trigger sulla `test.orders` tabella perché non ha un `CREATED` attributo. In base a Checking version compatibility nella documentazione di MySQL, questo messaggio è informativo e viene stampato per i trigger creati prima di MySQL 5.7.2.

Poiché si tratta di un avviso, non impedisce il proseguimento dell'aggiornamento. Tuttavia, se desideri risolvere il problema, puoi ricreare il trigger in questione e il precontrollo ha esito positivo senza avvisi.

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": []
},
```

Come eseguire un aggiornamento in loco

Ti consigliamo di rivedere il materiale di background in Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco.

Esegui qualsiasi pianificazione e test prima dell'aggiornamento, come descritto in. Pianificazione di un aggiornamento della versione principale per un cluster Aurora MySQL

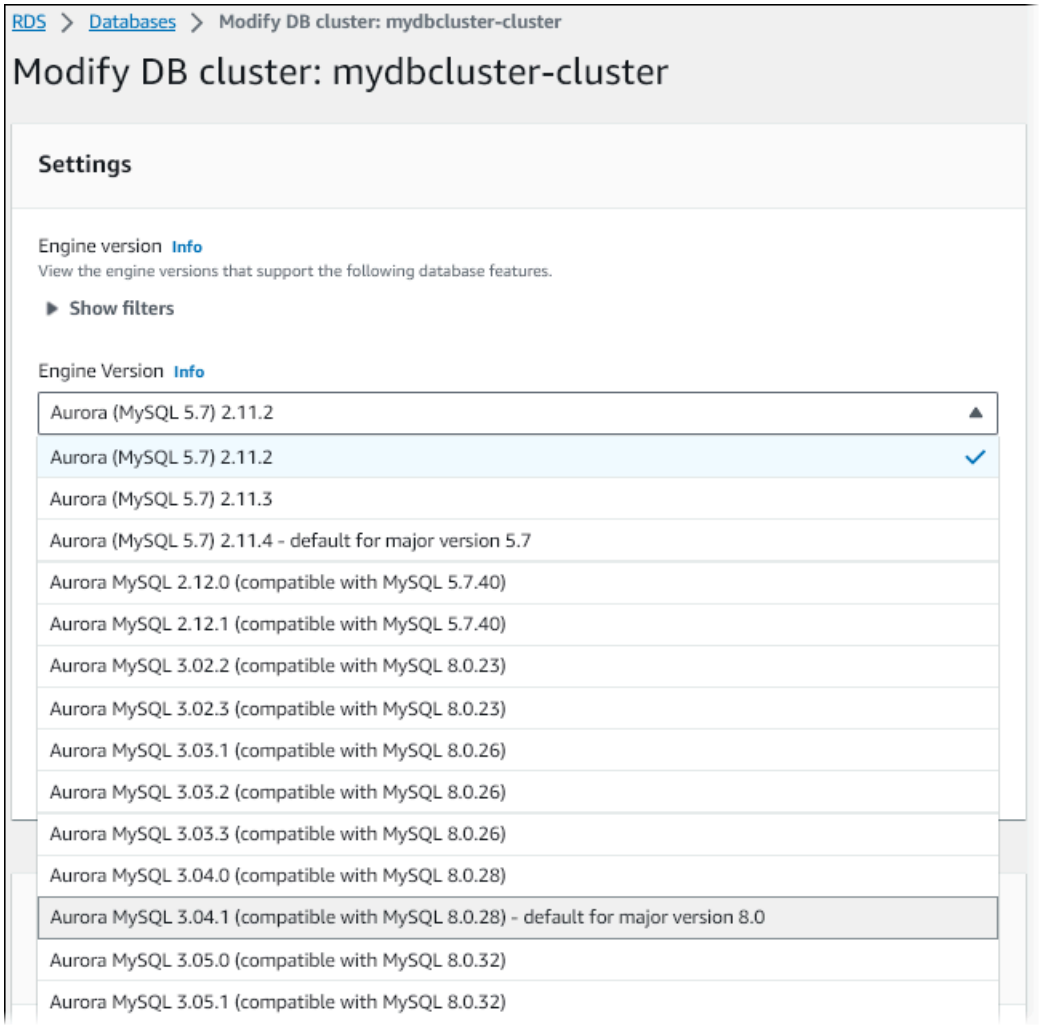
Console

L'esempio seguente aggiorna il cluster `mydbcluster-cluster` DB alla versione 3.04.1 di Aurora MySQL.

Per aggiornare la versione principale di un cluster di database Aurora MySQL

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Se è stato utilizzato un gruppo di parametri personalizzato con il cluster di database originale, crea un gruppo di parametri corrispondente compatibile con la nuova versione principale. Apportare le modifiche necessarie ai parametri di configurazione nel nuovo gruppo di parametri. Per ulteriori informazioni, consulta Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster.
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere il cluster di database che si desidera modificare.
5. Scegliere Modify (Modifica).
6. Per Versione, scegli una versione principale di Aurora MySQL.

   In genere consigliamo di utilizzare la versione secondaria più recente della versione principale. Qui, scegliamo la versione predefinita corrente.



7. Scegli Continue (Continua).
8. Nella pagina successiva specificare quando eseguire l'aggiornamento. Scegliere During the next scheduled maintenance window (Durante la successiva finestra di manutenzione programmata) o Immediately (Immediatamente).
9. (Facoltativo) Esaminare periodicamente la pagina Events (Eventi) nella console RDS durante l'aggiornamento. In questo modo è possibile monitorare lo stato di avanzamento dell'aggiornamento e identificare eventuali problemi. Se l'aggiornamento incontra dei problemi, consulta Risoluzione dei problemi relativi all'aggiornamento in-place di Aurora My SQL per la procedura da intraprendere.
10. Se all'inizio di questa procedura è stato creato un nuovo gruppo di parametri, associa il gruppo di parametri personalizzati al cluster aggiornato. Per ulteriori informazioni, consulta Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster.

> **Note**
>
> Per eseguire questo passaggio, è necessario riavviare nuovamente il cluster per applicare il nuovo gruppo di parametri.

11. (Facoltativo) Dopo avere completato i test successivi all'aggiornamento, eliminare lo snapshot manuale che Aurora ha creato all'inizio dell'aggiornamento.

AWS CLI

Per aggiornare la versione principale di un cluster Aurora MySQL DB, usa il AWS CLI modify-db-clustercomando con i seguenti parametri obbligatori:

- `--db-cluster-identifier`
- `--engine-version`
- `--allow-major-version-upgrade`
- `--apply-immediately` o `--no-apply-immediately`

Se il cluster utilizza gruppi di parametri personalizzati, includere anche una o entrambe le opzioni seguenti:

- `--db-cluster-parameter-group-name`, se il cluster utilizza un gruppo di parametri cluster personalizzato
- `--db-instance-parameter-group-name`, se alcune istanze del cluster utilizzano un gruppo di parametri database personalizzato

L'esempio seguente aggiorna il cluster `sample-cluster` DB alla versione 3.04.1 di Aurora MySQL. L'aggiornamento avviene immediatamente, invece di attendere la successiva finestra di manutenzione.

Example

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --engine-version 8.0.mysql_aurora.3.04.1 \
```

```
          --allow-major-version-upgrade \
          --apply-immediately
```

In Windows:

```
aws rds modify-db-cluster ^
          --db-cluster-identifier sample-cluster ^
          --engine-version 8.0.mysql_aurora.3.04.1 ^
          --allow-major-version-upgrade ^
          --apply-immediately
```

Puoi combinare altri comandi CLI con `modify-db-cluster` per creare un end-to-end processo automatizzato per l'esecuzione e la verifica degli aggiornamenti. Per maggiori informazioni ed esempi, vedi Aurora Il mio tutorial di aggiornamento SQL sul posto.

> **Note**
>
> Se il cluster fa parte di un database globale Aurora, la procedura di aggiornamento in loco è leggermente diversa. Si chiama l'operazione di modify-global-clustercomando invece di. `modify-db-cluster` Per ulteriori informazioni, consulta Principali aggiornamenti in loco per database globali.

API RDS

Per aggiornare la versione principale di un cluster Aurora MySQL DB, utilizza l'operazione RDS API Modify con i seguenti parametri richiesti: DBCluster

- `DBClusterIdentifier`
- `Engine`
- `EngineVersion`
- `AllowMajorVersionUpgrade`
- `ApplyImmediately` (impostato su `true` o `false`)

> **Note**
>
> Se il cluster fa parte di un database globale Aurora, la procedura di aggiornamento in loco è leggermente diversa. Si chiama l'operazione invece di.

ModifyGlobalClusterModifyDBCluster Per ulteriori informazioni, consulta Principali aggiornamenti in loco per database globali.

Come gli aggiornamenti in loco influiscono sui gruppi di parametri per un cluster

I gruppi di parametri di Aurora hanno diversi insiemi di impostazioni di configurazione per i cluster compatibili con MySQL 5.7 o 8.0. Quando si esegue un aggiornamento locale, il cluster aggiornato e tutte le relative istanze devono utilizzare i corrispondenti gruppi di parametri cluster e istanza corrispondenti.

Il cluster e le istanze potrebbero utilizzare i gruppi di parametri compatibili con 5.7 predefiniti. In tal caso, il cluster e l'istanza aggiornati iniziano con i gruppi di parametri compatibili con 8.0 predefiniti. Se il cluster e le istanze utilizzano gruppi di parametri personalizzati, accertarsi di creare i gruppi di parametri compatibili con 8.0 corrispondenti. Accertarsi inoltre di specificarli durante il processo di aggiornamento.

Note

Per la maggior parte delle impostazioni dei parametri, è possibile scegliere il gruppo di parametri personalizzati in due punti. Quando si crea il cluster o quando si associa il gruppo di parametri al cluster in un secondo momento.
Tuttavia, se si utilizza un'impostazione non predefinita per il parametro `lower_case_table_names`, è necessario impostare il gruppo di parametri personalizzati con questa impostazione in anticipo. Quindi specificare il gruppo di parametri quando si esegue il ripristino dello snapshot per creare il cluster. Qualsiasi modifica al parametro `lower_case_table_names` non ha effetto dopo la creazione del cluster.
È opportuno utilizzare la stessa impostazione per `lower_case_table_names` durante l'aggiornamento da Aurora MySQL versione 2 alla versione 3.
Con un database globale Aurora basato su Aurora MySQL, è possibile eseguire un aggiornamento sul posto da Aurora MySQL versione 2 alla versione 3 solo se si imposta il parametro su default e si riavvia il database globale. `lower_case_table_names` Per ulteriori informazioni sui metodi disponibili all'uso, consulta Aggiornamenti di una versione principale.

> **Important**
>
> Se si specifica un gruppo di parametri personalizzato durante il processo di aggiornamento, accertarsi di riavviare manualmente il cluster al termine dell'aggiornamento. In questo modo, il cluster inizia a utilizzare le impostazioni dei parametri personalizzati.

Modifiche delle proprietà del cluster tra versioni di Aurora MySQL

Quando esegui l'aggiornamento da Aurora MySQL versione 2 alla versione 3, assicurati di modificare le applicazioni o gli script utilizzati per configurare o gestire i cluster e le istanze database Aurora MySQL.

Inoltre, modifica il codice che manipola i gruppi di parametri per tenere conto del fatto che i nomi dei gruppi di parametri predefiniti sono diversi per i cluster compatibili con 5.7 e 8.0. I nomi dei gruppi di parametri predefiniti per i cluster di Aurora MySQL versione 2 e 3 sono `default.aurora-mysql5.7` e `default.aurora-mysql8.0`, rispettivamente.

Ad esempio, è possibile che si disponga di codice simile al seguente applicabile al cluster prima di un aggiornamento.

```
# Check the default parameter values for MySQL 5.7-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql5.7 --
region us-east-1
```

Dopo avere aggiornato la versione principale del cluster, è necessario modificare tale codice nel modo seguente.

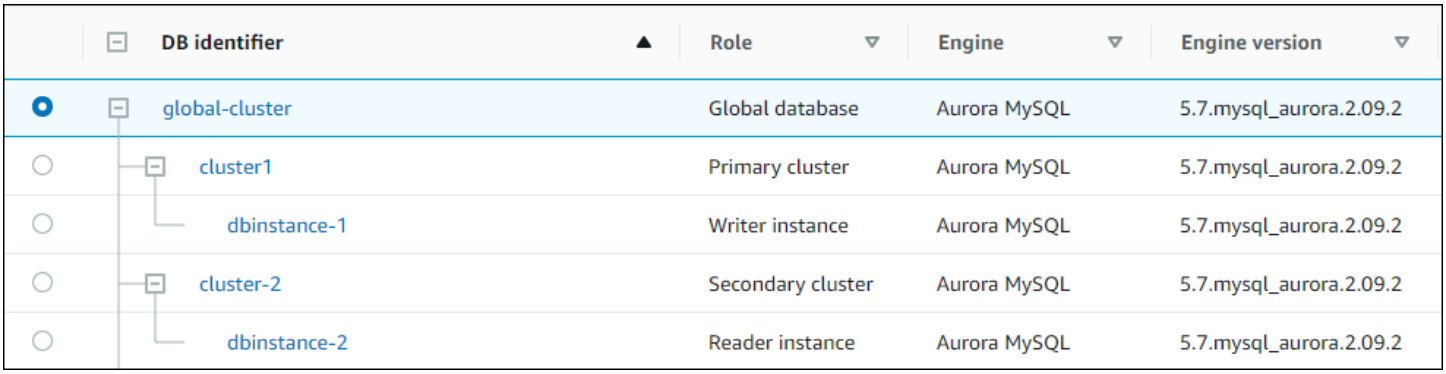
```
# Check the default parameter values for MySQL 8.0-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql8.0 --
region us-east-1
```

Principali aggiornamenti in loco per database globali

Per un database globale Aurora, è possibile aggiornare il cluster di database globale. Aurora aggiorna automaticamente tutti i cluster nello stesso momento e assicura che tutti eseguano la stessa versione del motore. Questo requisito è dovuto al fatto che tutte le modifiche apportate alle tabelle di sistema, ai formati di file di dati e così via vengono replicate automaticamente in tutti i cluster secondari.

Segui le istruzioni in Come funziona l'aggiornamento della versione principale di Aurora MySQL in loco. Quando specifichi cosa aggiornare, assicurati di scegliere il cluster di database globale anziché uno dei cluster in esso contenuti.

Se utilizzi il, scegli l'elemento con il ruolo Database AWS Management Console globale.



Se utilizzi l'API AWS CLI o RDS, avvia il processo di aggiornamento chiamando il modify-global-clustercomando o l'ModifyGlobalClusteroperazione. Utilizzare uno di questi anziché `modify-db-cluster` o `ModifyDBCluster`.

> **Note**
>
> Non è possibile specificare un gruppo di parametri personalizzato per il cluster di database globale mentre si esegue un aggiornamento della versione principale del database globale Aurora. Creare i gruppi di parametri personalizzati in ciascuna regione del cluster globale. Applicarli quindi manualmente ai cluster regionali dopo l'aggiornamento.

Per aggiornare la versione principale di un cluster di database globale Aurora MySQL utilizzando il AWS CLI, usa il modify-global-clustercomando con i seguenti parametri richiesti:

- `--global-cluster-identifier`
- `--engine aurora-mysql`
- `--engine-version`
- `--allow-major-version-upgrade`

L'esempio seguente aggiorna il cluster di database globale alla versione 3.04.2 di Aurora MySQL.

Example

In Linux, macOS, oppure Unix:

```
aws rds modify-global-cluster \
          --global-cluster-identifier global_cluster_identifier \
          --engine aurora-mysql \
          --engine-version 8.0.mysql_aurora.3.04.2 \
          --allow-major-version-upgrade
```

In Windows:

```
aws rds modify-global-cluster ^
          --global-cluster-identifier global_cluster_identifier ^
          --engine aurora-mysql ^
          --engine-version 8.0.mysql_aurora.3.04.2 ^
          --allow-major-version-upgrade
```

Aggiornamenti sul posto per cluster DB con repliche di lettura interregionali

È possibile aggiornare un cluster Aurora DB con una replica di lettura interregionale utilizzando la procedura di aggiornamento sul posto, ma ci sono alcune considerazioni:

- È necessario prima aggiornare il cluster DB di replica di lettura. Se si tenta di aggiornare prima il cluster primario, verrà visualizzato un messaggio di errore simile al seguente:

  Impossibile aggiornare il cluster DB test-xr-primary-cluster perché la replica Aurora Cross-Region associata test-xr-replica-cluster non è ancora stata patchata. Aggiorna la replica Aurora Cross-Region e riprova.

  Ciò significa che il cluster DB primario non può avere una versione del motore DB superiore a quella del cluster di replica.
- Prima di aggiornare il cluster DB primario, interrompi il carico di lavoro di scrittura e disabilita tutte le nuove richieste di connessione all'istanza Writer DB del cluster primario.
- Quando aggiorni il cluster primario, scegli un gruppo di parametri del cluster DB personalizzato con il `binlog_format` parametro impostato su un valore che supporti la replica con logging binario, ad esempio. `MIXED`

  Per ulteriori informazioni sull'utilizzo di registrazione binaria con Aurora, consulta Replica tra Aurora e SQL My o tra Aurora e un altro cluster Aurora DB (replica di log binari). Per ulteriori

informazioni sulla modifica dei parametri di configurazione di Aurora MySQL, consulta Parametri di configurazione Aurora MySQL e .

- Non aspettate troppo tempo per aggiornare il cluster DB primario dopo aver aggiornato il cluster di replica. Si consiglia di non attendere più a lungo della finestra di manutenzione successiva.
- Dopo aver aggiornato il cluster DB primario, riavvia la relativa istanza Writer DB. Il gruppo di parametri del cluster DB personalizzato che abilita la replica binlog non ha effetto finché l'istanza DB writer non viene riavviata.
- Non riprendete il carico di lavoro di scrittura né abilitate le connessioni all'istanza DB di scrittura finché non confermate che la replica tra regioni è stata riavviata e che il ritardo di replica nell'istanza secondaria è 0. Regione AWS

Aurora Il mio tutorial di aggiornamento SQL sul posto

I seguenti esempi di Linux mostrano come è possibile eseguire i passaggi generali della procedura di aggiornamento in loco utilizzando la AWS CLI.

Questo primo esempio crea un cluster Aurora DB che esegue una versione 2.x di Aurora My. SQL Il cluster include un'istanza database di scrittura e un'istanza database di lettura. Il comando `wait db-instance-available` si interrompe fino a quando l'istanza database di scrittura non è disponibile. Questo è il momento in cui il cluster è pronto per essere aggiornato.

```
aws rds create-db-cluster --db-cluster-identifier mynewdbcluster --engine aurora-mysql
 \
  --db-cluster-version 5.7.mysql_aurora.2.11.2
...
aws rds create-db-instance --db-instance-identifier mynewdbcluster-instance1 \
  --db-cluster-identifier mynewdbcluster --db-instance-class db.t4g.medium --engine
 aurora-mysql
...
aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

Le versioni di Aurora My SQL 3.x a cui è possibile aggiornare il cluster dipendono dalla versione 2.x attualmente in esecuzione sul cluster e dalla posizione in Regione AWS cui si trova il cluster. Il primo comando, con `--output text`, mostra soltanto la versione di destinazione disponibile. Il secondo comando mostra l'JSONoutput completo della risposta. Questa risposta contiene dettagli quali il valore `aurora-mysql` utilizzato per il parametro `engine`. È inoltre possibile notare che l'aggiornamento alla versione 3.04.0 rappresenta un aggiornamento della versione principale.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.11.2 \
  --query '*[].[ValidUpgradeTarget]'
...
{
    "Engine": "aurora-mysql",
    "EngineVersion": "8.0.mysql_aurora.3.04.0",
    "Description": "Aurora MySQL 3.04.0 (compatible with MySQL 8.0.28)",
    "AutoUpgrade": false,
    "IsMajorVersionUpgrade": true,
    "SupportedEngineModes": [
        "provisioned"
    ],
    "SupportsParallelQuery": true,
    "SupportsGlobalDatabases": true,
    "SupportsBabelfish": false,
    "SupportsIntegrations": false
},
...
```

Questo esempio mostra che se si immette un numero di versione di destinazione che non è una destinazione di aggiornamento valida per il cluster, Aurora non esegue l'aggiornamento. Aurora non esegue un aggiornamento della versione principale a meno che non si includa il parametro `--allow-major-version-upgrade`. In questo modo, non è possibile eseguire accidentalmente un aggiornamento che potrebbe richiedere modifiche e test estesi al codice dell'applicazione.

```
aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 5.7.mysql_aurora.2.09.2 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: Cannot find upgrade target from 5.7.mysql_aurora.2.11.2 with requested
 version 5.7.mysql_aurora.2.09.2.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --region us-east-1 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: The AllowMajorVersionUpgrade flag must be present when upgrading to a new
 major version.
```

```
aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --apply-immediately --allow-major-version-
upgrade
{
  "DBClusterIdentifier": "mynewdbcluster",
  "Status": "available",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.11.2"
}
```

Sono necessari alcuni istanti prima che lo stato del cluster e le istanze database associate vengano modificati in `upgrading`. I numeri di versione per il cluster e le istanze database cambiano solo al termine dell'aggiornamento. Ancora una volta, è possibile utilizzare il comando `wait db-instance-available` perché l'istanza database di scrittura attenda il completamento dell'aggiornamento prima di procedere.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[Status,EngineVersion]' --output text
upgrading 5.7.mysql_aurora.2.11.2

aws rds describe-db-instances --db-instance-identifier mynewdbcluster-instance1 \
  --query '*[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceStatus:DBInstanceStatus} | [0]'
{
    "DBInstanceIdentifier": "mynewdbcluster-instance1",
    "DBInstanceStatus": "upgrading"
}

aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

A questo punto, il numero di versione del cluster corrisponde a quello specificato per l'aggiornamento.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[EngineVersion]' --output text

8.0.mysql_aurora.3.04.0
```

L'esempio precedente ha eseguito un aggiornamento immediato specificando il parametro `--apply-immediately`. Per consentire l'aggiornamento in un momento opportuno, quando si prevede che il cluster non sia occupato, è possibile specificare il parametro `--no-apply-immediately`. In questo modo l'aggiornamento viene avviato durante la successiva finestra di manutenzione del

cluster. La finestra di manutenzione definisce il periodo durante il quale possono iniziare le operazioni di manutenzione. Un'operazione di lunga durata potrebbe non terminare durante la finestra di manutenzione. Pertanto, non è necessario definire una finestra di manutenzione più ampia anche se si prevede che l'aggiornamento potrebbe richiedere molto tempo.

L'esempio seguente aggiorna un cluster che inizialmente esegue Aurora SQL My versione 2.11.2. Nell'output di `describe-db-engine-versions`, i valori `False` e `True` rappresentano la proprietà `IsMajorVersionUpgrade`. Dalla versione 2.11.2, è possibile eseguire l'aggiornamento ad altre versioni 2.*. Questi aggiornamenti non sono considerati aggiornamenti di versione principali e quindi non richiedono un aggiornamento in loco. L'aggiornamento locale è disponibile solo per gli aggiornamenti alle versioni 3.* visualizzate nell'elenco.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.10.2 \
  --query '*[].[ValidUpgradeTarget][0][0]|[*].[EngineVersion,IsMajorVersionUpgrade]'
 --output text

5.7.mysql_aurora.2.11.3 False
5.7.mysql_aurora.2.11.4 False
5.7.mysql_aurora.2.11.5 False
5.7.mysql_aurora.2.11.6 False
5.7.mysql_aurora.2.12.0 False
5.7.mysql_aurora.2.12.1 False
5.7.mysql_aurora.2.12.2 False
5.7.mysql_aurora.2.12.3 False
8.0.mysql_aurora.3.04.0 True
8.0.mysql_aurora.3.04.1 True
8.0.mysql_aurora.3.04.2 True
8.0.mysql_aurora.3.04.3 True
8.0.mysql_aurora.3.05.2 True
8.0.mysql_aurora.3.06.0 True
8.0.mysql_aurora.3.06.1 True
8.0.mysql_aurora.3.07.1 True

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --no-apply-immediately --allow-major-
version-upgrade
...
```

Quando un cluster viene creato senza una finestra di manutenzione specificata, Aurora seleziona un giorno casuale della settimana. In questo caso, il comando `modify-db-cluster` viene inviato un lunedì. Pertanto, modifichiamo la finestra di manutenzione affinché sia di martedì mattina. Tutti gli orari sono rappresentati nel fuso orario. UTC La finestra `tue:10:00-tue:10:30` indica dalle 2:00 alle 2:30 ora del Pacifico. La modifica della finestra di manutenzione ha effetto immediato.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "sat:08:20-sat:08:50"
    ]
]

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster --preferred-
maintenance-window tue:10:00-tue:10:30"
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "tue:10:00-tue:10:30"
    ]
]
```

Nell'esempio seguente viene illustrato come ottenere un report degli eventi generati dall'aggiornamento. L'argomento `--duration` rappresenta il numero di minuti per recuperare le informazioni sull'evento. Questo argomento è necessario perché per impostazione predefinita restituisce `describe-events` solo gli eventi dell'ultima ora.

```
aws rds describe-events --source-type db-cluster --source-identifier mynewdbcluster --
duration 20160
{
    "Events": [
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "DB cluster created",
            "EventCategories": [
                "creation"
            ],
            "Date": "2022-11-17T01:24:11.093000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
```

```
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing online pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:08.450000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing offline pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:59.519000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-
mynewdbcluster-5-7-mysql-aurora-2-10-2-to-8-0-mysql-aurora-3-02-0-2022-11-18-22-55].",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:00:22.318000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Cloning volume.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:01:45.428000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
```

```
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
 Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:02:25.141000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
 Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:23.036000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Upgrading database objects.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:48.208000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster major version has been upgraded",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:10:28.999000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        }
    ]
}
```

Individuazione dei motivi degli errori di aggiornamento di Aurora My SQL major version

Nel tutorial, l'aggiornamento da Aurora My SQL versione 2 alla versione 3 è riuscito. Ma se l'aggiornamento non fosse riuscito, vorresti sapere perché.

È possibile iniziare utilizzando il `describe-events` AWS CLI comando per esaminare gli eventi del cluster DB. Questo esempio mostra gli eventi delle ultime 10 ore. `mydbcluster`

```
aws rds describe-events \
    --source-type db-cluster \
    --source-identifier mydbcluster \
    --duration 600
```

In questo caso, si è verificato un errore di precontrollo dell'aggiornamento.

```
{
    "Events": [
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:22.846000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        },
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster is in a state that cannot be upgraded: Upgrade
 prechecks failed. For more details, see the
             upgrade-prechecks.log file. For more information on troubleshooting the
 cause of the upgrade failure, see
             https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
AuroraMySQL.Upgrading.Troubleshooting.html",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:24.373000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        }
    ]
```

```
}
```

Per diagnosticare la causa esatta del problema, esamina i log del database per l'istanza Writer DB. Quando un aggiornamento ad Aurora My SQL versione 3 fallisce, l'istanza writer contiene un file di registro con il nome. `upgrade-prechecks.log` Questo esempio mostra come rilevare la presenza di quel log e quindi scaricarlo in un file locale per l'analisi.

```
aws rds describe-db-log-files --db-instance-identifier mydbcluster-instance \
    --query '*[].[LogFileName]' --output text

error/mysql-error-running.log
error/mysql-error-running.log.2024-04-11.20
error/mysql-error-running.log.2024-04-11.21
error/mysql-error.log
external/mysql-external.log
upgrade-prechecks.log

aws rds download-db-log-file-portion --db-instance-identifier mydbcluster-instance \
    --log-file-name upgrade-prechecks.log \
    --starting-token 0 \
    --output text >upgrade_prechecks.log
```

Il `upgrade-prechecks.log` file è in JSON formato. Lo scarichiamo utilizzando l'`--output text`opzione per evitare di codificare l'JSONoutput all'interno di un altro JSON wrapper. Per gli aggiornamenti di Aurora My SQL versione 3, questo registro include sempre determinati messaggi informativi e di avviso. Include messaggi di errore solo se l'aggiornamento non riesce. Se l'aggiornamento riesce, il file di log non viene prodotto.

Per riepilogare tutti gli errori e visualizzare i campi relativi all'oggetto e alla descrizione associati, è possibile eseguire il comando `grep -A 2 '"level": "Error"'` sul contenuto del file. `upgrade-prechecks.log` In questo modo viene visualizzata ogni riga di errore e le due righe successive. Queste contengono il nome dell'oggetto di database corrispondente e le indicazioni su come correggere il problema.

```
$ cat upgrade-prechecks.log | grep -A 2 '"level": "Error"'

"level": "Error",
"dbObject": "problematic_upgrade.dangling_fulltext_index",
"description": "Table `problematic_upgrade.dangling_fulltext_index` contains dangling
 FULLTEXT index. Kindly recreate the table before upgrade."
```

In questo esempio, è possibile eseguire il SQL comando seguente sulla tabella che causa il problema per cercare di risolvere il problema oppure è possibile ricreare la tabella senza l'indice pendente.

```
OPTIMIZE TABLE problematic_upgrade.dangling_fulltext_index;
```

Quindi riprova a eseguire l'aggiornamento.

Risoluzione dei problemi relativi all'aggiornamento in-place di Aurora My SQL

Utilizza i seguenti suggerimenti per risolvere i problemi relativi agli upgrade in-place di Aurora My. SQL Questi suggerimenti non si applicano a Aurora Serverless Cluster DB.

Motivo dell'annullamento o della lentezza dell'aggi ornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
La replica interregionale Aurora associata non è ancora stata patchata	Aurora annulla l'aggiorn amento.	Aggiorna la replica Aurora Cross-Region e riprova.
Il cluster ha transazioni XA nello stato preparato	Aurora annulla l'aggiorn amento.	Salvare o eseguire il rollback di tutte le transazioni XA preparate.
Il cluster sta elaborando tutte le istruzioni del linguaggio di definizione dei dati () DDL	Aurora annulla l'aggiorn amento.	Valuta la possibilità di attendere ed eseguire l'aggiornamento dopo che tutte le DDL istruzion i sono state completate.
Il cluster ha modifiche non salvate per molte righe	L'aggiorn amento potrebbe richiedere molto tempo.	Il processo di aggiornamento esegue il rollback delle modifiche non salvate. L'indicatore per questa condizione è il valore di `TRX_ROWS_ MODIFIED` nella tabella `INFORMATI ON_SCHEMA.INNODB_TRX` . Prendere in considerazione l'esecuzione dell'aggiornamento solo dopo avere salvato

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
		le modifiche o eseguito il rollback di tutte le transazioni di grandi dimensioni.
Il cluster ha un numero elevato di record di annullamento	L'aggiornamento potrebbe richiedere molto tempo.	Anche se le transazioni non salvate non interessano un numero elevato di righe, potrebbero comportare un grande volume di dati. Ad esempio, è possibile che stiate inserendo caratteri grandiBLOBs. Aurora non rileva o genera automaticamente un evento per questo tipo di attività di transazione. L'indicatore di questa condizione è la lunghezza dell'elenco cronologico (HLL). Il processo di aggiornamento esegue il rollback delle modifiche non salvate. È possibile verificarlo HLL nell'output del `SHOW ENGINE INNODB STATUS` SQL comando o direttamente utilizzando la seguente SQL query: `SELECT count FROM information_schema .innodb_metrics WHERE name = 'trx_rseg_history_len';` Puoi anche monitorare la `RollbackSegmentHistoryListLength` metrica in Amazon CloudWatch. Valuta la possibilità di eseguire l'aggiornamento solo dopo che HLL è stato ridotto.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
Il cluster è in fase di salvataggio di una transazione di log binario di grandi dimensioni	L'aggiornamento potrebbe richiedere molto tempo.	Il processo di aggiornamento attende fino a quando non vengono applicate le modifiche del log binario. Durante questo periodo potrebbero iniziare altre transazioni o DDL rendicontati, rallentando ulteriormente il processo di aggiornamento. Pianificare il processo di aggiornamento quando il cluster non è occupato con la generazione di modifiche alla replica del log binario. Aurora non rileva o genera automaticamente un evento per questa condizione.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione
Incongruenze dello schema derivanti dalla rimozione o dal danneggiamento dei file	Aurora annulla l'aggiornamento.	Cambia il motore di archiviazione predefinito per le tabelle temporanee da My ISAM a InnoDB. Esegui queste fasi: 1. Modificare il parametro database `default_tmp_storage_engine` impostandolo su `InnoDB`. 2. Riavviare il cluster database. 3. Dopo il riavvio, verificare che il parametro database `default_tmp_storage_engine` sia impostato su `InnoDB`. Utilizza il seguente comando: `show global variables like 'default_tmp_storage_engine';` 4. Assicurati di non creare tabelle temporanee che utilizzano il motore di ISAM archiviazione My. È consigliabile sospendere qualsiasi carico di lavoro del database e non creare nuove connessioni al database perché l'aggiornamento verrà rilasciato a breve. 5. Provare di nuovo l'aggiornamento sul posto.

Motivo dell'annullamento o della lentezza dell'aggiornamento in loco	Effetto	Soluzione per consentire il completamento dell'aggiornamento in loco all'interno della finestra di manutenzione

L'utente principale è stato eliminato	Aurora annulla l'aggiornamento.	**Important**
		Non eliminare l'utente principale.
		
		Tuttavia, se per qualche motivo dovessi eliminare l'utente master, ripristinalo utilizzando i seguenti SQL comandi:
		
		`CREATE USER 'master_username '@'% IDENTIFIED BY ' master_user_password ' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;`
		
		`GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, LOAD FROM S3, SELECT INTO S3, INVOKE LAMBDA, INVOKE SAGEMAKER , INVOKE COMPREHEND ON *.* TO 'master_username '@'% WITH GRANT OPTION;`

Per ulteriori dettagli sulla risoluzione dei problemi che causano il fallimento dei controlli preliminari di aggiornamento, consultate i seguenti blog:

- Elenco di controllo per l'aggiornamento da Amazon Aurora My SQL versione 2 (con compatibilità My SQL 5.7) alla versione 3 (con compatibilità My SQL 8.0), parte 1

- Elenco di controllo per l'aggiornamento da Amazon Aurora My SQL versione 2 (con compatibilità My SQL 5.7) alla versione 3 (con compatibilità My SQL 8.0), parte 2

È possibile utilizzare la procedura seguente per eseguire controlli personalizzati per alcune delle condizioni della tabella precedente. In questo modo, è possibile pianificare l'aggiornamento in un momento in cui si sa che il database si trova in uno stato in cui l'aggiornamento può essere completato correttamente e con rapidità.

- È possibile verificare la presenza di transazioni XA aperte eseguendo l'istruzione `XA RECOVER`. È quindi possibile salvare le modifiche o eseguire il rollback delle transazioni XA prima di iniziare l'aggiornamento.
- Puoi verificare la presenza di DDL istruzioni `SHOW PROCESSLIST` eseguendole e cercando`CREATE`,, `DROP` `ALTER`, `RENAME`, e `TRUNCATE` istruzioni nell'output. Consentite il completamento di tutte le DDL istruzioni prima di iniziare l'aggiornamento.
- È possibile controllare il numero totale di righe non salvate eseguendo una query sulla tabella `INFORMATION_SCHEMA.INNODB_TRX`. La tabella contiene una riga per ogni transazione. La colonna `TRX_ROWS_MODIFIED` contiene il numero di righe modificate o inserite dalla transazione.
- È possibile controllare la lunghezza della lista della cronologia InnoDB eseguendo l'istruzione `SHOW ENGINE INNODB STATUS SQL` e cercando la `History list length` nell'output. È inoltre possibile controllare direttamente il valore eseguendo la seguente query:

  ```
  SELECT count FROM information_schema.innodb_metrics WHERE name =
   'trx_rseg_history_len';
  ```

  La lunghezza dell'elenco cronologico corrisponde alla quantità di informazioni di annullamento memorizzate dal database per implementare il controllo simultaneo di più versioni (). MVCC

Pulizia post-aggiornamento per Aurora My versione 3 SQL

Dopo aver completato l'aggiornamento di qualsiasi cluster Aurora My SQL versione 2 ad Aurora My SQL versione 3, puoi eseguire queste altre azioni di pulizia:

- Crea nuove versioni compatibili con My SQL 8.0 di qualsiasi gruppo di parametri personalizzati. Applicare i valori dei parametri personalizzati necessari ai nuovi gruppi di parametri.

- Aggiorna eventuali CloudWatch allarmi, script di configurazione e così via per utilizzare i nuovi nomi per tutte le metriche i cui nomi sono stati influenzati da modifiche linguistiche complete. Per un elenco di parametri, consultare Modifiche linguistiche incluse per Aurora My versione 3 SQL.
- Aggiorna qualsiasi AWS CloudFormation modelli per utilizzare i nuovi nomi per tutti i parametri di configurazione i cui nomi sono stati influenzati da modifiche linguistiche complete. Per l'elenco completo dei parametri, consultare Modifiche linguistiche incluse per Aurora My versione 3 SQL.

Indici spaziali

Dopo l'aggiornamento ad Aurora My SQL versione 3, verifica se è necessario eliminare o ricreare oggetti e indici relativi agli indici spaziali. Prima di My SQL 8.0, Aurora era in grado di ottimizzare le query spaziali utilizzando indici che non contenevano un identificatore di risorse spaziali (). SRID Aurora My SQL versione 3 utilizza solo indici spaziali contenenti. SRIDs Durante un aggiornamento, Aurora elimina automaticamente tutti gli indici spaziali senza SRIDs e stampa i messaggi di avviso nel registro del database. Se osservate questi messaggi di avviso, create nuovi indici spaziali dopo l'aggiornamento. SRIDs Per ulteriori informazioni sulle modifiche alle funzioni spaziali e ai tipi di dati in My SQL 8.0, vedere Changes in My 8.0 nel My Reference SQL Manual. SQL

Aggiornamenti e correzioni del motore di database per Amazon Aurora MySQL

Puoi trovare le seguenti informazioni nelle note di rilascio per Amazon Aurora MySQL Compatible Edition:

- Aggiornamenti del motore di database per Amazon Aurora MySQL versione 3
- Aggiornamenti del motore di database per Amazon Aurora MySQL versione 2
- Aggiornamenti del motore di database per Amazon Aurora MySQL versione 1 (obsoleta)
- Bug MySQL corretti dagli aggiornamenti del motore di database Aurora MySQL
- Vulnerabilità di sicurezza corrette in Amazon Aurora MySQL

Utilizzo di Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL è un motore di database relazionale compatibile con ACID e PostgreSQL completamente gestito che combina la velocità, l'affidabilità e la gestibilità di Amazon Aurora con la semplicità e la convenienza dei database open source. Aurora PostgreSQL è un'alternativa a PostgreSQL che semplifica e rende più conveniente dal punto di vista dei costi la configurazione, l'utilizzo e il dimensionamento delle implementazioni PostgreSQL nuove ed esistenti, consentendoti di concentrarti sulle tue attività e applicazioni. Per ulteriori informazioni su Aurora in generale, consulta Che cos'è Amazon Aurora?.

Oltre ai vantaggi di Aurora, Aurora PostgreSQL offre un comodo percorso di migrazione da Amazon RDS ad Aurora, con strumenti di migrazione rapidi che convertono le applicazioni RDS per PostgreSQL esistenti in Aurora PostgreSQL. Anche le attività di database di routine, come il provisioning, l'applicazione di patch, il backup, il ripristino, il rilevamento di guasti e la correzione, sono facili da gestire con Aurora PostgreSQL.

Aurora PostgreSQL può funzionare con molti standard di settore. Ad esempio, è possibile utilizzare i database Aurora PostgreSQL per creare applicazioni conformi allo standard HIPAA e memorizzare le relative informazioni sanitarie, tra cui i dati sanitari protetti (PHI) da un Business Associate Agreement (BAA) stipulato con AWS.

Aurora PostgreSQL è idoneo per FedRAMP HIGH. Per i dettagli AWS e gli sforzi di conformità, consulta la sezione AWS Servizi rientranti nell'ambito del programma di conformità.

Important

Se riscontri un problema con il cluster DB Aurora PostgreSQL, il tuo agente di AWS supporto potrebbe aver bisogno di ulteriori informazioni sullo stato dei tuoi database. L'obiettivo è garantire che AWS Support riceva quante più informazioni richieste possibile nel più breve tempo possibile.

È possibile utilizzare PG Collector per raccogliere informazioni preziose sul database in un file HTML consolidato. Per ulteriori informazioni su PG Collector, su come eseguirlo e su come scaricare il report HTML, consultate PG Collector.

Una volta completato con successo, e salvo diversa indicazione, lo script restituisce l'output in un formato HTML leggibile. Lo script è progettato per escludere dal codice HTML tutti i dati o i dettagli di sicurezza che potrebbero compromettere l'azienda. Inoltre, non apporta

modifiche al database o all'ambiente. Tuttavia, se trovi informazioni nell'HTML che non ti senti a tuo agio a condividere, sentiti libero di rimuovere le informazioni problematiche prima di caricare il codice HTML. Quando il codice HTML è accettabile, caricalo utilizzando la sezione degli allegati nei dettagli del caso della tua richiesta di assistenza.

Argomenti

- Lavorare con l'ambiente di anteprima del database
- Sicurezza con Amazon Aurora PostgreSQL
- Aggiornamento delle applicazioni per la connessione ai cluster DB Aurora PostgreSQL utilizzando nuovi certificati SSL/TLS
- Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL
- Migrazione dei dati su Amazon Aurora con compatibilità Postgre SQL
- Ottimizzazione delle prestazioni delle query in Aurora PostgreSQL
- Utilizzo di tabelle non registrate in Aurora PostgreSQL
- Utilizzo dell'autovacuum di PostgreSQL su Aurora PostgreSQL
- Utilizzo di Babelfish per Aurora PostgreSQL
- Prestazioni e scalabilità per Amazon Aurora PostgreSQL
- Sintonizzazione degli eventi di attesa per Aurora PostgreSQL
- Ottimizzazione con informazioni proattive su Amazon SQL Guru DevOps
- Best practice con Amazon Aurora PostgreSQL
- Replica con Amazon Aurora PostgreSQL
- Inoltro di scrittura locale in Aurora PostgreSQL
- Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock
- Integrazione di Amazon Aurora SQL Postgre con altri servizi AWS
- Monitoraggio dei piani di esecuzione delle query e della memoria di picco per Aurora Postgre SQL
- Gestione dei piani di esecuzione delle query per Aurora PostgreSQL
- Utilizzo di estensioni e wrapper di dati esterni
- Utilizzo di Trusted Language Extensions per PostgreSQL
- Riferimento Amazon Aurora PostgreSQL

- Aggiornamenti del motore di database per Amazon Aurora Postgre SQL

Lavorare con l'ambiente di anteprima del database

La SQL comunità Postgre rilascia ogni anno una nuova versione principale di SQL Postgre. Allo stesso modo, Amazon Aurora rende disponibili le versioni SQL principali di Postgre come versioni di anteprima. Ciò consente di creare un cluster DB utilizzando la versione di anteprima e di testarne le funzionalità nell'ambiente di anteprima del database.

I cluster Aurora Postgre SQL DB nel Database Preview Environment sono funzionalmente simili agli altri cluster Aurora Postgre DB. SQL Tuttavia, una versione di anteprima non può essere utilizzata in produzione.

Tieni presenti le importanti limitazioni riportate di seguito:

- Tutte le istanze DB e i cluster DB vengono eliminati 60 giorni dopo la loro creazione, insieme a tutti i backup e le istantanee.
- Puoi creare un'istanza DB solo in un cloud privato virtuale (VPC) basato sul VPC servizio Amazon.
- Non puoi copiare uno snapshot di un'istanza database in un ambiente di produzione.

Le seguenti opzioni sono supportate dall'anteprima.

- Puoi creare istanze DB utilizzando solo i tipi di istanze r5, r6g, r6i, r7g, r7i, r8g, x2g, t3 e t4g. Per ulteriori informazioni sulle classi di istanza, consulta Classi di istanze DB Amazon Aurora.
- È possibile utilizzare distribuzioni AZ singola e Multi-AZ.
- È possibile utilizzare le funzioni standard di dump e load di Postgre per esportare database da o importare database nel Database Preview EnvironmentSQL.

Tipi di classi di istanze DB supportati

Amazon Aurora Postgre SQL supporta le seguenti classi di istanze DB nella regione di anteprima:

Classi ottimizzate per la memoria

- db.r5: classi di istanze ottimizzate per la memoria
- db.r6g: classi di istanze ottimizzate per la memoria alimentate da processori Graviton2 AWS

- db.r6i: classi di istanze ottimizzate per la memoria
- db.r7g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton3 AWS
- db.r7i — classi di istanze ottimizzate per la memoria
- db.r8g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton4 AWS
- db.x2g — classi di istanze ottimizzate per la memoria alimentate da processori Graviton2 AWS

> **Note**
>
> Per ulteriori informazioni sull'elenco delle classi di istanze, vedere. Tipi di classi di istanza database

Classi burstable

- db.t3.medium
- db.t3.large
- db.t4g.medium
- db.t4g.large

Funzionalità non supportate nell'ambiente di anteprima

Le seguenti funzionalità non sono disponibili nell'ambiente di anteprima:

- Aurora Serverless v1 e v2
- Principali aggiornamenti delle versioni () MVU
- Non verranno rilasciate nuove versioni secondarie nell'area di anteprima
- RDSper la replica in entrata da Postgre ad SQL Aurora Postgre SQL
- Implementazione di Amazon RDS Blue/Green
- Copia di snapshot tra regioni diverse
- Database globale Aurora
- Flussi di attività del database (DAS), proxy e RDS AWS DMS
- Ridimensionamento automatico delle repliche di lettura
- AWS Base rocciosa

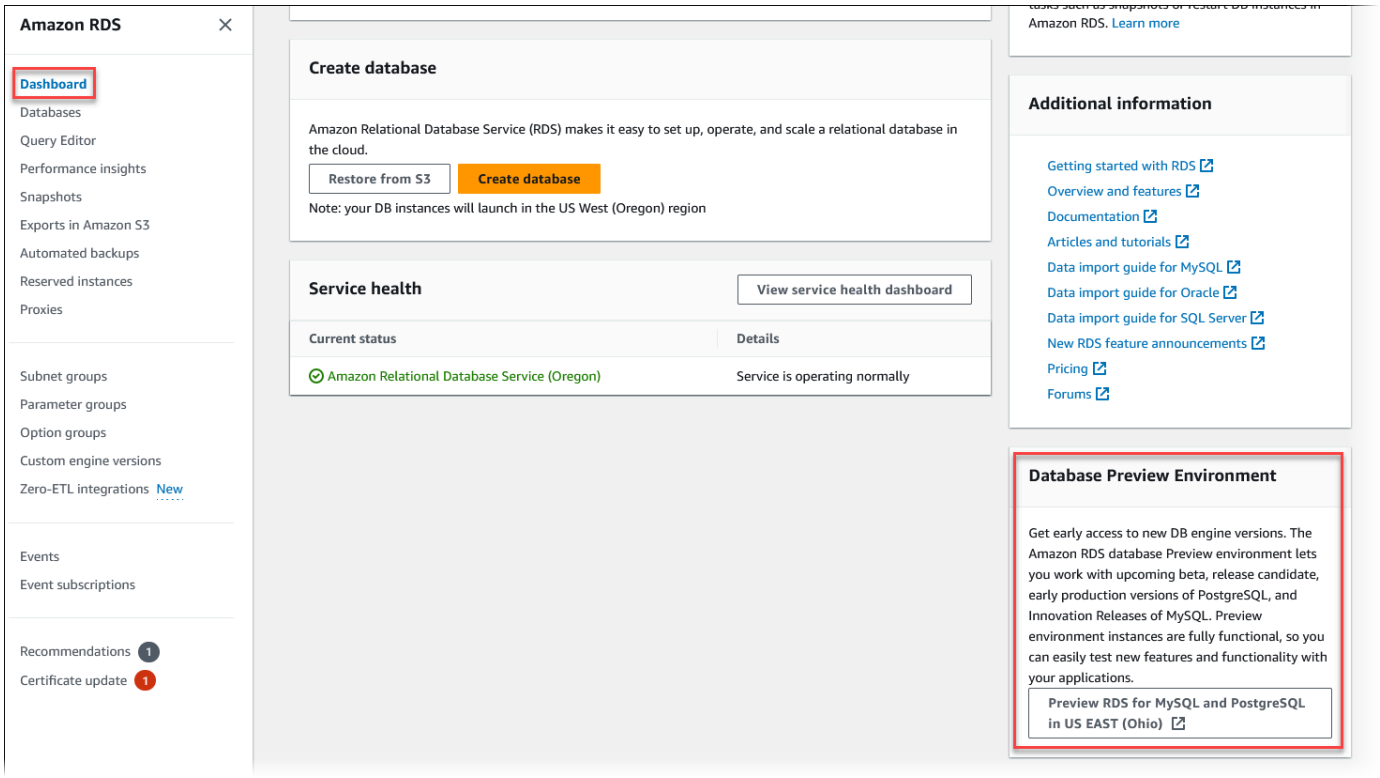
- esportazione di RDS
- Approfondimenti sulle prestazioni
- Endpoint personalizzati
- Copia istantanea
- Zero- ETL
- Babelfish
- Tutte le estensioni

Creazione di un nuovo cluster DB nell'ambiente di anteprima

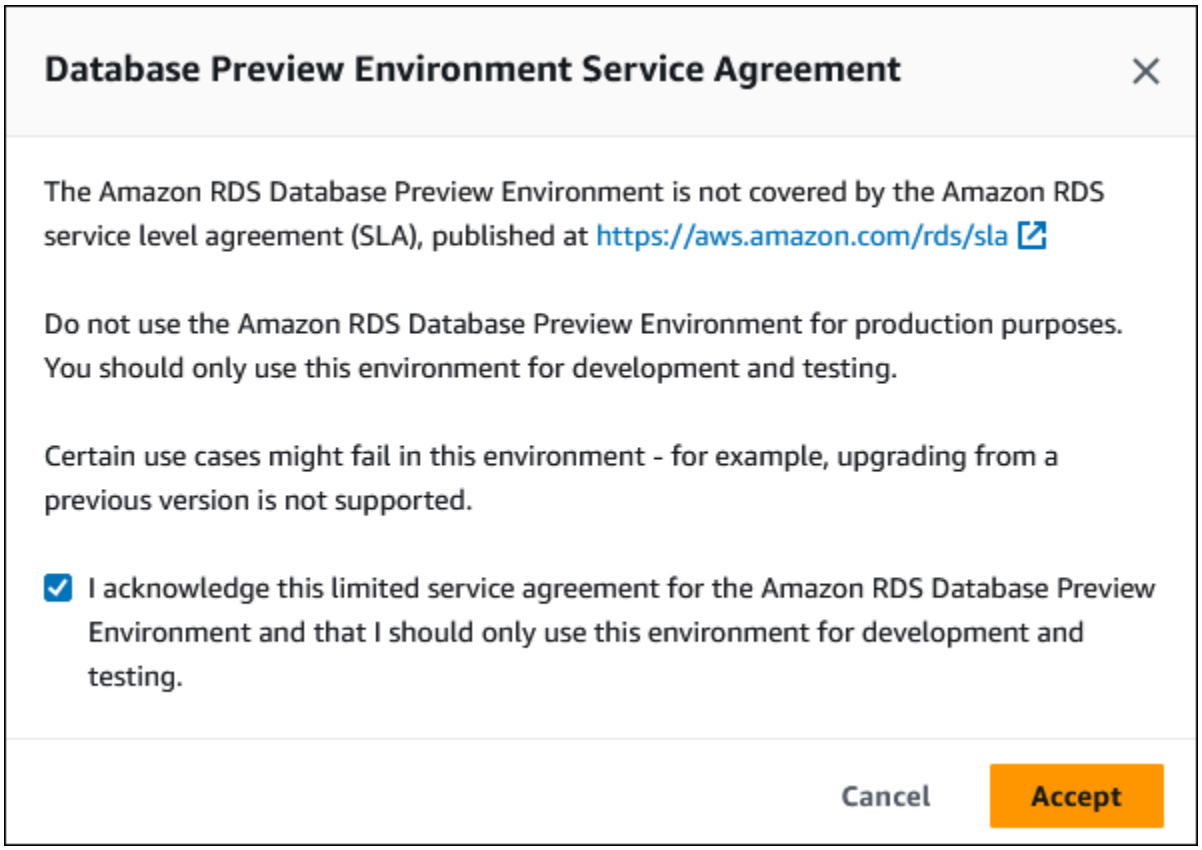
Utilizzare la procedura seguente per creare un cluster DB nell'ambiente di anteprima.

Per creare un cluster DB nell'ambiente di anteprima

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Dashboard (Pannello di controllo) nel pannello di navigazione.
3. Nella pagina Dashboard (Pannello di controllo), individua la sezione Database Preview Environment (Ambiente di anteprima del database), come mostrato nell'immagine seguente.



L'ambiente di anteprima del database è accessibile direttamente. Prima di poter procedere, è necessario capire e accettare le limitazioni.



4. Per creare il cluster Aurora Postgre SQL DB, segui lo stesso processo di creazione di qualsiasi cluster Aurora DB. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

Per creare un'istanza nell'ambiente di anteprima del database utilizzando Aurora API o il AWS CLI, usa il seguente endpoint.

```
rds-preview.us-east-2.amazonaws.com
```

Postgre SQL versione 17 nell'ambiente Database Preview

⚠ Questa è la documentazione di anteprima per Aurora SQL Postgre versione 17. ed è soggetta a modifiche.

La SQL versione 17.0 di Postgre è ora disponibile nell'ambiente Amazon RDS Database Preview. La SQL versione 17 di Postgre contiene diversi miglioramenti descritti nella seguente documentazione di Postgre: SQL

- È SQL stato rilasciato Postgre 17

Per informazioni sull'ambiente di anteprima del database, consulta Lavorare con l'ambiente di anteprima del database. Per accedere all'ambiente di anteprima dalla console, selezionate. https://console.aws.amazon.com/rds-preview/

Sicurezza con Amazon Aurora PostgreSQL

Per una panoramica generale della sicurezza in Aurora, consulta Sicurezza in Amazon Aurora. Puoi gestire la sicurezza per Amazon Aurora PostgreSQL a diversi livelli:

- Per controllare chi può eseguire azioni di gestione di Amazon RDS su cluster e istanze DB PostgreSQL Aurora, usa (IAM). AWS Identity and Access Management IAM gestisce l'autenticazione dell'identità utente prima che questi possa accedere al servizio. Gestisce inoltre l'autorizzazione, ovvero se l'utente è autorizzato a fare ciò che sta cercando di fare. L'autenticazione del database IAM è un metodo di autenticazione aggiuntivo che è possibile scegliere quando si crea un cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.

  Se utilizzi IAM con il tuo cluster Aurora PostgreSQL DB, accedi prima con le tue credenziali IAM, prima di aprire la AWS Management Console console Amazon RDS all'indirizzo. https://console.aws.amazon.com/rds/
- Assicurati che i cluster di database Aurora vengano creati in un cloud privato virtuale (VPC) utilizzando il servizio Amazon VPC. Per controllare quali dispositivi e EC2 istanze Amazon possono aprire connessioni all'endpoint e alla porta dell'istanza DB per i cluster Aurora DB in un VPC, utilizza un gruppo di sicurezza VPC. Le connessioni a endpoint e porte possono essere stabilite tramite Secure Sockets Layer (SSL). Le regole del firewall aziendale possono inoltre determinare se i dispositivi in esecuzione nell'azienda possono aprire connessioni a un'istanza database. Per ulteriori informazioni su, consulta. VPCs Amazon VPC e Aurora

  La tenancy VPC supportata dipende dalla classe di istanze database usata dai cluster DB di Aurora PostgresSQL. Con la tenancy VPC `default`, il cluster di database viene eseguito nell'hardware condiviso. Con la tenancy VPC `dedicated`, il cluster di database viene eseguito in un'istanza

hardware dedicata. Le classi di istanza database delle prestazioni in modalità burst supportano solo la tenancy VPC di default. Le classi di istanza database delle prestazioni in modalità burst includono le classi di istanza database db.t3 e db.t4g. Tutte le altre classi di istanza database di Aurora PostgreSQL DB supportano sia il tenancy VPC di default che dedicato.

Per ulteriori informazioni sulle classi di istanza, consulta Classi di istanze DB Amazon Aurora. Per ulteriori informazioni sulle tenancy VPC `default` e `dedicated`, consulta Istanze dedicate nella Guida per l'utente di Amazon Elastic Compute Cloud.

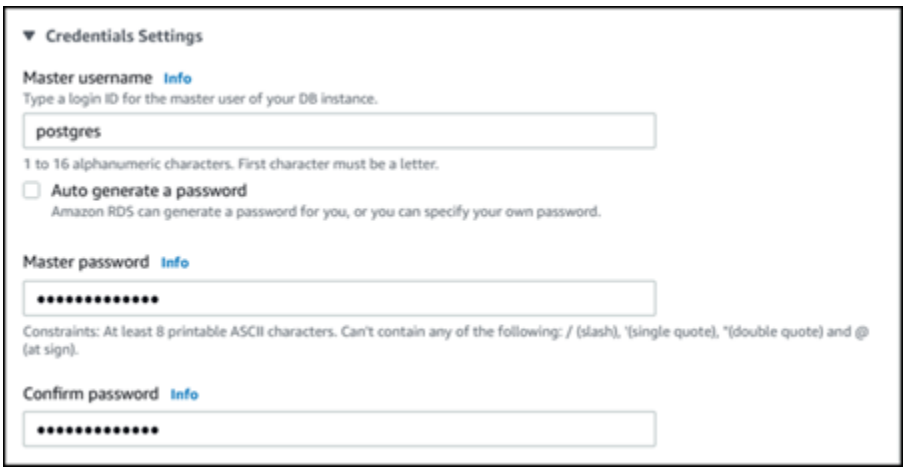
- Per concedere autorizzazioni ai database PostgreSQL eseguiti sul cluster di database Amazon Aurora, puoi adottare lo stesso approccio generale utilizzato con le istanze standalone di PostgreSQL. I comandi come `CREATE ROLE`, `ALTER ROLE`, `GRANT` e `REVOKE` funzionano esattamente come nei database on-premise, ovvero in modo analogo alla modifica diretta delle tabelle dello schema del database.

  PostgreSQL gestisce i privilegi mediante i ruoli. Il ruolo `rds_superuser` è quello più privilegiato a livello di cluster di database Aurora PostgreSQL. Questo ruolo viene creato automaticamente e viene concesso all'utente che crea il cluster di database (l'account utente principale, `postgres` per impostazione predefinita). Per ulteriori informazioni, consulta Informazioni su ruoli e autorizzazioni di PostgreSQL.

Tutte le versioni di Aurora PostgreSQL disponibili, incluse le versioni 10, 11, 12, 13, 14 e versioni successive, supportano il Salted Challenge Response Authentication Mechanism (SCRAM) per le password come alternativa a message digest (). MD5 Ti consigliamo di utilizzare SCRAM perché è più sicuro di. MD5 Per ulteriori informazioni, incluso come migrare le password degli utenti del database da SCRAM MD5 a SCRAM, consulta. Utilizzo delle crittografia password SCRAM per PostgreSQL

Informazioni su ruoli e autorizzazioni di PostgreSQL

Quando si crea un'istanza RDS del cluster Aurora PostgreSQL DB . AWS Management Console Per impostazione predefinita, verrà chiamato `postgres`, come mostrato nello screenshot seguente:



Anziché accettare il valore predefinito (`postgres`) è possibile scegliere un nome diverso. In tal caso, il nome scelto deve iniziare con una lettera e contenere da 1 a 16 caratteri alfanumerici. Per semplicità, facciamo riferimento a questo account utente principale utilizzando il suo valore predefinito (`postgres`) in tutta la Guida.

Se si utilizza il `create-db-cluster` AWS CLI anziché il AWS Management Console, si crea il nome utente passandolo con il `master-username` parametro. Per ulteriori informazioni, consulta Fase 2: creazione di un cluster di database Aurora PostgreSQL.

Sia che utilizzi l' AWS Management Console API Amazon RDS AWS CLI, che utilizzi il `postgres` nome predefinito o scelga un nome diverso, questo primo account utente del database è membro del `rds_superuser` gruppo e dispone di `rds_superuser` privilegi.

Argomenti

- Comprendere il ruolo rds_superuser
- Controllo dell'accesso utente al database PostgreSQL
- Delega e controllo della gestione delle password utente
- Utilizzo delle crittografia password SCRAM per PostgreSQL

Comprendere il ruolo rds_superuser

In PostgreSQL, un ruolo può definire un utente, un gruppo o un insieme di autorizzazioni specifiche concesse a un gruppo o a un utente per vari oggetti nel database. I comandi PostgreSQL `CREATE USER` e `CREATE GROUP` sono stati sostituiti dal comando `CREATE ROLE` più generico, ma con proprietà specifiche per distinguere gli utenti del database. Un utente del database può essere paragonato a un ruolo con il privilegio LOGIN.

> **Note**
>
> È comunque possibile continuare a utilizzare i comandi `CREATE USER` e `CREATE GROUP`. Per ulteriori informazioni, consulta la sezione relativa ai ruoli di database nella documentazione di PostgreSQL.

L'utente `postgres` è l'utente di database più privilegiato nel cluster di database Aurora PostgreSQL. Ha le caratteristiche definite dalla seguente istruzione `CREATE ROLE`.

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

Le proprietà `NOSUPERUSER`, `NOREPLICATION`, `INHERIT` e `VALID UNTIL 'infinity'` sono le opzioni predefinite per CREATE ROLE, se non diversamente specificato.

Per impostazione predefinita, `postgres` dispone dei privilegi concessi al `rds_superuser` ruolo e delle autorizzazioni per creare ruoli e database. Il ruolo `rds_superuser` consente all'utente `postgres` di eseguire le seguenti operazioni:

- Aggiungere le estensioni che sono disponibili per l'uso con Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo di estensioni e wrapper di dati esterni.
- Creare ruoli per gli utenti e concedere i relativi privilegi. Per ulteriori informazioni, consulta CREATE ROLE e GRANTnella documentazione di PostgreSQL.
- Creare database. Per ulteriori informazioni, consulta CREATE DATABASE nella documentazione di PostgreSQL.
- Concedere privilegi `rds_superuser` a ruoli utente che non dispongono di questi privilegi e revocare i privilegi, se necessario. Si consiglia di concedere questo ruolo solo agli utenti che eseguono attività superuser. In altre parole, è possibile concedere questo ruolo agli amministratori del database (DBAs) o agli amministratori di sistema.
- Concedere (e revocare) il ruolo `rds_replication` per gli utenti del database che non hanno il ruolo `rds_superuser`.
- Concedere (e revocare) il ruolo `rds_password` per gli utenti del database che non hanno il ruolo `rds_superuser`.
- Ottenere informazioni sullo stato di tutte le connessioni al database utilizzando la vista `pg_stat_activity`. Quando necessario, il ruolo `rds_superuser` può arrestare qualsiasi connessione utilizzando il comando `pg_terminate_backend` o `pg_cancel_backend`.

Nell'istruzione `CREATE ROLE postgres...`, si può vedere che il ruolo utente `postgres` non concede specificamente autorizzazioni PostgreSQL `superuser`. Aurora PostgreSQL è un servizio gestito e pertanto non è possibile accedere al sistema operativo host, né connettersi utilizzando l'account PostgreSQL`superuser`. Molte delle attività che richiedono l'accesso di tipo `superuser` su un PostgreSQL autonomo viene gestito automaticamente da Aurora.

Per ulteriori informazioni sulla concessione dei privilegi, consulta la sezione relativa al comando GRANT nella documentazione di PostgreSQL.

Il ruolo `rds_superuser` è uno dei diversi ruoli predefinito in uncluster di database Aurora PostgreSQL.

> **Note**
>
> In PostgreSQL 13 e versioni precedenti, i ruoli di default sono conosciuti come ruoli predefiniti.

L'elenco seguente fornisce alcuni degli altri ruoli predefiniti creati automaticamente per un nuovo cluster di database Aurora PostgreSQL. I ruoli predefiniti e i relativi privilegi non possono essere modificati. Non è possibile eliminare, rinominare o modificare i privilegi per questi ruoli predefiniti. Qualsiasi tentativo comporta la generazione di un errore.

- rds_password - Un ruolo in grado di modificare le password e configurare vincoli di password per gli utenti del database. Il `rds_superuser` ruolo viene concesso con questo ruolo per impostazione predefinita e può essere concesso agli utenti del database. Per ulteriori informazioni, consulta Controllo dell'accesso utente al database PostgreSQL.
  - Per le versioni di RDS per PostgreSQL precedenti alla 14`rds_password`, role può modificare le password e impostare vincoli di password per gli utenti del database e gli utenti con ruolo. `rds_superuser` A partire dalla versione 14 di RDS per PostgreSQL`rds_password`, role può modificare le password e impostare vincoli di password solo per gli utenti del database. Solo gli utenti con `rds_superuser` ruolo possono eseguire queste azioni su altri utenti con ruolo. `rds_superuser`
- rdsadmin – Un ruolo creato per gestire molte delle attività di gestione che l'amministratore con privilegi `superuser` esegue su un database PostgreSQL autonomo. Questo ruolo viene utilizzato internamente da Aurora PostgreSQL per molte attività di gestione.

Visualizzazione dei ruoli e dei relativi privilegi

È possibile visualizzare i ruoli predefiniti e i relativi privilegi nell'istanza DB RDS for PostgreSQL utilizzando comandi diversi a seconda della versione di PostgreSQL in uso.

Per PostgreSQL 15 e versioni precedenti

Connettiti alla tua istanza DB RDS per PostgreSQL e usa il comando in psql: `\du`

```
postgres=> \du
                                                             List of roles
    Role name    |                            Attributes                            |
                  Member of
-----------------+----------------------------------------------------------------
+-----------------------------------------------------
 postgres        | Create role, Create DB                                          +|
 {rds_superuser}
                 | Password valid until infinity                                    |
 rds_ad          | Cannot login                                                     | {}
 rds_iam         | Cannot login                                                     | {}
 rds_password    | Cannot login                                                     | {}
 rds_replication | Cannot login                                                     | {}
 rds_superuser   | Cannot login                                                     |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
                 | Password valid until infinity                                    |
```

Per PostgreSQL 16 e versioni successive

A partire da PostgreSQL 16, usa il comando per visualizzare informazioni dettagliate sull'`\drg +`appartenenza ai ruoli:

```
postgres=> \drg+
                            List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

Per verificare l'appartenenza ai ruoli indipendentemente dalla versione di PostgreSQL, puoi usare la seguente query SQL:

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

Nell'output, si vede che `rds_superuser` non è un ruolo utente del database (non può effettuare il login), ma ha i privilegi di molti altri ruoli. È inoltre possibile vedere che l'utente di database `postgres` è membro del ruolo `rds_superuser`. Come accennato in precedenza, `postgres` è il valore predefinito nella pagina Crea database della console Amazon RDS. Se si sceglie un altro nome, tale nome viene visualizzato nell'elenco dei ruoli.

> **Note**
>
> Aurora PostgreSQL versioni 15.2 e 14.7 hanno introdotto un funzionamento restrittivo del ruolo `rds_superuser`. A un utente Aurora PostgreSQL deve essere concesso il privilegio `CONNECT` per consentire la connessione al database corrispondente anche se tale utente è associato al ruolo `rds_superuser`. Prima delle versioni 14.7 e 15.2 di Aurora PostgreSQL, un utente era in grado di connettersi a qualsiasi database e tabella di sistema se disponeva del ruolo `rds_superuser`. Questo comportamento restrittivo è in linea con AWS gli impegni di Amazon Aurora per il miglioramento continuo della sicurezza.
> Aggiorna la rispettiva logica nelle tue applicazioni se sono state interessate dal miglioramento precedentemente descritto.

Controllo dell'accesso utente al database PostgreSQL

I nuovi database in PostgreSQL vengono sempre creati con un set predefinito di privilegi nel schema `public` del database, che consente a tutti gli utenti e i ruoli del database di creare oggetti. I privilegi predefiniti permettono agli utenti del database di connettersi al database e di creare tabelle temporanee durante la connessione.

Per controllare meglio l'accesso degli utenti alle istanze database create sul nodo primario del cluster Aurora PostgreSQL , si consiglia di revocare questi privilegi `public` predefiniti. Dopo averlo fatto,

è consigliabile concedere privilegi specifici agli utenti del database su base più granulare, come mostrato nella procedura seguente.

Per impostare ruoli e privilegi per una nuova istanza database

Si supponga di aver configurando un database in un cluster di database Aurora PostgreSQL per poter essere usato da diversi ricercatori, che dovranno avere l'accesso in lettura-scrittura al database.

1. Utilizzare `psql` (o pgAdmin) per connettersi all'istanza database primaria sul cluster di database Aurora PostgreSQL:

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Specifica la password, quando richiesto. Il client `psql` si connette e visualizza il database di connessione amministrativa predefinito `postgres=>` come prompt.

2. Per impedire agli utenti del database di creare oggetti nello schema `public`, eseguire le seguenti operazioni:

   ```
   postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
   REVOKE
   ```

3. Creare quindi una nuova istanza database:

   ```
   postgres=> CREATE DATABASE lab_db;
   CREATE DATABASE
   ```

4. Revocare tutti i privilegi dallo schema `PUBLIC` in questo nuovo database.

   ```
   postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
   REVOKE
   ```

5. Creare un ruolo per gli utenti del database.

   ```
   postgres=> CREATE ROLE lab_tech;
   CREATE ROLE
   ```

6. Concedere agli utenti del database con questo ruolo la possibilità di connettersi al database.

   ```
   postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
   ```

```
GRANT
```

7. Concedere a tutti gli utenti con il ruolo `lab_tech` tutti i privilegi per questo database.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. Creare utenti del database, come segue:

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. Concedere a questi due utenti i privilegi associati al ruolo lab_tech:

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
postgres=> GRANT lab_tech TO lab_user2;
GRANT ROLE
```

A questo punto, `lab_user1` e `lab_user2` possono connettersi al database `lab_db`. Questo esempio non segue le best practice per l'utilizzo aziendale, che potrebbero includere la creazione di più istanze database, schemi diversi e la concessione di autorizzazioni limitate. Per informazioni più complete e scenari aggiuntivi, consulta Gestione di utenti e ruoli PostgreSQL.

Per ulteriori informazioni sui privilegi in database PostgreSQL, consulta la sezione relativa al comando GRANT nella documentazione di PostgreSQL.

Delega e controllo della gestione delle password utente

Un amministratore di database (DBA) potrebbe voler delegare la gestione delle password utente. In alternativa, è possibile impedire agli utenti del database di modificare le password o di riconfigurare i vincoli delle password, ad esempio la durata della password. Per garantire che solo gli utenti del database scelti possano modificare le impostazioni della password, è possibile attivare la funzione di gestione delle password con restrizioni. Quando si attiva questa funzione, solo gli utenti del database a cui è stato concesso il ruolo `rds_password` saranno in grado di gestire le password.

> **Note**
>
> Per utilizzare la gestione delle password limitate, il cluster di database Aurora PostgreSQL deve eseguire Amazon Aurora PostgreSQL 10.6 o superiore.

Per impostazione predefinita, questa funzione è impostata su `off`, come mostrato di seguito:

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

Per attivare questa funzione, utilizzare un gruppo di parametri personalizzato e modificare l'impostazione per `rds.restrict_password_commands` su 1. Assicurarsi di riavviare l'istanza database principale Aurora PostgreSQL per implementare l'impostazione.

Con questa funzione attiva, i privilegi `rds_password` sono obbligatori per i seguenti comandi SQL:

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

La ridenominazione di un role (`ALTER ROLE myrole RENAME TO newname`) è inoltre limitata se la password utilizza l'algoritmo di MD5 hashing.

Con questa funzionalità attiva, se si tenta di eseguire uno di questi comandi SQL senza le autorizzazioni di ruolo `rds_password`, viene generato il seguente errore:

```
ERROR: must be a member of rds_password to alter passwords
```

Si consiglia di concedere i privilegi `rds_password` solo a ruoli utilizzati esclusivamente per la gestione delle password. Se si concedono i privilegi `rds_password` agli utenti del database sprovvisti dei privilegi `rds_superuser`, è necessario concedere loro anche l'attributo `CREATEROLE`.

Assicurarsi di verificare i requisiti della password come la scadenza e la complessità necessaria sul lato client. Se si utilizza la propria utilità lato client per le modifiche relative alla password, l'utilità deve essere membro di `rds_password` e avere i privilegi `CREATE ROLE`.

Utilizzo delle crittografia password SCRAM per PostgreSQL

Il Salted Challenge Response Authentication Mechanism (SCRAM) è un'alternativa all'algoritmo Message Digest (MD5) predefinito di PostgreSQL per la crittografia delle password. Il meccanismo di autenticazione SCRAM è considerato più sicuro di. MD5 Per ulteriori informazioni su questi due diversi approcci di protezione delle password, consulta la sezione relativa alla autenticazione password nella documentazione di PostgreSQL.

Si consiglia di utilizzare SCRAM anziché MD5 come schema di crittografia delle password per il cluster Aurora PostgreSQL DB. SCRAM è supportato in Aurora PostgreSQL versione 10 e in tutte le versioni principali e secondarie successive. È un meccanismo crittografico di richiesta/risposta che utilizza l'algoritmo scram-sha-256 per l'autenticazione e la crittografia delle password.

Potrebbe essere necessario aggiornare le librerie per le applicazioni client per supportare SCRAM. Ad esempio, le versioni JDBC precedenti alla 42.2.0 non supportano SCRAM. Per ulteriori informazioni, consulta PostgreSQL JDBC Driver nella documentazione di PostgreSQL JDBC Driver. Per un elenco di altri driver PostgreSQL e il supporto SCRAM, consulta Elenco dei driver nella documentazione di PostgreSQL.

Aurora PostgreSQL versione 14 e successive supportano scram-sha-256 per la crittografia password per impostazione predefinita per nuovi cluster database. Per queste versioni, il parametro predefinito del cluster DB group (`default.aurora-postgresql14`) ha il valore impostato su scram-sha-256. `password_encryption` SCRAM non è supportato per Aurora Serverless v1.

Configurazione di cluster database Aurora PostgreSQL per richiedere SCRAM

Per Aurora PostgreSQL 14.3 e versioni successive, puoi richiedere che il cluster database Aurora PostgreSQL accetti solo password che utilizzano l'algoritmo scram-sha-256.

> **Important**
>
> Per i proxy RDS esistenti con database PostgreSQL, se si modifica l'autenticazione del database in modo da utilizzare solo `SCRAM`, il proxy diventa non disponibile per un massimo di 60 secondi. Per evitare il problema, procedi in uno dei seguenti modi:
>
> - Assicurati che il database consenta entrambe le autenticazioni `SCRAM` e `MD5`.

- Per utilizzare solo l'autenticazione SCRAM, crea un nuovo proxy, esegui la migrazione del traffico dell'applicazione sul nuovo proxy, quindi elimina il proxy precedentemente associato al database.

Prima di apportare modifiche al sistema, assicurati di comprendere il processo completo, come segue:

- Ottieni informazioni su tutti i ruoli e la crittografia password per tutti gli utenti del database.
- Verifica le impostazioni dei parametri per il cluster database Aurora PostgreSQL per i parametri che controllano la crittografia password.
- Se il cluster database Aurora PostgreSQL utilizza un gruppo di parametri predefinito, devi creare un gruppo di parametri cluster database e applicarlo al cluster database Aurora PostgreSQL in modo da poter modificare i parametri quando necessario. Se il cluster database Aurora PostgreSQL utilizza un gruppo di parametri personalizzati, puoi modificare i parametri necessari in seguito nel processo, in base alle esigenze.
- Modifica il parametro `password_encryption` in `scram-sha-256`.
- Invia una notifica a tutti gli utenti del database per informarli che devono aggiornare le password. Esegui la stessa operazione per l'account `postgres`. Le nuove password sono crittografate e archiviate utilizzando l'algoritmo scram-sha-256.
- Verifica che tutte le password siano crittografate utilizzando come il tipo di crittografia.
- Se tutte le password utilizzano scram-sha-256, puoi modificare il `rds.accepted_password_auth_method` da `md5+scram` a `scram-sha-256`.

> **Warning**
>
> Dopo aver modificato `rds.accepted_password_auth_method` in scram-sha-256, gli eventuali utenti (ruoli) con password crittografate `md5` non potranno connettersi.

Preparazione alla richiesta di SCRAM per il cluster database Aurora PostgreSQL

Prima di apportare modifiche al cluster database Aurora PostgreSQL, controlla tutti gli account utente del database esistenti. Inoltre, controlla il tipo di crittografia utilizzato per le password. Puoi eseguire queste attività utilizzando l'estensione `rds_tools`. Per vedere quali versioni di PostgreSQL sono `rds_tools` supportate, consulta Versioni di estensione per Amazon RDS for PostgreSQL.

Per ottenere un elenco di utenti del database (ruoli) e metodi di crittografia password

1. Utilizza `psql` per connetterti all'istanza database principale del cluster database Aurora PostgreSQL , come mostrato di seguito.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Installa l'estensione `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. Ottieni un elenco di ruoli e crittografia.

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

L'output visualizzato è simile al seguente.

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

Creazione di un gruppo di parametri del cluster DB personalizzato

> **Note**
>
> Se il cluster database Aurora PostgreSQL utilizza già un gruppo di parametri personalizzati, non è necessario crearne uno nuovo.

Per una panoramica dei gruppi di parametri per Aurora, consulta Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.

Il tipo di crittografia password utilizzato per le password è impostato in un parametro, `password_encryption`. La crittografia consentita dal cluster database Aurora PostgreSQL è impostata in un altro parametro, `rds.accepted_password_auth_method`. La modifica di uno di questi rispetto ai valori predefiniti richiede la creazione di un gruppo di parametri del cluster database personalizzato e l'applicazione al cluster.

Per ulteriori informazioni, consulta Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.

Ora puoi associare il gruppo di parametri personalizzati all'istanza database.

Creare un gruppo di parametri del cluster database personalizzato

1. Utilizza il comando CLI `create-db-cluster-parameter-group` per creare il gruppo di parametri personalizzati per il cluster. Il seguente esempio utilizza `aurora-postgresql13` come l'origine per questo gruppo di parametri personalizzati.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-lab-scram-passwords' \
     --db-parameter-group-family aurora-postgresql13  --description 'Custom DB cluster parameter group for SCRAM'
   ```

   In Windows:

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-lab-scram-passwords" ^
     --db-parameter-group-family aurora-postgresql13  --description "Custom DB cluster parameter group for SCRAM"
   ```

   Ora puoi associare il gruppo di parametri personalizzati al cluster.

2. Utilizza il comando CLI `modify-db-cluster` per applicare questo gruppo di parametri personalizzati al cluster database Aurora PostgreSQL.

   In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster --db-cluster-identifier 'your-instance-name' \
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

In Windows:

```
aws rds modify-db-cluster --db-cluster-identifier "your-instance-name" ^
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

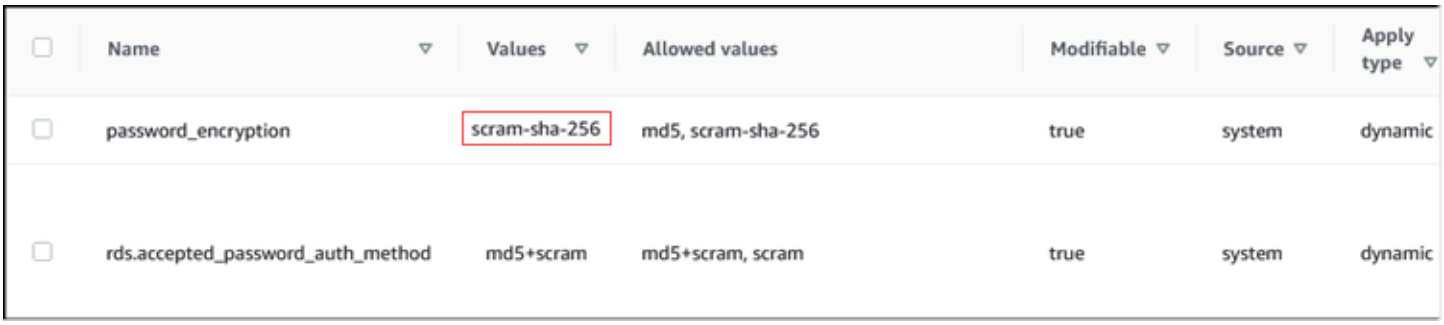
Per ripetere la sincronizzazione del cluster database Aurora PostgreSQL con il gruppo di parametri del cluster DB personalizzati, è necessario riavviare l'istanza principale e tutte le altre istanze del cluster.

Configurazione della crittografia password per utilizzare SCRAM

Il meccanismo di crittografia password utilizzato da un cluster database Aurora PostgreSQL è impostato nel gruppo di parametri del cluster DB nel parametro `password_encryption`. I valori consentiti sono unset, `md5` o `scram-sha-256`. Il valore predefinito dipende dalla versione di Aurora PostgreSQL come segue:

- Aurora PostgreSQL 14:-l'impostazione predefinita è `scram-sha-256`
- Aurora PostgreSQL 13: l'impostazione predefinita è `md5`

Con un gruppo di parametri del cluster database personalizzato collegato al cluster database Aurora PostgreSQL, puoi modificare i valori per il parametro di crittografia password.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Per modificare l'impostazione di crittografia password in scram-sha-256

- Modifica il valore della crittografia password in scram-sha-256, come mostrato di seguito. La modifica può essere applicata immediatamente perché il parametro è dinamico, quindi non è necessario un riavvio per rendere effettiva la modifica.

  In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name \
  'docs-lab-scram-passwords' --parameters
 'ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate'
```

  In Windows:

```
aws rds modify-db-parameter-group --db-parameter-group-name ^
  "docs-lab-scram-passwords" --parameters
 "ParameterName=password_encryption,ParameterValue=scram-
sha-256,ApplyMethod=immediate"
```

Migrazione delle password per i ruoli utente in SCRAM

Puoi migrare le password per i ruoli utente a SCRAM come descritto di seguito.

Per migrare le password degli utenti (ruoli) del database da a SCRAM MD5

1. Accedi come utente amministratore (nome utente predefinito, `postgres`) come mostrato di seguito.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Controlla l'impostazione del parametro `password_encryption` sull'istanza database RDS per PostgreSQL utilizzando il comando seguente.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 md5
 (1 row)
```

3. Modifica il valore di questo parametro in scram-sha-256. Si tratta di un parametro dinamico, quindi non è necessario riavviare l'istanza dopo aver apportato questa modifica. Controlla nuovamente il valore per essere certo che ora sia impostato su `scram-sha-256`, come descritto di seguito.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 scram-sha-256
 (1 row)
```

4. Invia una notifica a tutti gli utenti del database con la richiesta di modificare le password. Assicurati di modificare anche la password per l'account `postgres` (l'utente del database con privilegi `rds_superuser`).

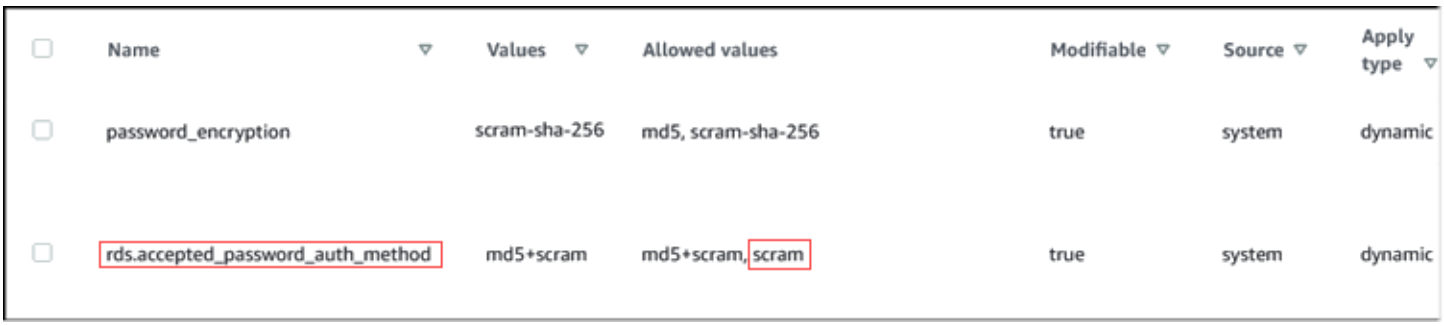
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

5. Ripeti il processo per tutti i database sul cluster database Aurora PostgreSQL.

Modifica del parametro per richiedere SCRAM

Questo è il passaggio finale del processo. Dopo aver apportato la modifica nella procedura seguente, gli eventuali account utente (ruoli) che ancora utilizzano la crittografia `md5` per le password non possono accedere al cluster database Aurora PostgreSQL.

Il `rds.accepted_password_auth_method` specifica il metodo di crittografia accettato dal cluster database Aurora PostgreSQL per una password utente durante il processo di accesso. Il valore predefinito è `md5+scram`, il che significa che entrambi i metodi sono accettati. Nell'immagine seguente, è disponibile l'impostazione predefinita per questo parametro.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

I valori consentiti per questo parametro sono `md5+scram` o `scram`. La modifica del valore di questo parametro in `scram` lo rende un requisito.

Modificare il valore del parametro per richiedere l'autenticazione SCRAM per le password

1. Verifica che tutte le password degli utenti del database per tutti i database sul cluster database Aurora PostgreSQL utilizzino `scram-sha-256` per la crittografia password. A questo proposito, esegui la query su `rds_tools` per il ruolo (utente) e il tipo di crittografia, come segue.

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. Ripeti la query su tutte le istanze database nel cluster database Aurora PostgreSQL.

   Se tutte le password utilizzano scram-sha-256, puoi procedere.

3. Modifica il valore dell'autenticazione password accettata in scram-sha-256, come riportato di seguito.

   In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-
lab-scram-passwords' \
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

   In Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-
lab-scram-passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Sicurezza dei dati Aurora PostgreSQL con SSL/TLS

Amazon RDS supporta la crittografia SSecure Socket Layer (SSL) e Transport Layer Security (TLS) per i cluster DB Aurora PostgreSQL. Utilizzo dell'SSL/TLS, you can encrypt a connection between your applications and your Aurora PostgreSQL DB clusters. You can also force all connections to your Aurora PostgreSQL DB cluster to use SSL/TLS. Amazon Aurora PostgreSQL supporta Transport Layer Security (TLS) versioni 1.1 e 1.2. Si consiglia di utilizzare TLS 1.2 per connessioni crittografate. Abbiamo aggiunto il supporto per la versione TLSv1 2.3 delle seguenti versioni di Aurora PostgreSQL:

- 15.3 e tutte le versioni successive
- 14.8 o versioni successive alla 14
- 13.11 o versioni successive alla 13
- 12.15 e versioni successive alla 12
- 11.20 e versioni successive alla 11

Per informazioni generali sul supporto SSL/TLS e sui database PostgreSQL, consulta Supporto SSL nella documentazione di PostgreSQL. Per informazioni sull'utilizzo della connessione SSL/TLS in JDBC, consulta Configurazione del client nella documentazione di PostgreSQL.

Argomenti

- Richiesta di una connessione SSL/TLS a un cluster DB Aurora PostgreSQL
- Determinazione dello stato di connessione SSL/TLS
- Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL

Il supporto SSL/TLS è disponibile in tutte le regioni per AWS Aurora PostgreSQL. Amazon RDS crea un SSL/TLS certificate for your Aurora PostgreSQL DB cluster when the DB cluster is created. If you enable SSL/TLS certificate verification, then the SSL/TLS certificate includes the DB cluster endpoint as the Common Name (CN) for the SSL/TLS certificato per proteggersi dagli attacchi di spoofing.

Per effettuare la connessione a un cluster DB Aurora PostgreSQL tramite SSL/TLS

1. Scaricare il certificato.

   Per ulteriori informazioni sul download dei certificati, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

2. Importare il certificato nel proprio sistema operativo.
3. Effettua la connessione al tuo cluster DB Aurora PostgreSQL su SSL/TLS.

   Quando ti connetti tramite SSL/TLS, il tuo client può scegliere di verificare o meno la catena di certificati. Se i parametri di connessione specificano `sslmode=verify-ca` o `sslmode=verify-full`, il client richiede che i certificati CA RDS siano nell'archivio attendibilità o facciano riferimento all'URL della connessione. Questo requisito è verificare la catena di certificati che firma il certificato del database.

   Quando un client, come psql o JDBC, è configurato con. SSL/TLS support, the client first tries to connect to the database with SSL/TLS by default. If the client can't connect with SSL/TLS, it reverts to connecting without SSL/TLS Per impostazione predefinita, l'opzione `sslmode` per i client JDBC e basati su libpq è impostata su `prefer`.

   Utilizzare il parametro `sslrootcert` come riferimento per il certificato, ad esempio `sslrootcert=rds-ssl-ca-cert.pem`.

Di seguito è riportato un esempio di utilizzo di psql per la connessione a un cluster DB Aurora PostgreSQL.

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
    "dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Richiesta di una connessione SSL/TLS a un cluster DB Aurora PostgreSQL

Per richiedere connessioni SSL/TLS al cluster Aurora PostgreSQL DB, usa il parametro. `rds.force_ssl`

- Per richiedere connessioni SSL/TLS, imposta il valore del parametro su 1 (attivo). `rds.force_ssl`
- Per disattivare le connessioni SSL/TLS richieste, imposta il valore del parametro su 0 (disattivato). `rds.force_ssl`

Il valore predefinito di questo parametro dipende dalla versione di Aurora PostgreSQL:

- Per Aurora PostgreSQL versione 17 e successive: il valore predefinito è 1 (attivo).
- Per le versioni 16 e precedenti di Aurora PostgreSQL: il valore predefinito è 0 (disattivato).

> **Note**
>
> Quando si esegue un aggiornamento della versione principale da Aurora PostgreSQL versione 16 o precedente alla versione 17 o successiva, il valore predefinito del parametro cambia da 0 (disattivato) a 1 (attivo). Questa modifica può causare errori di connettività per le applicazioni non configurate per SSL. È possibile ripristinare il comportamento predefinito precedente impostando questo parametro su 0 (disattivato).

Per ulteriori informazioni sulla gestione dei parametri, vedere.

L'aggiornamento del parametro `rds.force_ssl` imposta inoltre il parametro PostgreSQL `ssl` su 1 (attivato) e modifica il file `pg_hba.conf` del cluster DB per supportare la nuova configurazione SSL/TLS.

Quando il parametro `rds.force_ssl` è impostato su 1 per un cluster DB, vedrai un risultato simile al seguente quando effettui la connessione, per indicare che la crittografia SSL/TLS non è necessaria:

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Determinazione dello stato di connessione SSL/TLS

Lo stato crittografato della tua connessione è indicato nel banner di accesso quando ti connetti al cluster DB:

```
Password for user master:
psql (9.3.12)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.
```

```
postgres=>
```

Puoi anche caricare l'estensione `sslinfo` e quindi richiamare la funzione `ssl_is_used()` per determinare se viene utilizzata la crittografia SSL/TLS. La funzione restituisce `t` se la connessione utilizza la crittografia SSL/TLS, altrimenti restituisce `f`.

```
postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
---------
t
(1 row)
```

Puoi utilizzare il comando `select ssl_cipher()` per determinare la crittografia SSL/TLS:

```
postgres=> select ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

Se abiliti `set rds.force_ssl` e riavvi il tuo cluster DB, le connessioni non SSL vengono rifiutate con il seguente messaggio:

```
$ export PGSSLMODE=disable
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database
 "postgres", SSL off
$
```

Per informazioni sull'opzione `sslmode`, consulta l'argomento relativo alle funzioni di controllo della connessione del database nella documentazione di PostgreSQL.

Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di cifratura che si desidera consentire per proteggere le connessioni SSL/TLS client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. Ciò aiuta a prevenire l'uso di crittografie obsolete o non sicure.

Le suite di cifratura configurabili sono supportate nelle versioni 11.8 e successive di Aurora PostgreSQL.

Per specificare l'elenco di cifrature consentite per la crittografia delle connessioni, modifica il parametro del cluster `ssl_ciphers`. Imposta il `ssl_ciphers` parametro su una stringa di valori di cifratura separati da virgole in un gruppo di parametri del cluster utilizzando l'API RDS AWS Management Console AWS CLI, the o. Per impostare i parametri del cluster, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.

La tabella seguente mostra le cifrature supportate per le versioni valide del motore Aurora PostgreSQL.

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
9.6, 10.20 e versioni precedenti, 11.15 e versioni precedenti, 12.10 e versioni precedenti, 13.6 e versioni precedenti	DHE-RSA- -SHA AES128	Sì	No	No
	DHE-RSA- AES128 -SHA256	No	Sì	No
	DHE-RSA- -GCM-AES128 SHA256	No	Sì	No
	DHE-RSA- AES256 -SHA	No	Sì	No
	DHE-RSA- AES256 -SHA256	No	Sì	No
		Sì	Sì	No
		No	Sì	No
		No	Sì	No

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	DHE-RSA- -GCM- AES256 SHA384	Sì	Sì	No
	ECDHE-ECDSA- AES256 -SHA	No	Sì	No
	ECDHE-ECDSA- -GCM- AES256 SHA384	No	Sì	No
	ECDH-RSA- - AES256 SHA384	Sì	Sì	No
	ECDHE-RSA- AES128 -SHA	No	Sì	No
	ECDHE-RSA- - AES128 SHA256	No	Sì	No
	ECDHE-RSA- -GCM AES128 - SHA256			
	ECDHE-RSA- AES256 -SHA			
	ECDHE-RSA- AES256 -GCM- SHA384			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
10.21, 11.16, 12.11, 13.7, 14.3 e 14.4	ECDHE-RSA-AES128 -SHATLS_ECDHE_RSA_CON_AES_128_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_256_GCM_SHA384	No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
	TLS_ECDHE_RSA_CON_	No	Sì	No

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	AES_256_GCM_ SHA384 TLS_RSA_C ON_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_C ON_AES_128_GCM_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_CON_ 0_ CHACHA2 05_ POLY13 SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
10.22, 11.17, 12.12, 13.8, 14.5 e 15.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	Sì	Sì	No
		No	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_256_GCM_SHA384	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA TLS_ECDHE _RSA_CON_ AES_256_GCM_ SHA384 TLS_RSA_C ON_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_C ON_AES_128_GCM_ SHA256 TLS_RSA_C ON_AES_128_CBC_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_CON_ 0_ CHACHA2 05_ POLY13 SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
11.20, 12.15, 13.11, 14.8, 15.3, 16.1 e versioni successive	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256	Sì	Sì	No
		No	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_CON_AES_256_GCM_SHA384	No	Sì	No
		Sì	Sì	No
		No	Sì	No
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Sì	Sì	No
		Sì	Sì	No
	TLS_ECDHE_RSA_CON_AES_128_CBC_SHA256	No	Sì	No
		No	No	Sì
		No	No	Sì
	TLS_ECDHE_RSA_CON_AES_128_GCM_SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA TLS_ECDHE _RSA_CON_ AES_256_GCM_ SHA384 TLS_RSA_C ON_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_C ON_AES_128_GCM_ SHA256 TLS_RSA_C ON_AES_128_CBC_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_CON_ 0_ CHACHA2 05_ POLY13 SHA256 TLS_AES_128_GCM_ SHA256			

Versioni del motore Aurora PostgreSQL	Cifrature supportate	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_AES_256_GCM_SHA384			

È inoltre possibile utilizzare il comando CLI describe-engine-default-cluster-parameters per determinare quali suite di crittografia sono attualmente supportate per una famiglia di gruppi di parametri specifica. L'esempio seguente mostra come ottenere i valori consentiti per il parametro del cluster `ssl_cipher` per Aurora PostgreSQL 11.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
postgresql11

    ...some output truncated...
 {
  "ParameterName": "ssl_ciphers",
  "Description": "Sets the list of allowed TLS ciphers to be used on secure
 connections.",
  "Source": "engine-default",
  "ApplyType": "dynamic",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,
  ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-
SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-
SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384,

 TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,T

 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AE
  "IsModifiable": true,
  "MinimumEngineVersion": "11.8",
  "SupportedEngineModes": [
   "provisioned"
  ]
 },
    ...some output truncated...
```

Il parametro `ssl_ciphers` è impostato per tutte le suite di cifrature consentite. Per ulteriori informazioni sulla crittografia, consulta la variabile ssl_ciphers nella documentazione di PostgreSQL.

Aggiornamento delle applicazioni per la connessione ai cluster DB Aurora PostgreSQL utilizzando nuovi certificati SSL/TLS

A partire dal 13 gennaio 2023, sono stati pubblicati da Amazon RDS nuovi certificati dell'autorità di certificazione (CA) per la connessione ai cluster database Aurora utilizzando Secure Socket Layer o Transport Layer Security (SSL/TLS). Di seguito sono disponibili le informazioni sull'aggiornamento delle applicazioni per utilizzare i nuovi certificati.

Le informazioni in questo argomento possono aiutarti a stabilire se le applicazioni client utilizzano SSL/TLS per connettersi ai cluster DB. In caso affermativo, puoi determinare anche se le applicazioni richiedono la verifica del certificato per la connessione.

> **Note**
>
> Alcune applicazioni sono configurate per connettersi ai cluster DB Aurora PostgreSQL solo se sono in grado di verificare il certificato del server.
> Per queste applicazioni, è necessario aggiornare gli archivi di trust delle applicazioni client per includere i nuovi certificati CA.

Dopo aver aggiornato i certificati CA negli archivi attendibilità delle applicazioni client, puoi ruotare i certificati nei cluster DB. Consigliamo vivamente di testare queste procedure in un ambiente di sviluppo o di gestione temporanea prima di implementarle negli ambienti di produzione.

Per ulteriori informazioni sulla rotazione dei certificati, consulta Rotazione del certificato SSL/TLS. Per ulteriori informazioni sul download, consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Per informazioni sull'utilizzo di SSL/TLS con i cluster DB PostgreSQL, consulta Sicurezza dei dati Aurora PostgreSQL con SSL/TLS.

Argomenti

- Determinare se un'applicazione si connette al cluster DB Aurora PostgreSQL utilizzando SSL
- Determinare se un client richiede la verifica del certificato per la connessione
- Aggiornare l'archivio di trust delle applicazioni
- Utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni

Determinare se un'applicazione si connette al cluster DB Aurora PostgreSQL utilizzando SSL

Controlla la configurazione del cluster DB per individuare il valore del parametro `rds.force_ssl`. Per impostazione predefinita, il parametro `rds.force_ssl` è impostato su `0` (disattivato). Se il parametro `rds.force_ssl` è impostato su 1 (attivato), i client devono utilizzare SSL/TLS per le connessioni. Per ulteriori informazioni sui gruppi di parametri, consultare .

Se `rds.force_ssl` non è impostato su 1 (attivato), esegui una query sulla visualizzazione `pg_stat_ssl` per ottenere le connessioni che utilizzano SSL. Ad esempio, la query seguente restituisce solo le connessioni SSL e le informazioni sui client che utilizzano SSL.

```
select datname, usename, ssl, client_addr from pg_stat_ssl inner join pg_stat_activity
 on pg_stat_ssl.pid = pg_stat_activity.pid where ssl is true and usename<>'rdsadmin';
```

Vengono visualizzate solo le righe che utilizzano connessioni SSL/TLS con informazioni sulla connessione. Di seguito è riportato un output di esempio.

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

La query precedente visualizza solo le connessioni in uso al momento della query. L'assenza di risultati non indica che nessuna applicazione stia utilizzando connessioni SSL. Altre connessioni SSL potrebbero essere stabilite in un momento diverso.

Determinare se un client richiede la verifica del certificato per la connessione

Quando un client, come psql o JDBC, è configurato con il supporto SSL, per impostazione predefinita tenta innanzitutto di connettersi al database con SSL. Se il client non riesce a connettersi con SSL, torna a connettersi senza SSL. La `sslmode` modalità predefinita utilizzata sia per i client basati su libpq (come psql) che per JDBC è impostata su. `prefer` Il certificato sul server viene verificato solo quando `sslrootcert` viene fornito con set to o. `sslmode` `verify-ca` `verify-full` Se il certificato non è valido viene generato un errore.

PGSSLROOTCERTDa utilizzare per verificare il certificato con la variabile di PGSSLMODE ambiente, PGSSLMODE impostata su `verify-ca` o`verify-full`.

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U primaryuser -d postgres
```

Utilizza l'`sslrootcert`argomento per verificare il certificato `sslmode` nel formato della stringa di connessione, con `sslmode` impostato su `verify-ca` o`verify-full`.

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=primaryuser dbname=postgres"
```

Ad esempio, nel caso precedente, se utilizzi un certificato root non valido, sul client viene visualizzato un errore simile al seguente.

```
psql: SSL error: certificate verify failed
```

Aggiornare l'archivio di trust delle applicazioni

Per informazioni sull'aggiornamento dell'archivio attendibilità per le applicazioni PostgreSQL, consulta l'argomento relativo alle connessioni TCP/IP sicure con SSL nella documentazione di PostgreSQL.

> **Note**
>
> Quando aggiorni l'archivio di trust puoi conservare i certificati meno recenti oltre ad aggiungere i nuovi certificati.

Aggiornare l'archivio di trust delle applicazioni per JDBC

Puoi aggiornare l'archivio di trust delle applicazioni che utilizzano JDBC per le connessioni SSL/TLS.

Per ulteriori informazioni sul download del certificato root, consulta Utilizzo SSL/TLS per crittografare una connessione a un'.

Per gli script di esempio che importano i certificati, consulta Script di esempio per l'importazione di certificati nel tuo archivio di trust.

Utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni

Di seguito vengono fornite le informazioni sull'utilizzo delle connessioni SSL/TLS per diversi tipi di applicazioni.

- psql

  Il client viene invocato dalla riga di comando specificando le opzioni come stringa di connessione o variabili di ambiente. Le opzioni rilevanti per le connessioni SSL/TLS sono `sslmode` (variabile di ambiente `PGSSLMODE`), `sslrootcert` (variabile di ambiente `PGSSLROOTCERT`).

  Per l'elenco completo delle opzioni, consulta l'argomento relativo alle parole chiave dei parametri nella documentazione di PostgreSQL. Per l'elenco completo delle variabili di ambiente, consulta l'argomento relativo alle variabili di ambiente nella documentazione di PostgreSQL.

- pgAdmin

  Questo client basato su browser fornisce un'interfaccia più intuitiva per la connessione a un database PostgreSQL.

  Per informazioni sulla configurazione delle connessioni, consulta la documentazione di pgAdmin.

- JDBC

  JDBC abilita le connessioni al database con le applicazioni Java.

  Per informazioni generali sulla connessione a un database PostgreSQL con JDBC, consulta l'argomento relativo alla connessione al database nella documentazione di PostgreSQL. Per informazioni sulla connessione con SSL/TLS, consulta l'argomento relativo alla configurazione del client nella documentazione di PostgreSQL.

- Python

  Una popolare libreria Python per la connessione ai database PostgreSQL è `psycopg2`.

  Per informazioni sull'utilizzo di `psycopg2`, consulta la documentazione psycopg2. Per un breve tutorial su come connettersi a un database PostgreSQL, consulta il tutorial di Psycopg2. Le informazioni sulle opzioni accettate dal comando di connessione sono disponibili nell'argomento relativo al contenuto del modulo Psycopg2.

> **Important**
>
> Dopo aver stabilito che le connessioni al database utilizzano SSL/TLS e aver aggiornato l'archivio attendibile dell'applicazione, è possibile aggiornare il database per utilizzare i certificati 2048-g1. rds-ca-rsa Per istruzioni, consulta la fase 3 in Aggiornamento del certificato CA modificando l'istanza il cluster di database.

Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL

Puoi utilizzare Kerberos per autenticare gli utenti quando si connettonoal cluster di database che esegue PostgreSQL. A tale scopo, configura l' del cluster DB da utilizzare AWS Directory Service for Microsoft Active Directory per l'autenticazione Kerberos. AWS Directory Service for Microsoft Active Directory viene anche chiamato. AWS Managed Microsoft ADÈ una funzionalità disponibile con AWS Directory Service. Per ulteriori informazioni, vedi Cos'è AWS Directory Service? nella Guida all'AWS Directory Service amministrazione.

Per iniziare, crea una AWS Managed Microsoft AD directory per memorizzare le credenziali dell'utente. Per il cluster di database PostgreSQL, specifica quindi il dominio di Active Directory e altre informazioni. Quando gli utenti eseguono l'autenticazione con il cluster database PostgreSQL, le richieste di autenticazione vengono inoltrate alla directory AWS Managed Microsoft AD .

Mantenere tutte le credenziali nella stessa directory consente di ridurre il tempo e l'impegno. È disponibile una posizione centralizzata per archiviare e gestire le credenziali per più cluster di database. L'uso di una directory può inoltre migliorare il profilo di sicurezza complessivo.

Puoi inoltre accedere alle credenziali da Microsoft Active Directory on-premise. A tale scopo, crea una relazione di dominio trusting in modo che la directory AWS Managed Microsoft AD consideri attendibile Microsoft Active Directory on-premise. In questo modo, gli utenti possono accedere ai cluster PostgreSQL con la stessa esperienza SSO (Single Sign-On) Windows dei carichi di lavoro nella rete locale.

Un database può utilizzare Kerberos AWS Identity and Access Management (IAM) o entrambe le procedure di autenticazione Kerberos e IAM. Tuttavia, poiché l'autenticazione Kerberos e IAM forniscono metodi di autenticazione diversi, un utente del database specifico può accedere a un database utilizzando solo uno o l'altro metodo di autenticazione, ma non entrambi. Per ulteriori informazioni sull'autenticazione IAM, consulta Autenticazione del database IAM .

Argomenti

- Disponibilità di regioni e versioni
- Panoramica di Autenticazione Kerberos per cluster di database di PostgreSQL
- Configurazione dell'autenticazione Kerberos per cluster di database di PostgreSQL
- Gestione di un cluster Aurora PostgreSQL DB (RDS per PostgreSQL) per un'istanza DB PostgreSQL in un Directory
- Connessione a PostgreSQL con Autenticazione Kerberos
- Utilizzo dei gruppi di sicurezza AD per il controllo degli accessi di Aurora SQL Postgree

Disponibilità di regioni e versioni

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni di Aurora PostgreSQL con autenticazione Kerberos, consultare Autenticazione Kerberos con Aurora PostgreSQL.

Panoramica di Autenticazione Kerberos per cluster di database di PostgreSQL

Per configurare l'autenticazione Kerberos per un cluster di database di PostgreSQL, segui queste fasi, descritte dettagliatamente più avanti:

1. Utilizzare AWS Managed Microsoft AD per creare una AWS Managed Microsoft AD directory. È possibile utilizzare l' AWS Management Console AWS CLI, la o l' AWS Directory Service API per creare la directory. Assicurati di aprire le porte in uscita rilevanti nel gruppo di sicurezza della directory in modo che la directory possa comunicare con l'del cluster.
2. Crea un ruolo che fornisca ad Amazon Aurora l'accesso ad Amazon per effettuare chiamate alla tua directory. AWS Managed Microsoft AD A tale scopo, crea un ruolo AWS Identity and Access Management (IAM) che utilizzi la policy IAM gestita. `AmazonRDSDirectoryServiceAccess`

   Affinché il ruolo IAM consenta l'accesso, l'endpoint AWS Security Token Service (AWS STS) deve essere attivato nella AWS regione corretta per il tuo AWS account. AWS STS Gli endpoint sono tutti Regioni AWS attivi per impostazione predefinita e puoi utilizzarli senza ulteriori azioni. Per ulteriori informazioni, consulta Attivazione e disattivazione AWS STS in una AWS regione nella Guida per l'utente IAM.
3. Crea e configura gli utenti nella AWS Managed Microsoft AD directory utilizzando gli strumenti di Microsoft Active Directory. Per ulteriori informazioni sulla creazione di utenti in Active Directory,

consulta Gestire utenti e gruppi in AWS Managed Microsoft AD nella Guida all'AWS Directory Service amministrazione.

4. Se prevedi di localizzare la directory e l'istanza DB in diversi AWS account o cloud privati virtuali (VPCs), configura il peering VPC. Per ulteriori informazioni, consulta Che cos'è il peering di VPC? nella Amazon VPC Peering Guide.
5. Creare o modificare un cluster di database di PostgreSQL dalla console, da CLI o dall'API di RDS utilizzando uno dei seguenti metodi:
   - Creazione e connessione di un cluster di database Aurora PostgreSQL
   - Modifica di un cluster database Amazon Aurora
   - Ripristino da uno snapshot cluster database
   - Ripristino di un cluster di database a un determinato momento

   Puoi localizzare l' del cluster nello stesso Amazon Virtual Private Cloud (VPC) della directory o in un account AWS o VPC diverso. Quando crei o modifichi l' cluster database PostgreSQL, completa le seguenti operazioni:
   - Specifica l'identificativo del dominio (identificativo `d-*`) generato al momento della creazione della directory.
   - Specifica anche il nome del ruolo IAM creato.
   - Assicurati che il gruppo di sicurezza dell'istanza database possa ricevere traffico in entrata dal gruppo di sicurezza della directory.
6. Utilizzare le credenziali dell'utente master RDS per connettersi al cluster di database di PostgreSQL. Creare l'utente in PostgreSQL per l'identificazione esterna. Gli utenti identificati esternamente possono accedere al cluster database di PostgreSQL con l'autenticazione Kerberos.

Configurazione dell'autenticazione Kerberos per cluster di database di PostgreSQL

Per configurare l'autenticazione Kerberos, completa la procedura seguente.

Argomenti

- Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD
- Fase 2: (facoltativa) creazione di una relazione di trust tra Active Directory on-premise e AWS Directory Service
- Fase 3: creare un ruolo IAM per Amazon Aurora Amazon per accedere a AWS Directory Service

- Fase 4: creazione e configurazione di utenti
- Fase 5: abilitazione del traffico tra VPC tra la directory e l'istanza database
- Fase 6: creazione o modifica di un cluster database PostgreSQL
- Fase 7: creazione di utenti PostgreSQL per i principali Kerberos
- Fase 8: configurazione di un client PostgreSQL

Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD

AWS Directory Service crea una Active Directory completamente gestita nel AWS cloud. Quando crei una AWS Managed Microsoft AD directory, AWS Directory Service crea due controller di dominio e server DNS per te. I server di directory vengono creati in sottoreti diverse in un VPC. Questa ridondanza assicura che la directory rimanga accessibile anche se si verifica un errore.

Quando si crea una AWS Managed Microsoft AD AWS directory, Directory Service esegue le seguenti attività per conto dell'utente:

- Configura una Active Directory all'interno del VPC.
- Crea un account amministratore della directory con il nome utente `Admin` e la password specificata. Puoi utilizzare questo account per gestire le directory.

  > **Important**
  >
  > Assicurati di salvare questa password. AWS Directory Service non memorizza questa password e non può essere recuperata o reimpostata.

- Crea un gruppo di sicurezza per i controller della directory. Il gruppo di sicurezza deve consentire la comunicazione con il cluster database PostgreSQL.

All'avvio AWS Directory Service for Microsoft Active Directory, AWS crea un'unità organizzativa (OU) che contiene tutti gli oggetti della directory. Questa unità organizzativa, che ha lo stesso nome NetBIOS che hai immesso al momento della creazione della directory, si trova nella radice del dominio. La radice del dominio è di proprietà e gestita da AWS.

L'`Admin`account creato con la AWS Managed Microsoft AD directory dispone delle autorizzazioni per le attività amministrative più comuni dell'unità organizzativa:

- Creazione, aggiornamento o eliminazione di utenti

- Aggiungi risorse al dominio, come file server o server di stampa, e assegna le autorizzazioni per tali risorse a utenti dell'unità organizzativa
- Crea contenitori aggiuntivi OUs
- Delega dell'autorità
- Ripristino degli oggetti eliminati dal cestino di Active Directory
- Esegui i moduli Active Directory e Domain Name Service (DNS) per Windows PowerShell sul servizio Web Active Directory

L'account `Admin` dispone anche dei diritti per eseguire queste attività in tutto il dominio:

- gestione delle configurazioni DNS (aggiunta, eliminazione o aggiornamento di record, zone e server d'inoltro);
- visualizzazione di log di eventi DNS;
- visualizzazione di log di eventi di sicurezza.

Per creare una directory con AWS Managed Microsoft AD

1. Nel riquadro di navigazione della console AWS Directory Service, scegliere Directory, quindi selezionare Configurazione della directory.
2. Scegli AWS Managed Microsoft AD. AWS Managed Microsoft AD è l'unica opzione attualmente supportata per l'uso con Amazon Aurora RDS.
3. Scegli Next (Successivo).
4. Nella pagina Enter directory information (Inserisci le informazioni sulla directory) inserisci le seguenti informazioni:

   Edizione

   Scegliere l'edizione più adatta alle proprie esigenze.

   Nome DNS directory

   Il nome completo della directory, ad esempio **`corp.example.com`**.

   Nome NetBIOS della directory

   Nome breve opzionale della directory, ad esempio `CORP`.

Descrizione della directory

Descrizione opzionale della directory.

Password amministratore

La password dell'amministratore della directory. Con il processo di creazione della directory viene generato un account amministratore con nome utente `Admin` e questa password.

La password dell'amministratore della directory non può includere il termine "admin". La password distingue tra maiuscole e minuscole e la lunghezza deve essere compresa tra 8 e 64 caratteri. Deve anche contenere un carattere di almeno tre delle seguenti quattro categorie:

- Lettere minuscole (a–z)
- Lettere maiuscole (A–Z)
- Numeri (0–9)
- Caratteri non alfanumerici (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirm password (Conferma password)

Digitare di nuovo la password dell'amministratore.

> **Important**
>
> Assicurati di salvare questa password. AWS Directory Service non memorizza questa password e non può essere recuperata o reimpostata.

5. Scegli Next (Successivo).
6. Nella pagina Choose VPC and subnets (Scegli VPC e sottoreti) fornire le seguenti informazioni:

   VPC

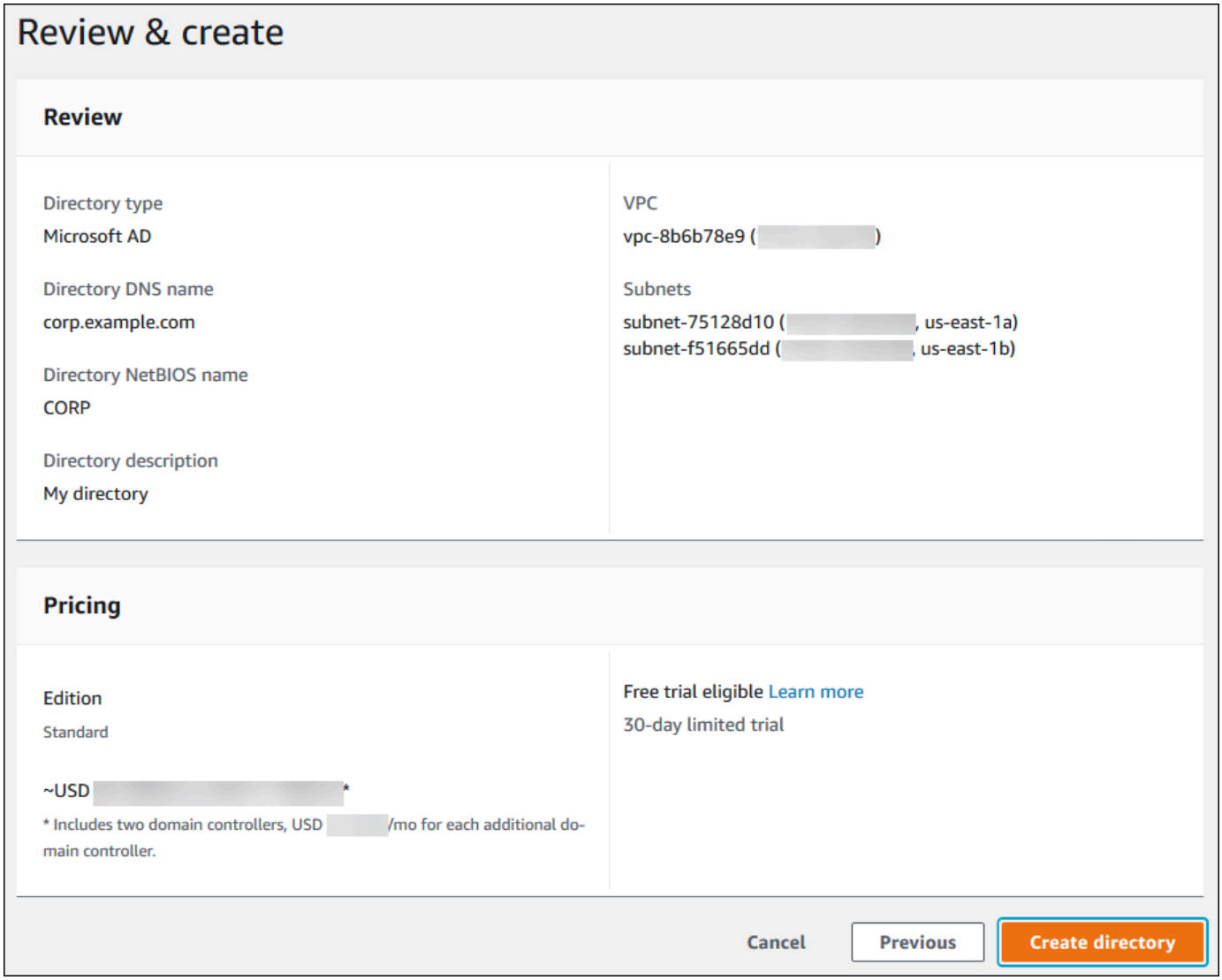
   Scegliere il VPC per la directory. È possibile creare il cluster di database di PostgreSQL in questo VPC o in uno diverso.

   Sottoreti

   Seleziona le sottoreti per i server di directory. Le due sottoreti devono trovarsi in diverse zone di disponibilità.

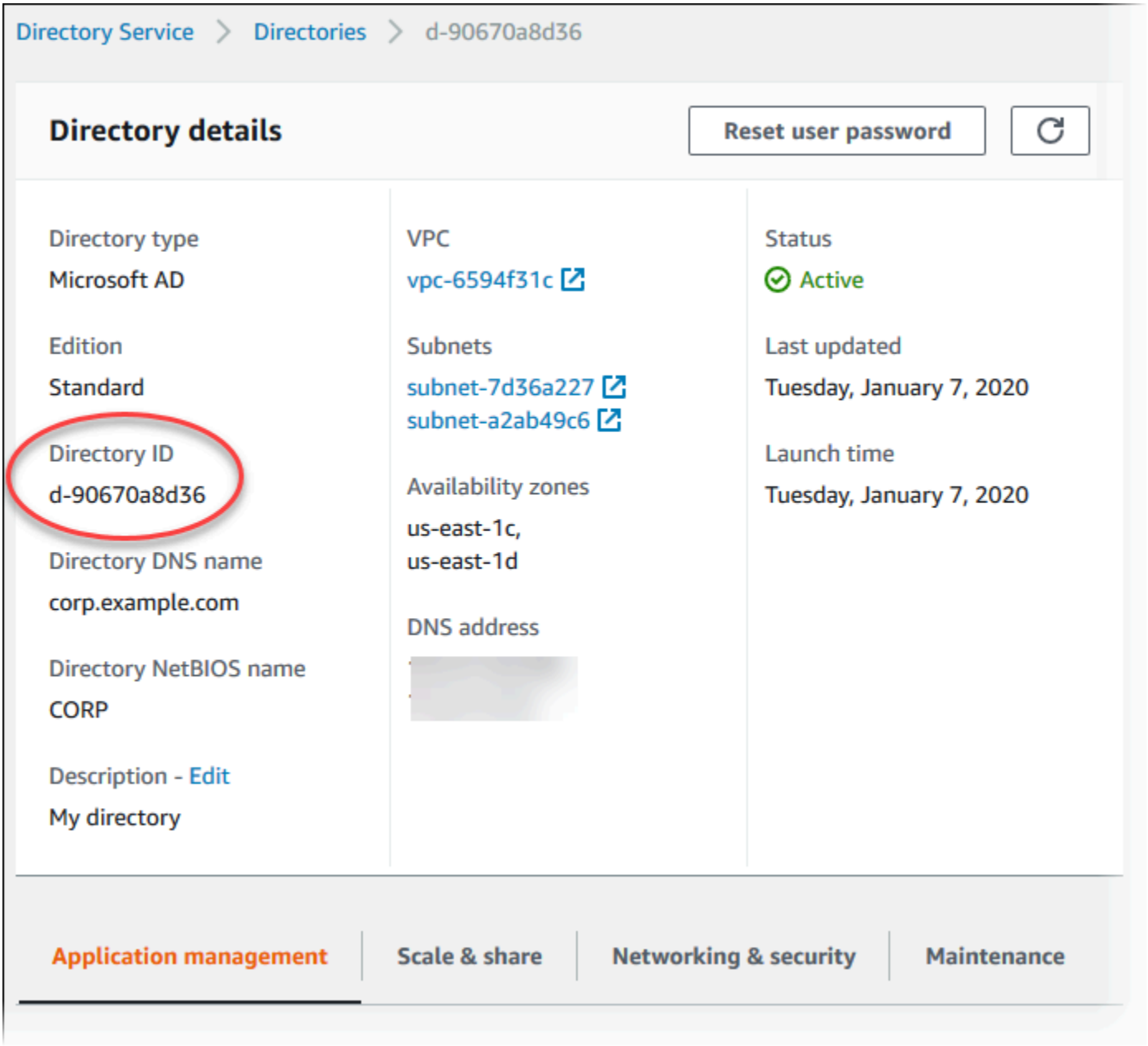
7. Seleziona Next (Successivo).

8. Verificare le informazioni della directory Se sono necessarie modifiche, seleziona Previous (Precedente) e apporta le modifiche. Quando le informazioni sono corrette, scegli Create Directory (Crea directory).



Per creare la directory sono necessari alcuni minuti. Una volta creata correttamente la directory, il valore Status (Stato) viene modificato in Active (Attivo).

Per consultare le informazioni sulla directory, selezionare l'ID della directory nell'elenco di directory. Prendere nota del valore Directory ID (ID directory). Questo valore è necessario per creare o modificare l'istanza database PostgreSQL.



Fase 2: (facoltativa) creazione di una relazione di trust tra Active Directory on-premise e AWS Directory Service

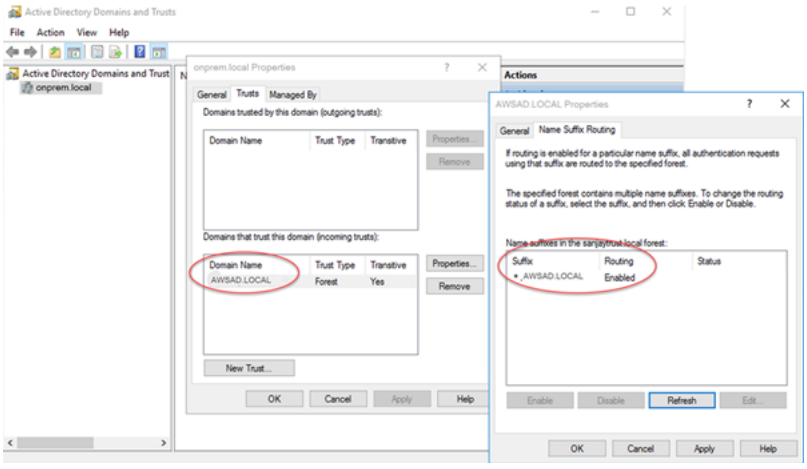
Se non prevedi di utilizzare Microsoft Active Directory locale, passa a Fase 3: creare un ruolo IAM per Amazon Aurora Amazon per accedere a AWS Directory Service.

Per ottenere l'autenticazione Kerberos utilizzando l'Active Directory locale, è necessario creare una relazione di dominio affidabile utilizzando un trust di foresta tra Microsoft Active Directory locale e la AWS Managed Microsoft AD directory (creata in). Passaggio 1: creare una directory utilizzando AWS Managed Microsoft AD L'attendibilità può essere unidirezionale, in cui la AWS Managed

Microsoft AD directory considera attendibile Microsoft Active Directory locale. Il trust può anche essere bidirezionale, in cui entrambe le Active Directory si considerano reciprocamente attendibili. Per ulteriori informazioni sulla configurazione dei trust utilizzando AWS Directory Service, vedere Quando creare una relazione di trust nella Administration Guide.AWS Directory Service

> **Note**
>
> Se utilizzi una Microsoft Active Directory locale:
>
> - I client Windows devono connettersi utilizzando il nome di dominio di nell'endpoint anziché rds.amazonaws.com AWS Directory Service . Per ulteriori informazioni, consulta Connessione a PostgreSQL con Autenticazione Kerberos.
> - I client Windows non possono connettersi con endpoint personalizzati Aurora. Per ulteriori informazioni, consulta Connessioni endpoint Amazon Aurora.
> - Per i database globali:
>   - I client Windows possono connettersi utilizzando endpoint di istanza o endpoint di cluster solo nel database primario del database globale. Regione AWS
>   - I client Windows non possono connettersi utilizzando gli endpoint del cluster in modalità secondaria. Regioni AWS

Assicurati che il nome di dominio di Microsoft Active Directory locale includa un routing del suffisso DNS che corrisponde alla nuova relazione di trust creata. Il risultato è mostrato nella screenshot seguente.



Fase 3: creare un ruolo IAM per Amazon Aurora Amazon per accedere a AWS Directory Service

Affinché Amazon Aurora Amazon possa AWS Directory Service chiamarti, il tuo AWS account deve avere un ruolo IAM che utilizzi la policy IAM gestita. `AmazonRDSDirectoryServiceAccess` Questo ruolo permette ad Amazon Aurora di effettuare chiamate ad AWS Directory Service. (Tieni presente che questo ruolo IAM a cui accedere AWS Directory Service è diverso dal ruolo IAM per Autenticazione del database IAM cui viene utilizzato.)

Quando crei un'istanza DB utilizzando l'account utente della console AWS Management Console e l'account utente della console dispone dell'`iam:CreateRole`autorizzazione, la console crea automaticamente il ruolo IAM necessario. In questo caso, il nome del ruolo è `rds-directoryservice-kerberos-access-role`. In caso contrario, è necessario creare manualmente il ruolo IAM. Quando crei questo ruolo IAM`Directory Service`, scegli e allega la policy AWS gestita `AmazonRDSDirectoryServiceAccess` ad esso.

Per ulteriori informazioni sulla creazione di ruoli IAM per un servizio, consulta Creating a role to delegate permissions to an AWS service nella IAM User Guide.

> **Note**
>
> Il ruolo IAM utilizzato per l'autenticazione Windows per RDS per Microsoft SQL Server non può essere utilizzato per Amazon Aurora.

Facoltativamente, puoi creare policy con le autorizzazioni richieste anziché utilizzare la policy `AmazonRDSDirectoryServiceAccess` gestita. In questo caso, il ruolo IAM deve avere la seguente policy di attendibilità IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Il ruolo deve anche disporre della seguente policy del ruolo IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Per l'attivazione Regioni AWS, utilizza i principali di servizio specifici della regione nelle policy di fiducia dei ruoli IAM. Quando crei una politica di fiducia per i servizi in queste regioni, specifica il codice regionale nell'entità del servizio.

L'esempio seguente mostra una politica di fiducia che include i principali di servizio specifici della regione:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Sostituisci REGION-CODE con il codice per la tua regione specifica. Ad esempio, utilizza i seguenti principali di servizio per la regione Asia Pacifico (Melbourne):

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

Fase 4: creazione e configurazione di utenti

Puoi creare gli utenti con lo strumento Utenti Active Directory e computer. Questo è uno degli strumenti di Active Directory Domain Services e Active Directory Lightweight Directory Services. Per ulteriori informazioni, consulta Add Users and Computers to the Active Directory domain (Aggiungere utenti e computer al dominio di Active Directory) nella documentazione di Microsoft. In questo caso, gli utenti sono individui o altre entità, ad esempio i computer, che fanno parte del dominio e le cui identità vengono conservate nella directory.

Per creare utenti in una AWS Directory Service directory, devi essere connesso a un' EC2 istanza Amazon basata su Windows che fa parte della AWS Directory Service directory. Allo stesso tempo, devi essere connesso come un utente che dispone di privilegi per creare utenti. Per ulteriori informazioni, consulta Creazione di un utente nella Guida di amministrazione di AWS Directory Service .

Fase 5: abilitazione del traffico tra VPC tra la directory e l'istanza database

Se prevedi di individuare la directory e il cluster di database nello stesso VPC, ignora questa fase e passa a Fase 6: creazione o modifica di un cluster database PostgreSQL.

Se prevedi di localizzare la directory e l'istanza DB in modo diverso VPCs, configura il traffico cross-VPC utilizzando il peering VPC o il Transit Gateway.AWS

La procedura seguente abilita il traffico tra un VPCs utilizzo e l'altro del peering VPC. Segui le istruzioni in Che cos'è il peering di VPC? nella Amazon Virtual Private Cloud Peering Guide.

Per abilitare il traffico tra VPC utilizzando il peering di VPC

1. Configurare le regole di routing VPC appropriate per garantire che il traffico di rete possa scorrere in entrambe le direzioni.
2. Assicurati che il gruppo di sicurezza dell'istanza database possa ricevere traffico in entrata dal gruppo di sicurezza della directory.
3. Assicurati che non sia presente una regola della lista di controllo accessi (ACL) di rete per bloccare il traffico.

Se la directory è di proprietà di un altro AWS account, è necessario condividerla.

Per condividere la cartella tra AWS account

1. Inizia a condividere la directory con l' AWS account in cui verrà creata l'istanza DB seguendo le istruzioni in Tutorial: Sharing your AWS Managed Microsoft AD directory for seamless EC2 Domain-join nella Administration Guide.AWS Directory Service
2. Accedi alla AWS Directory Service console utilizzando l'account per l'istanza DB e assicurati che il dominio abbia lo stato prima di procedere. `SHARED`
3. Dopo aver effettuato l'accesso alla AWS Directory Service console utilizzando l'account per l'istanza DB, annota il valore Directory ID. Utilizzare questo ID directory per aggiungere l'istanza database al dominio.

Fase 6: creazione o modifica di un cluster database PostgreSQL

Crea o modifica un cluster di database di PostgreSQL da usare con la directory. Puoi utilizzare la console, CLI o l'API di RDS per associare un cluster di database a una directory. Questa operazione può essere eseguita in uno dei seguenti modi:

- Crea un nuovo cluster DB PostgreSQL utilizzando la console, il comando create-db-clusterCLI o l'operazione Create RDS API. DBCluster Per istruzioni, consultare Creazione e connessione di un cluster di database Aurora PostgreSQL.
- Modifica un cluster PostgreSQL DB esistente utilizzando la console, il comando modify-db-clusterCLI o l'operazione Modify RDS API. DBCluster Per istruzioni, consultare Modifica di un cluster database Amazon Aurora.

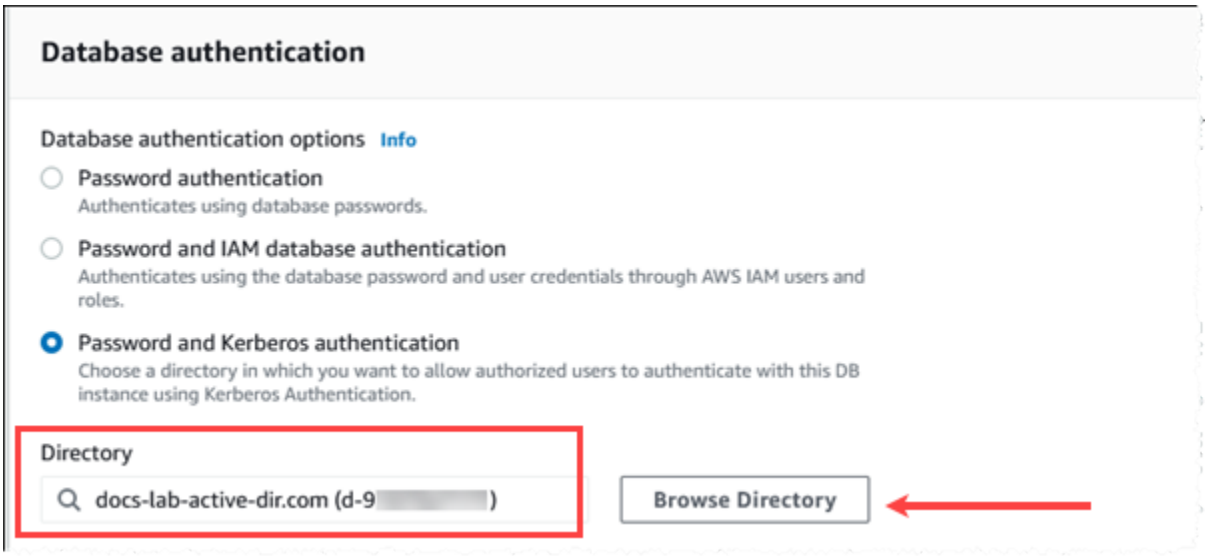
- Ripristina un cluster DB PostgreSQL da un'istantanea DB utilizzando la console, il comando CLI restore-db-cluster-from-db-snapshot o l'operazione dell'API Restore From RDS. DBCluster DBSnapshot Per istruzioni, consultare Ripristino da uno snapshot cluster database.
- Ripristina un cluster DB PostgreSQL utilizzando la console, il comando restore-db-instance-to-point-in-time CLI o l'operazione Restore RDS API. point-in-time DBCluster ToPointInTime Per istruzioni, consultare Ripristino di un cluster di database a un determinato momento.

L'autenticazione Kerberos è supportata solo per cluster di PostgreSQL DB in un VPC. Il cluster di database può trovarsi nello stesso VPC della directory o in un VPC diverso. Il cluster database deve utilizzare un gruppo di sicurezza che accetta traffico in ingresso e in uscita all'interno del VPC della directory, in modo che il cluster database possa comunicare con la directory.

> **Note**
>
> L'abilitazione dell'autenticazione Kerberos non è attualmente supportata sul cluster Aurora PostgreSQL DB durante la migrazione da RDS per PostgreSQL. È possibile abilitare l'autenticazione Kerberos solo su un cluster Aurora PostgreSQL DB autonomo.

Console

Quando utilizzi la console per creare, modificare o ripristinare un cluster di database, scegli Autenticazione Kerberos nella sezione Autenticazione database. Quindi scegli Sfoglia directory. Seleziona la directory o seleziona Crea una nuova directory per utilizzare il servizio directory.



AWS CLI

Quando si utilizza il AWS CLI, sono necessari i seguenti parametri affinché l' del cluster DB possa utilizzare la directory creata:

- Per il parametro `--domain`, utilizza l'identificatore di dominio (identificatore "d-*") generato durante la creazione della directory.
- Per il parametro `--domain-iam-role-name`, utilizza il ruolo creato che utilizza la policy IAM gestita `AmazonRDSDirectoryServiceAccess`.

Ad esempio, il comando CLI seguente modifica un cluster di database per utilizzare una directory.

```
aws rds  modify-db-cluster --db-cluster-identifier mydbinstance --domain d-Directory-ID
 --domain-iam-role-name role-name
```

> **Important**
>
> Se modifichi un cluster database per abilitare l'autenticazione Kerberos, riavvia il cluster database dopo aver apportato la modifica.

Fase 7: creazione di utenti PostgreSQL per i principali Kerberos

A questo punto, il cluster di database Aurora PostgreSQL viene aggiunto al dominio AWS Managed Microsoft AD . Gli utenti che hai creato nella directory in Fase 4: creazione e configurazione di utenti devono essere impostati come utenti del database PostgreSQL e devono essere concessi loro i privilegi per accedere al database. Puoi farlo accedendo come utente del database con privilegi `rds_superuser`. Ad esempio, se hai accettato i valori predefiniti quando hai creato il cluster database Aurora PostgreSQL, utilizzi `postgres`, come illustrato nei passaggi seguenti.

Per creare utenti del database PostgreSQL per i principali Kerberos

1. Usa `psql` per la connessione all'endpoint dell'istanza database del cluster database Aurora PostgreSQL utilizzando `psql`. L'esempio seguente utilizza l'account `postgres` predefinito per il ruolo `rds_superuser`.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Crea un nome utente del database per ogni principale Kerberos (nome utente di Active Directory) a cui desideri fornire l'accesso al database. Utilizza il nome utente canonico (identità) come definito nell'istanza Active Directory, ovvero un `alias` minuscolo (nome utente in Active Directory) e il nome maiuscolo del dominio Active Directory per quel nome utente. Il nome utente di Active Directory è un utente autenticato esternamente, quindi usa le virgolette intorno al nome come mostrato di seguito.

```
postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
CREATE ROLE
```

3. Autorizza il ruolo `rds_ad` per l'utente del database.

```
postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
GRANT ROLE
```

Dopo aver completato la creazione di tutti gli utenti PostgreSQL per le identità utente Active Directory, gli utenti possono accedere al cluster database Aurora PostgreSQL utilizzando le proprie credenziali Kerberos.

È necessario che gli utenti del database che si autenticano tramite Kerberos lo facciano da computer client membri del dominio Active Directory.

Gli utenti del database a cui è stato concesso il ruolo `rds_ad` non possono avere anche il ruolo `rds_iam`. Questo vale anche per le appartenenze nidificate. Per ulteriori informazioni, consulta Autenticazione del database IAM .

Configurazione del cluster database Aurora PostgreSQL per nomi utente senza distinzione tra maiuscole e minuscole

Le versioni di Aurora PostgreSQL 14.5, 13.8, 12.12 e 11.17 supportano il parametro PostgreSQL `krb_caseins_users`. Questo parametro supporta i nomi utente di Active Directory senza distinzione tra maiuscole e minuscole. Per impostazione predefinita, questo parametro è impostato su false, quindi i nomi utente vengono interpretati applicando la distinzione tra maiuscole e minuscole da Aurora PostgreSQL. Questo è il comportamento predefinito in tutte le precedenti versioni di Aurora PostgreSQL. Tuttavia, puoi impostare questo parametro su `true` nel gruppo di parametri del·cluster database personalizzato e consentire al cluster database Aurora PostgreSQL di interpretare i nomi utente, senza distinzione tra maiuscole e minuscole. Questa operazione può essere utile per gli utenti

del database, che a volte potrebbero digitare in modo errato le maiuscole e le minuscole del proprio nome utente durante l'autenticazione tramite Active Directory.

Per modificare il parametro `krb_caseins_users`, il cluster database Aurora PostgreSQL deve usare un gruppo di parametri del cluster database personalizzato. Per informazioni sull'utilizzo di un gruppo di parametri del·cluster database personalizzato, consulta .

È possibile utilizzare AWS CLI o il AWS Management Console per modificare l'impostazione. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.

Fase 8: configurazione di un client PostgreSQL

Per configurare un client PostgreSQL, procedi come indicato di seguito:

- Crea un file krb5.conf (o equivalente) che punti al dominio.
- Verifica che il traffico possa fluire tra l'host del client e AWS Directory Service. Utilizza un'utilità di rete come Netcat per le operazioni seguenti:
  - Verifica il traffico su DNS per la porta 53.
  - Verifica il traffico su TCP/UDP per la porta 53 e per Kerberos, che include le porte 88 e 464 per AWS Directory Service.
- Verifica che il traffico scorra senza problemi tra l'host client e l'istanza database sulla porta del database. Ad esempio, utilizza psql per connetterti e accedere al database.

Di seguito è riportato un esempio di contenuto krb5.conf per. AWS Managed Microsoft AD

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Di seguito è riportato un esempio di contenuto krb5.conf per una Microsoft Active Directory locale.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Gestione di un cluster Aurora PostgreSQL DB (RDS per PostgreSQL) per un'istanza DB PostgreSQL in un Directory

Puoi utilizzare la console, la CLI o l'API RDS per gestire l' e la sua relazione con Microsoft Active Directory. Ad esempio, puoi associare una Active Directory per abilitare l'autenticazione Kerberos. Puoi anche annullare l'associazione ad Active Directory per disabilitare l'autenticazione Kerberos. Puoi anche spostare un cluster di database affinché venga autenticata esternamente da una Microsoft Active Directory a un'altra.

Ad esempio, utilizzando la CLI, puoi effettuare quanto segue:

- Per riprovare ad abilitare l'autenticazione Kerberos per un'iscrizione non riuscita, usa il comando CLI. Specifica l'ID della directory dell'appartenenza corrente per l'opzione `--domain`.
- Per disabilitare l'autenticazione Kerberos su un'istanza DB, utilizzare il comando CLI . Specifica `none` per l'opzione `--domain`.
- Per spostare un'istanza DB da un dominio all'altro, usa il comando CLI. Specifica l'identificatore del nuovo dominio per l'opzione `--domain`.

Appartenenza al dominio

Dopo avere creato o modificato il cluster database, le istanze database diventano membri del dominio. È possibile visualizzare lo stato dell'appartenenza al dominio nella console o eseguendo il comando describe-db-instancesCLI. Lo stato dell'istanza di database può essere uno dei seguenti:

- `kerberos-enabled`: l'autenticazione Kerberos è abilitata nell'istanza database.
- `enabling-kerberos`— AWS sta abilitando l'autenticazione Kerberos su questa istanza DB.
- `pending-enable-kerberos`: l'abilitazione dell'autenticazione Kerberos è in corso su questa istanza database.
- `pending-maintenance-enable-kerberos`— AWS tenterà di abilitare l'autenticazione Kerberos sull'istanza DB durante la successiva finestra di manutenzione pianificata.
- `pending-disable-kerberos`: la disabilitazione dell'autenticazione Kerberos è in corso su questa istanza database.
- `pending-maintenance-disable-kerberos`— AWS tenterà di disabilitare l'autenticazione Kerberos sull'istanza DB durante la successiva finestra di manutenzione programmata.
- `enable-kerberos-failed`— Un problema di configurazione ha AWS impedito l'attivazione dell'autenticazione Kerberos sull'istanza DB. Correggi il problema di configurazione prima di inviare nuovamente il comando per modificare l'istanza database.
- `disabling-kerberos`— AWS sta disabilitando l'autenticazione Kerberos su questa istanza DB.

Una richiesta per abilitare l'autenticazione Kerberos potrebbe non andare a buon fine a causa di un problema di connettività di rete o un ruolo IAM non corretto. In alcuni casi, il tentativo di abilitare l'autenticazione Kerberos potrebbe non riuscire quando crei o modifichi un cluster di database. In questo caso, verifica di utilizzare il ruolo IAM corretto, quindi modifica il cluster di database per l'aggiunta al dominio.

Connessione a PostgreSQL con Autenticazione Kerberos

Puoi connetterti a PostgreSQL con l'autenticazione Kerberos tramite l'interfaccia pgAdmin o un'interfaccia a riga di comando come psql. Per ulteriori informazioni sulla connessione, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL. Per informazioni su come ottenere l'endpoint, il numero di porta e altri dettagli necessari per la connessione, consulta Visualizzazione degli endpoint per un cluster Aurora.

> **Note**
>
> L'autenticazione e la crittografia GSSAPI in PostgreSQL sono implementate dalla libreria Kerberos. `libkrb5.so` Funzionalità come `postgres_fdw` e si basano `dblink` anche su questa stessa libreria per le connessioni in uscita con autenticazione o crittografia Kerberos.

pgAdmin

Per utilizzare pgAdmin per connetterti a PostgreSQL con l'autenticazione Kerberos, completa la procedura seguente:

1. Avviare l'applicazione pgAdmin sul computer client.
2. Nella scheda Dashboard (Pannello di controllo) selezionare Add New Server (Aggiungi nuovo server).
3. Nella finestra di dialogo Crea - Server, immettere un nome nella scheda Generale per identificare il server in pgAdmin.
4. Nella scheda Connection (Connessione), immetti le informazioni seguenti relative al database Aurora PostgreSQL.
   - In Host, immetti l'endpoint per l'istanza di scrittura del cluster database Aurora PostgreSQL. Un endpoint è simile al seguente:

     ```
     AUR-cluster-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Per connettersi a un Microsoft Active Directory locale da un client Windows, si utilizza il nome di dominio di AWS Managed Active Directory anziché `rds.amazonaws.com` nell'endpoint host. Ad esempio, supponiamo che il nome di dominio per AWS Managed Active Directory sia. `corp.example.com` In Host, l'endpoint viene quindi specificato come segue:

     ```
     AUR-cluster-instance.111122223333.aws-region.corp.example.com
     ```

   - Per Porta, immettere la porta assegnata.
   - In Database di manutenzione immettere il nome del database iniziale a cui si connetterà il client.
   - In Nome utente, immettere il nome utente immesso per l'autenticazione Kerberos in Fase 7: creazione di utenti PostgreSQL per i principali Kerberos .
5. Scegli Save (Salva).

Psql

Per utilizzare psql per connetterti a PostgreSQL con l'autenticazione Kerberos, completare la procedura seguente:

1. Al prompt dei comandi, eseguire questo comando.

   ```
   kinit username
   ```

   Sostituire *username* con il nome utente. Al prompt, immettere la password per l'utente memorizzata in Microsoft Active Directory.

2. Se l'istanza del DB PostgreSQL utilizza un VPC accessibile pubblicamente, inserisci l'indirizzo IP per l'endpoint dell'istanza del DB nel tuo file sul client. `/etc/hosts` EC2 Ad esempio, i comandi seguenti ottengono l'indirizzo IP e lo inseriscono nel file `/etc/hosts`.

   ```
   % dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
   ;; Truncated, retrying in TCP mode.
   ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
   34.210.197.118

   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
   hosts
   ```

   Se utilizzi una Microsoft Active Directory locale da un client Windows, dovrai connetterti utilizzando un endpoint speciale. Invece di utilizzare il dominio Amazon `rds.amazonaws.com` nell'endpoint host, utilizza il nome di dominio di AWS Managed Active Directory.

   Ad esempio, supponiamo che il nome di dominio per la tua Active Directory AWS gestita sia. `corp.example.com` Quindi utilizzare il formato *PostgreSQL-endpoint.AWS-Region.corp.example.com* per l'endpoint e inserirlo nel file `/etc/hosts`.

   ```
   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
   hosts
   ```

3. Utilizzare il comando psql seguente per accedere a un cluster di database PostgreSQL integrata con Active Directory. Utilizzare un cluster o un endpoint dell'istanza.

   ```
   psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
   Region.rds.amazonaws.com postgres
   ```

Per accedere al cluster di database PostgreSQL da un client Windows utilizzando una Active Directory locale, utilizzare il comando psql seguente con il nome di dominio del passaggio precedente (`corp.example.com`):

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

Utilizzo dei gruppi di sicurezza AD per il controllo degli accessi di Aurora SQL Postgree

Dalle versioni SQL 14.10 e 15.5 di Aurora Postgre, il controllo degli accessi di Aurora Postgre può essere gestito tramite SQL AWS Directory Service per gruppi di sicurezza Microsoft Active Directory (AD). Le versioni precedenti di Aurora Postgre SQL supportano l'autenticazione basata su Kerberos con AD solo per singoli utenti. Ogni utente AD doveva essere fornito in modo esplicito al cluster DB per ottenere l'accesso.

Invece di assegnare esplicitamente il provisioning di ogni utente AD al cluster DB in base alle esigenze aziendali, puoi sfruttare i gruppi di sicurezza AD come spiegato di seguito:

- Gli utenti AD sono membri di vari gruppi di sicurezza AD in Active Directory. Questi non sono dettati dall'amministratore del cluster DB, ma si basano su requisiti aziendali e sono gestiti da un amministratore AD.
- Gli amministratori di cluster DB creano ruoli DB in istanze DB in base ai requisiti aziendali. Questi ruoli DB possono avere autorizzazioni o privilegi diversi.
- Gli amministratori di cluster DB configurano una mappatura dai gruppi di sicurezza AD ai ruoli DB per ogni cluster DB.
- Gli utenti DB possono accedere ai cluster DB utilizzando le proprie credenziali AD. L'accesso si basa sull'appartenenza al gruppo di sicurezza AD. Gli utenti AD ottengono o perdono automaticamente l'accesso in base all'appartenenza ai gruppi AD.

Prerequisiti

Assicurati di disporre di quanto segue prima di configurare l'estensione per i gruppi di AD Security:

- Configura l'autenticazione Kerberos per i cluster SQL Postgree DB. Per ulteriori informazioni, consulta Configurazione dell'autenticazione Kerberos per i cluster Postgre DB. SQL

  > **Note**
  >
  > Per i gruppi di sicurezza AD, salta il Passaggio 7: Crea SQL utenti Postgre per i principali Kerberos in questa procedura di configurazione.

- Gestione di un cluster DB in un dominio. Per ulteriori informazioni, vedere Gestione di un cluster DB in un dominio.

Configurazione dell'estensione pg_ad_mapping

Aurora Postgre fornisce ora SQL un'`pg_ad_mapping`estensione per gestire la mappatura tra gruppi di sicurezza AD e ruoli DB nel cluster Aurora Postgre. SQL Per ulteriori informazioni sulle funzioni fornite da, vedere. `pg_ad_mapping` Utilizzo delle funzioni dell'estensione `pg_ad_mapping`

Per configurare l'`pg_ad_mapping`estensione sul cluster Aurora Postgre SQL DB, è necessario innanzitutto aggiungerla `pg_ad_mapping` alle librerie condivise nel gruppo di parametri del cluster DB personalizzato per il cluster Aurora Postgre DB. SQL Per informazioni sulla creazione di un gruppo di parametri del cluster DB personalizzato, consulta. Successivamente, si installa l'`pg_ad_mapping`estensione. Le procedure in questa sezione mostrano come fare. Puoi utilizzare il plugin AWS Management Console o il AWS CLI.

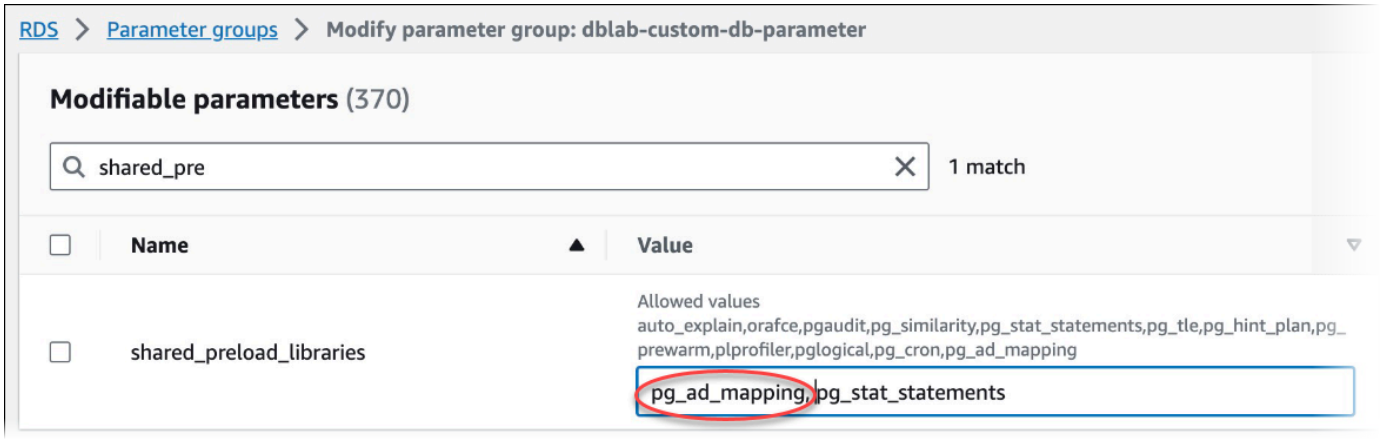
Per eseguire tutte queste attività, sono richieste autorizzazioni come il ruolo `rds_superuser`.

I passaggi seguenti presuppongono che il cluster Aurora Postgre SQL DB sia associato a un gruppo di parametri del cluster DB personalizzato.

Console

Per configurare l'estensione **`pg_ad_mapping`**

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli l'istanza Writer del cluster Aurora Postgre SQL DB.
3. Apri la scheda Configurazione per l'istanza del cluster writer Aurora Postgre SQL DB. Tra i dettagli dell'istanza, individua il collegamento Gruppo di parametri.
4. Scegli il link per aprire i parametri personalizzati associati al tuo cluster Aurora SQL Postgre DB.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pg_ad_mapping` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Riavvia l'istanza writer del cluster Aurora SQL Postgre DB in modo che la modifica al `shared_preload_libraries` parametro abbia effetto.
9. Quando l'istanza è disponibile, verifica che `pg_ad_mapping` sia stato inizializzato. Utilizzalo `psql` per connetterti all'istanza writer del tuo cluster Aurora Postgre SQL DB, quindi esegui il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_ad_mapping
(1 row)
```

10. Con `pg_ad_mapping` inizializzato, puoi ora creare l'estensione. È necessario creare l'estensione dopo aver inizializzato la libreria per iniziare a utilizzare le funzioni fornite da questa estensione.

```
CREATE EXTENSION pg_ad_mapping;
```

11. Chiudi la sessione `psql`.

```
labdb=> \q
```

AWS CLI

Per configurare pg_ad_mapping

Per configurare pg_ad_mapping usando il AWS CLI, si chiama l'modify-db-parameter-groupoperazione per aggiungere questo parametro nel gruppo di parametri personalizzato, come illustrato nella procedura seguente.

1. Utilizzate quanto segue AWS CLI comando da aggiungere `pg_ad_mapping` al `shared_preload_libraries` parametro.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_ad_mapping,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Usa quanto segue AWS CLI comando per riavviare l'istanza writer del cluster Aurora SQL Postgre DB in modo che pg_ad_mapping sia inizializzato.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando l'istanza è disponibile, verifica che `pg_ad_mapping` sia stato inizializzato. Utilizzalo `psql` per connetterti all'istanza writer del tuo cluster Aurora Postgre SQL DB, quindi esegui il comando seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_ad_mapping
(1 row)
```

   Con pg_ad_mapping inizializzato, ora puoi creare l'estensione.

```
CREATE EXTENSION pg_ad_mapping;
```

4. Chiudi la sessione in modo da poter utilizzare il `psql` AWS CLI.

```
labdb=> \q
```

Recupero del gruppo Active Directory in SID PowerShell

Un identificatore di sicurezza (SID) viene utilizzato per identificare in modo univoco un'entità di sicurezza o un gruppo di sicurezza. Ogni volta che un gruppo o un account di sicurezza viene creato in Active Directory, gli SID viene assegnato un. Per recuperare il gruppo di sicurezza AD SID da Active Directory, è possibile utilizzare il ADGroup cmdlet Get- dal computer client Windows che fa parte di quel dominio Active Directory. Il parametro Identity specifica il nome del gruppo Active Directory per ottenere il corrispondente. SID

L'esempio seguente restituisce il SID gruppo AD *adgroup1*.

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID

             SID
-----------------------------------------------
S-1-5-21-3168537779-1985441202-1799118680-1612
```

Mappatura del ruolo DB con il gruppo di sicurezza AD

È necessario fornire in modo esplicito i gruppi di sicurezza AD nel database come ruolo SQL Postgre DB. Un utente AD, che fa parte di almeno un gruppo di sicurezza AD assegnato, avrà accesso al database. Non dovresti concedere `rds_ad role` al gruppo AD un ruolo DB basato sulla sicurezza. L'autenticazione Kerberos per il gruppo di sicurezza verrà attivata utilizzando il suffisso del nome di dominio come *user1@example.com*. Questo ruolo DB non può utilizzare la password o IAM l'autenticazione per accedere al database.

> **Note**
>
> Gli utenti AD che hanno un ruolo DB corrispondente nel database a cui è stato concesso il `rds_ad` ruolo non possono accedere come parte del gruppo di sicurezza AD. Avranno accesso tramite il ruolo DB come utente individuale.

Ad esempio, accounts-group è un gruppo di sicurezza in AD in cui desideri fornire questo gruppo di sicurezza in Aurora Postgree come accounts-role. SQL

Gruppo di sicurezza AD	Ruolo Poster DB SQL
gruppo di conti	conti-ruolo

Quando si esegue il mapping del ruolo DB con il gruppo di sicurezza AD, è necessario assicurarsi che il ruolo DB abbia l'LOGINattributo impostato e disponga dei CONNECT privilegi sul database di accesso richiesto.

```
postgres => alter role accounts-role login;

ALTER ROLE
postgres => grant connect on database accounts-db to accounts-role;
```

L'amministratore può ora procedere alla creazione della mappatura tra il gruppo di sicurezza AD e il ruolo SQL Postgre DB.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', <SID>, <Weight>);
```

Per informazioni sul recupero del gruppo SID di sicurezza AD, vedi. Recupero del gruppo Active Directory in SID PowerShell

Potrebbero verificarsi casi in cui un utente AD appartiene a più gruppi, in tal caso, l'utente AD erediterà i privilegi del ruolo DB, a cui è stato assegnato il ruolo con il peso più elevato. Se i due ruoli hanno lo stesso peso, l'utente AD erediterà i privilegi del ruolo DB corrispondente alla mappatura aggiunta di recente. La raccomandazione è di specificare pesi che riflettano le autorizzazioni/privilegi relativi dei singoli ruoli DB. Più alti sono i permessi o i privilegi di un ruolo DB, maggiore è il peso che deve essere associato alla voce di mappatura. Ciò eviterà l'ambiguità di due mappature aventi lo stesso peso.

La tabella seguente mostra un esempio di mappatura dai gruppi di sicurezza AD ai ruoli DB di Aurora SQL Postgre.

Gruppo di sicurezza AD	Ruolo Poster DB SQL	Weight
gruppo di conti	conti-ruolo	7
gruppo di vendita	ruolo di vendita	10
gruppo di sviluppo	ruolo di sviluppo	7

Nell'esempio seguente, `user1` erediterà i privilegi di sales-role poiché ha il peso maggiore mentre `user2` erediterà i privilegi di `dev-role` come è stata creata la mappatura per questo ruolo, che hanno lo stesso peso di. `accounts-role` `accounts-role`

Username	Appartenenza al gruppo di sicurezza
user1	accounts-group sales-group
user2	gruppo di sviluppo accounts-group

I comandi psql per stabilire, elencare e cancellare le mappature sono mostrati di seguito. Attualmente non è possibile modificare una singola voce di mappatura. La voce esistente deve essere eliminata e la mappatura deve essere ricreata.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', 'S-1-5-67-890',
 7);
admin=>select pgadmap_set_mapping('sales-group', 'sales-role', 'S-1-2-34-560', 10);
admin=>select pgadmap_set_mapping('dev-group', 'dev-role', 'S-1-8-43-612', 7);

admin=>select * from pgadmap_read_mapping();

ad_sid       | pg_role        | weight | ad_grp
-------------+----------------+--------+---------------
S-1-5-67-890 | accounts-role | 7      | accounts-group
S-1-2-34-560 | sales-role    | 10     | sales-group
S-1-8-43-612 | dev-role      | 7      | dev-group
(3 rows)
```

Registrazione/controllo dell'identità degli utenti AD

Utilizza il comando seguente per determinare il ruolo del database ereditato dall'utente corrente o dalla sessione:

```
postgres=>select session_user, current_user;

session_user | current_user
-------------+--------------
dev-role     | dev-role

(1 row)
```

Per determinare l'identità principale di sicurezza AD, utilizzare il comando seguente:

```
postgres=>select principal from pg_stat_gssapi where pid = pg_backend_pid();

 principal
-------------------------
 user1@example.com

(1 row)
```

Attualmente, l'identità dell'utente AD non è visibile nei registri di controllo. Il `log_connections` parametro può essere abilitato per registrare l'istituzione della sessione DB. Per ulteriori informazioni, vedere log_connections. L'output a tale scopo include l'identità dell'utente AD, come illustrato di seguito. Il backend PID associato a questo output può quindi aiutare ad attribuire le azioni all'utente AD effettivo.

```
pgrole1@postgres:[615]:LOG: connection authorized: user=pgrole1
 database=postgres application_name=psql GSS (authenticated=yes, encrypted=yes,
 principal=Admin@EXAMPLE.COM)
```

Limitazioni

- L'ID Microsoft Entra noto come Azure Active Directory non è supportato.

Utilizzo delle funzioni dell'estensione **pg_ad_mapping**

`pg_ad_mapping`l'estensione ha fornito supporto alle seguenti funzioni:

pgadmap_set_mapping

Questa funzione stabilisce la mappatura tra il gruppo di sicurezza AD e il ruolo del database con un peso associato.

Sintassi

```
pgadmap_set_mapping(
ad_group,
db_role,
ad_group_sid,
weight)
```

Argomenti

Parametro	Descrizione
ad_group	Nome del gruppo AD. Il valore non può essere una stringa nulla o vuota.
db_role	Ruolo del database da mappare al gruppo AD specificato. Il valore non può essere una stringa nulla o vuota.
ad_group_sid	Identificatore di sicurezza utilizzato per identificare in modo univoco il gruppo AD. Il valore inizia con 'S-1-' e non può essere una stringa nulla o vuota. Per ulteriori informazioni, consulta Recupero del gruppo Active Directory in SID PowerShell.
peso	Peso associato al ruolo del database. Il ruolo con il peso più elevato ha la precedenza quando l'utente è membro di più gruppi. Il valore predefinito del peso è 1.

Tipo restituito

None

Note per l'utilizzo

Questa funzione aggiunge una nuova mappatura dal gruppo di sicurezza AD al ruolo del database. Può essere eseguita solo sull'istanza DB principale del cluster DB da un utente con privilegio rds_superuser.

Esempi

```
postgres=> select pgadmap_set_mapping('accounts-group','accounts-
role','S-1-2-33-12345-67890-12345-678',10);

pgadmap_set_mapping

(1 row)
```

pgadmap_read_mapping

Questa funzione elenca le mappature tra il gruppo di sicurezza AD e il ruolo DB che sono state impostate utilizzando la funzione. `pgadmap_set_mapping`

Sintassi

```
pgadmap_read_mapping()
```

Argomenti

None

Tipo restituito

Parametro	Descrizione
ad_group_sid	Identificatore di sicurezza utilizzato per identificare in modo univoco il gruppo AD. Il valore inizia con 'S-1-' e non può essere una stringa nulla o vuota. Per ulteriori informazioni,

Parametro	Descrizione
	vedere .accounts-role@example.com Recupero del gruppo Active Directory in SID PowerShell
db_role	Ruolo del database da mappare al gruppo AD specificato. Il valore non può essere una stringa nulla o vuota.
peso	Peso associato al ruolo del database. Il ruolo con il peso più elevato ha la precedenza quando l'utente è membro di più gruppi. Il valore predefinito del peso è 1.
ad_group	Nome del gruppo AD. Il valore non può essere una stringa nulla o vuota.

Note per l'utilizzo

Chiama questa funzione per elencare tutte le mappature disponibili tra il gruppo di sicurezza AD e il ruolo DB.

Esempi

```
postgres=> select * from pgadmap_read_mapping();

ad_sid                              | pg_role       | weight | ad_grp
------------------------------------+---------------+--------+------------------
S-1-2-33-12345-67890-12345-678      | accounts-role | 10     | accounts-group
(1 row)

(1 row)
```

pgadmap_reset_mapping

Questa funzione ripristina una o tutte le mappature impostate utilizzando function. `pgadmap_set_mapping`

Sintassi

```
pgadmap_reset_mapping(
ad_group_sid,
```

```
db_role,
weight)
```

Argomenti

Parametro	Descrizione
ad_group_sid	Identificatore di sicurezza utilizzato per identificare in modo univoco il gruppo AD.
db_role	Ruolo del database da mappare al gruppo AD specificato.
peso	Peso associato al ruolo del database.

Se non viene fornito alcun argomento, tutte le mappature dei ruoli da gruppo AD a DB vengono ripristinate. È necessario fornire tutti gli argomenti o non specificarne nessuno.

Tipo restituito

None

Note per l'utilizzo

Chiama questa funzione per eliminare uno specifico gruppo AD nella mappatura dei ruoli del DB o per ripristinare tutte le mappature. Questa funzione può essere eseguita solo sull'istanza DB principale del cluster DB da un utente con privilegi. `rds_superuser`

Esempi

```
postgres=> select * from pgadmap_read_mapping();

   ad_sid                       | pg_role      | weight      | ad_grp
--------------------------------+--------------+-------------+-------------------
 S-1-2-33-12345-67890-12345-678 | accounts-role| 10          | accounts-group
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(2 rows)
postgres=> select pgadmap_reset_mapping('S-1-2-33-12345-67890-12345-678', 'accounts-
role', 10);
```

```
pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+---------------
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(1 row)
postgres=> select pgadmap_reset_mapping();

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+--------------
 (0 rows)
```

Migrazione dei dati su Amazon Aurora con compatibilità Postgre SQL

Sono disponibili diverse opzioni per migrare i dati dal database esistente a un cluster DB Amazon Aurora SQL Postgre -Compatible Edition. Le opzioni di migrazione dipendono anche dal database da cui esegui la migrazione e dalle dimensioni dei dati sottoposti a migrazione. Le opzioni sono:

Migrazione di un'istanza DB per Postgre tramite un'istantanea RDS SQL

È possibile migrare i dati direttamente da uno snapshot RDS di Postgre SQL DB a un cluster Aurora Postgre DB. SQL

Migrazione di un'istanza database RDS for PostgreSQL tramite una replica di lettura Aurora

È inoltre possibile migrare da un'istanza SQL DB RDS for Postgre creando una replica di SQL lettura Aurora Postgre di un'istanza DB for Postgre. RDS SQL Quando il ritardo della replica tra l'istanza del SQL database RDS for Postgre e la replica di SQL lettura di Aurora Postgre è pari a zero, è possibile interrompere la replica. A questo punto, puoi rendere la replica di lettura Aurora un cluster Aurora SQL Postgre DB autonomo per la lettura e la scrittura.

[Importazione di dati da Amazon S3 in Aurora PostgreSQL]

Puoi migrare i dati importandoli da Amazon S3 in una tabella appartenente a un cluster Aurora Postgre DB. SQL

Migrazione da un database non compatibile con Postgre SQL

È possibile utilizzare… AWS Database Migration Service (AWS DMS) per migrare dati da un database che non è compatibile con Postgre. SQL Per ulteriori informazioni su AWS DMS, vedi [Cos'è AWS Servizio di migrazione del Database?] nel AWS Database Migration Service Guida per l'utente.

> **Note**
>
> L'abilitazione dell'autenticazione Kerberos non è attualmente supportata sul cluster Aurora Postgre SQL DB durante la migrazione da per Postgre. RDS SQL È possibile abilitare l'autenticazione Kerberos solo su un cluster SQL Aurora Postgre DB autonomo.

Per un elenco di Regioni AWS dove è disponibile Aurora, consulta [Amazon Aurora] nel Riferimenti generali di AWS.

> **Important**
>
> Se prevedi di migrare un'istanza RDS di Postgre SQL DB su un SQL cluster Aurora Postgre DB nelle prossime future, ti consigliamo vivamente di disattivare gli aggiornamenti automatici delle versioni secondarie per l'istanza DB all'inizio della fase di pianificazione della migrazione. La migrazione ad Aurora Postgre SQL potrebbe subire ritardi se la versione RDS per Postgre SQL non è ancora supportata da Aurora Postgre. SQL
> Per informazioni sulle versioni di Aurora Postgre, consulta SQL Versioni [del motore per Amazon Aurora] Postgre. SQL

Migrazione di un'istantanea di un'istanza di RDS For Postgre SQL DB a un cluster Aurora Postgre DB SQL

Per creare un cluster Aurora Postgre SQL DB, è possibile migrare uno snapshot DB di un'istanza di For Postgre DB. RDS SQL Il nuovo cluster Aurora Postgre SQL DB è popolato con i dati dell'istanza

DB originale RDS per Postgre. SQL Per informazioni sulla creazione di uno snapshot DB, consulta Creazione di uno snapshot DB.

In alcuni casi, lo snapshot del DB potrebbe non essere presente in Regione AWS dove vuoi collocare i tuoi dati. In tal caso, usa la RDS console Amazon per copiare lo snapshot del DB su quella Regione AWS. Per informazioni sulla copia di un'istantanea del DB, vedere Copia di un'istantanea del DB.

È possibile eseguire RDS la migrazione per ottenere SQL istantanee Postgre compatibili con le versioni di Aurora Postgre disponibili nei seguenti paesi SQL Regione AWS. Ad esempio, è possibile migrare un'istantanea da un'istanza SQL DB RDS per Postgre 11.1 alla SQL versione 11.4, 11.7, 11.8 o 11.9 di Aurora Postgre nella regione Stati Uniti occidentali (California settentrionale). È possibile migrare lo SQL snapshot RDS di Postgre 10.11 ad Aurora Postgre 10.11, 10.12, 10.13 e 10.14SQL. In altre parole, lo SQL snapshot RDS for Postgre deve utilizzare la stessa versione secondaria o inferiore di Aurora Postgre. SQL

Puoi anche scegliere che il tuo nuovo cluster Aurora Postgre SQL DB venga crittografato a riposo utilizzando un AWS KMS key. Questa opzione è disponibile solo per le istantanee DB non crittografate.

Per migrare uno RDS snapshot di Postgre SQL DB a un cluster Aurora SQL Postgre DB, puoi usare il AWS Management Console, il AWS CLI, o il RDSAPI. Quando si utilizza il AWS Management Console, la console esegue le azioni necessarie per creare sia il cluster DB che l'istanza primaria.

Console

Per migrare uno snapshot di Postgree SQL DB utilizzando la console RDS

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. SelezionareSnapshots (Snapshot).
3. Nella pagina Istantanee, scegli lo snapshot RDS per Postgre SQL che desideri migrare in un cluster Aurora Postgre DB. SQL
4. Per Actions (Operazioni) scegliere Migrate snapshot (Migra snapshot).
5. Impostare i valori seguenti nella pagina Migrate database (Migra database):

    - DB engine version (Versione del motore database): scegli la versione del motore database che desideri utilizzare per la nuova istanza migrata.
    - Identificatore dell'istanza DB: inserisci un nome per il cluster DB che sia univoco per il tuo account nel Regione AWS che hai scelto. Questo identificatore viene utilizzato negli indirizzi

degli endpoint per le istanze nel cluster di database. Potresti scegliere di aggiungere un po' di intelligenza al nome, ad esempio includere il Regione AWS e il motore DB che hai scelto, ad esempio**`aurora-cluster1`**.

L'identificatore istanze database presenta i seguenti vincoli:

- Deve contenere da 1 –a 63 caratteri alfanumerici o trattini.
- Il primo carattere deve essere una lettera.
- Non può terminare con un trattino o contenere due trattini consecutivi.
- Deve essere unico per tutte le istanze DB per AWS account, per Regione AWS.

- Classe di istanza database: scegli una classe dell'istanza database che abbia lo spazio di archiviazione e la capacità richiesti per il database, ad esempio `db.r6g.large`. I volumi dei cluster Aurora aumentano automaticamente quando aumenta la quantità di dati nel database. È quindi sufficiente scegliere una classe di istanza database che soddisfi i requisiti di storage correnti. Per ulteriori informazioni, consulta Panoramica dell'archiviazione di Amazon Aurora.
- Cloud privato virtuale (VPC): se ne hai uno esistenteVPC, puoi utilizzarlo VPC con il tuo cluster Aurora Postgre SQL DB scegliendo il tuo VPC identificatore, ad esempio. `vpc-a464d1c1` Per informazioni sulla creazione di un, consulta. VPC Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4

  Altrimenti, puoi scegliere di fare in modo che Amazon ne RDS VPC crei uno per te scegliendo Crea nuovo VPC.
- Gruppo di sottoreti DB: se disponi di un gruppo di sottoreti esistente, puoi utilizzare quel gruppo di sottoreti con il tuo cluster Aurora Postgre SQL DB scegliendo, ad esempio, l'identificatore del gruppo di sottorete. `gs-subnet-group1`
- Accesso pubblico: scegli No per specificare che le istanze del tuo cluster DB sono accessibili solo da risorse interne al tuo. VPC Scegliere Yes (Si) per specificare che le istanze nel cluster DB sono accessibili dalle risorse nella rete pubblica.

> **Note**
>
> Il cluster DB di produzione potrebbe non trovarsi in una sottorete pubblica, poiché solo i server applicativi richiedono l'accesso al cluster DB. Se il cluster di database non deve necessariamente trovarsi in una sottorete pubblica, impostare Public access (Accesso pubblico) su No.

- VPCgruppo di sicurezza: scegli un gruppo VPC di sicurezza per consentire l'accesso al tuo database.
- Zona di disponibilità: scegli la zona di disponibilità per ospitare l'istanza principale per il tuo cluster Aurora SQL Postgre DB. Per fare in modo che Amazon RDS scelga una zona di disponibilità per te, scegli Nessuna preferenza.
- Porta del database: immettere la porta predefinita da utilizzare per la connessione alle istanze nel cluster Aurora SQL Postgre DB. Il valore predefinito è 5432.

  > **Note**
  >
  > Potresti trovarti dietro un firewall aziendale che non consente l'accesso a porte predefinite come la porta predefinita di SQL Postgre, 5432. In questo caso, fornire un valore di porta permesso dal firewall aziendale. Ricorda quel valore di porta in seguito, quando ti connetti al cluster Aurora SQL Postgre DB.

- Abilita crittografia: scegli Abilita crittografia per crittografare il tuo nuovo cluster Aurora Postgre SQL DB a riposo. Scegliete anche una chiave come KMS AWS KMS key value.
- Aggiornamento automatico della versione secondaria: scegli Abilita aggiornamento automatico della versione secondaria per consentire al cluster Aurora Postgre SQL DB di ricevere automaticamente gli aggiornamenti di versione minori del motore Postgre SQL DB quando diventano disponibili.

  L'opzione Aggiornamento automatico della versione secondaria si applica solo agli aggiornamenti alle versioni SQL secondarie del motore Postgre per il cluster Aurora Postgre DB. SQL Non riguarda le patch normali applicate per mantenere la stabilità del sistema.

6. Selezionare Migrate (Migra) per effettuare la migrazione della snapshot DB.
7. Scegliere Databases (Database) per visualizzare il nuovo cluster di database. Scegliere il nuovo cluster di database per monitorare l'avanzamento della migrazione. Una volta completata la migrazione, lo stato del cluster è Disponibile. Nella scheda Connectivity & security (Connettività e sicurezza) è possibile trovare l'endpoint del cluster da utilizzare per la connessione all'istanza di scrittura principale del cluster di database. Per ulteriori informazioni sulla connessione a un cluster Aurora Postgre SQL DB, vedere. Connessione a un cluster database Amazon Aurora

AWS CLI

Utilizzo di AWS CLI migrare uno snapshot di un SQL database RDS per Postgre su Aurora Postgre richiede due operazioni distinte SQL AWS CLI comandi. Innanzitutto, si utilizza il `restore-db-cluster-from-snapshot` AWS CLI comando crea un nuovo cluster Aurora SQL Postgre DB. Quindi viene utilizzato il comando `create-db-instance` per creare l'istanza database primaria nel nuovo cluster per completare la migrazione. La procedura seguente crea un cluster Aurora Postgre SQL DB con istanza DB primaria che ha la stessa configurazione dell'istanza DB utilizzata per creare lo snapshot.

Per migrare uno snapshot di An RDS for Postgre SQL DB a un cluster Aurora Postgre DB SQL

1. Usa il describe-db-snapshotscomando per ottenere informazioni sullo snapshot DB che desideri migrare. È possibile specificare il parametro `--db-instance-identifier` o il parametro `--db-snapshot-identifier` nel comando. Se non si specifica uno di questi parametri, si ottengono tutti gli snapshot.

   ```
   aws rds describe-db-snapshots --db-instance-identifier <your-db-instance-name>
   ```

2. Il comando restituisce tutti i dettagli di configurazione per gli snapshot creati dall'istanza database specificata. Nell'output, trova lo snapshot che desideri migrare e individua il relativo Amazon Resource Name ()ARN. Per ulteriori informazioni su Amazon RDSARNs, consulta Amazon Relational Database Service (RDSAmazon). An ARN ha un aspetto simile all'output seguente.

   ```
   “DBSnapshotArn": "arn:aws:rds:aws-region:111122223333:snapshot:<snapshot_name>"
   ```

   Nell'output puoi anche trovare i dettagli di configurazione RDS per l'istanza SQL DB di Postgre, come la versione del motore, lo spazio di archiviazione allocato, se l'istanza DB è crittografata o meno e così via.

3. Utilizzate il comando restore-db-cluster-from-snapshot per avviare la migrazione. Specifica i seguenti parametri:

   - `--db-cluster-identifier`— Il nome che si desidera assegnare al cluster Aurora SQL Postgre DB. Questo cluster di database di Aurora è la destinazione per la migrazione dello snapshot DB.
   - `--snapshot-identifier`— L'Amazon Resource Name (ARN) dello snapshot DB da migrare.

- `--engine`: specificare `aurora-postgresql` per il cluster motore di database Aurora.
- `--kms-key-id`— Questo parametro opzionale consente di creare un cluster Aurora Postgre SQL DB crittografato da un'istantanea DB non crittografata. Consente inoltre di scegliere una chiave di crittografia diversa per il cluster di database rispetto alla chiave utilizzata per lo snapshot DB.

> **Note**
>
> Non è possibile creare un cluster Aurora SQL Postgre DB non crittografato da un'istantanea DB crittografata.

Senza il `--kms-key-id` parametro specificato come mostrato di seguito, -snapshot restore-db-cluster-from AWS CLI il comando crea un cluster Aurora Postgre SQL DB vuoto che è crittografato utilizzando la stessa chiave dello snapshot DB o non è crittografato se lo snapshot DB di origine non è crittografato.

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier cluster-name \
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name \
    --engine aurora-postgresql
```

In Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier new_cluster ^
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name ^
    --engine aurora-postgresql
```

4. Il comando restituisce dettagli sul cluster Aurora Postgre SQL DB che viene creato per la migrazione. È possibile controllare lo stato del cluster Aurora Postgre SQL DB utilizzando il describe-db-clusters AWS CLI comando.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-name
```

5. Quando il cluster DB diventa «disponibile», usi il create-db-instancecomando per popolare il cluster DB Aurora SQL Postgre con l'istanza DB basata sullo snapshot di Amazon DB. RDS Specifica i seguenti parametri:

   - `--db-cluster-identifier`— Il nome del nuovo cluster Aurora Postgre SQL DB creato nel passaggio precedente.
   - `--db-instance-identifier`: il nome da assegnare all'istanza database. Questa istanza diventa il nodo principale del cluster Aurora SQL Postgre DB.
   - `----db-instance-class` : specificare la classe di istanza database da utilizzare. Scegli tra le classi di istanze DB supportate dalla SQL versione di Aurora Postgre verso cui stai migrando. Per ulteriori informazioni, consulta Tipi di classi di istanza database e Motori DB supportati per classi di istanza database.
   - `--engine`: specificare `aurora-postgresql` per l'istanza database.

   Puoi anche creare l'istanza DB con una configurazione diversa rispetto allo snapshot del DB di origine, inserendo le opzioni appropriate nel `create-db-instance` AWS CLI comando. Per ulteriori informazioni, vedere il create-db-instancecomando.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-instance \
       --db-cluster-identifier cluster-name \
       --db-instance-identifier --db-instance-class db.instance.class \
       --engine aurora-postgresql
   ```

   In Windows:

   ```
   aws rds create-db-instance ^
       --db-cluster-identifier cluster-name ^
       --db-instance-identifier --db-instance-class db.instance.class ^
       --engine aurora-postgresql
   ```

Al termine del processo di migrazione, il cluster Aurora SQL Postgre dispone di un'istanza DB primaria popolata.

Migrazione di dati da un'istanza database RDS for PostgreSQL a un cluster di database Aurora PostgreSQL tramite una replica di lettura Aurora

Puoi utilizzare un'istanza database RDS for PostgreSQL come base per un nuovo cluster di database Aurora PostgreSQL utilizzando una replica di lettura Aurora per il processo di migrazione. L'opzione di replica di lettura Aurora è disponibile solo per la migrazione all'interno dello stesso account Regione AWS ed è disponibile solo se la regione offre una versione compatibile di Aurora PostgreSQL per l'istanza DB RDS per PostgreSQL. Per compatibile si intende che la versione di Aurora PostgreSQL è la stessa utilizzata da RDS for PostgreSQL o che è una versione secondaria successiva della stessa famiglia di versioni principali.

Ad esempio, per utilizzare questa tecnica per migrare un'istanza database RDS for PostgreSQL 11.14, la regione deve offrire Aurora PostgreSQL versione 11.14 o una versione secondaria superiore nella famiglia di PostgreSQL versione 11.

Argomenti

- Panoramica della migrazione dei dati tramite una replica di lettura Aurora
- Preparazione alla migrazione dei dati tramite una replica di lettura Aurora
- Creazione di una replica di lettura Aurora
- Promozione di una replica di lettura Aurora

Panoramica della migrazione dei dati tramite una replica di lettura Aurora

La migrazione da un'istanza database RDS for PostgreSQL a un cluster di database Aurora PostgreSQL richiede una procedura in più fasi. In primo luogo, si crea una replica di lettura Aurora dell'istanza database RDS for PostgreSQL di origine. Questo avvia un processo di replica dall'istanza database RDS for PostgreSQL a un cluster di database per finalità speciali noto come cluster di replica. Il cluster di replica è costituito esclusivamente da una replica di lettura Aurora (un'istanza di lettura).

> **Note**
>
> Per completare la migrazione possono essere necessarie diverse ore per ogni terabyte di dati.

Promozione di una replica Aurora PostgreSQL

Dopo aver creato un cluster Aurora PostgreSQL DB, segui questi passaggi per promuovere la replica Aurora:

1. Arresta tutto il carico di lavoro di scrittura del database sull'istanza DB RDS for PostgreSQL di origine.
2. Ottieni la corrente `WAL LSN` dall'istanza DB RDS for PostgreSQL di origine:

   ```
   SELECT pg_current_wal_lsn();
   pg_current_wal_lsn
   --------------------
   0/F0000318
   (1 row)
   ```

3. Sul cluster Aurora PostgreSQL Replica, verifica che il LSN riprodotto sia maggiore del LSN del passaggio 2:

   ```
   SELECT pg_last_wal_replay_lsn();
   pg_last_wal_replay_lsn
   ------------------------
   0/F0000400
   (1 row)
   ```

   In alternativa, puoi utilizzare la seguente query sull'istanza DB RDS for PostgreSQL di origine:

   ```
   SELECT restart_lsn FROM pg_replication_slots;
   ```

4. Promuovi il cluster Aurora PostgreSQL Replica.

   Quando la replica si interrompe, il cluster di replica viene promosso a cluster Aurora PostgreSQL DB autonomo e il reader viene promosso a istanza di scrittura per il cluster. A questo punto, puoi aggiungere istanze al cluster Aurora PostgreSQL DB per dimensionarlo in base al tuo caso d'uso. Se non è più necessaria l'istanza DB RDS for PostgreSQL originale, è possibile eliminarla.

Non è possibile creare una replica di lettura Aurora se l'istanza database RDS for PostgreSQL ha già una replica di lettura Aurora o se ha una replica di lettura tra regioni.

Preparazione alla migrazione dei dati tramite una replica di lettura Aurora

Durante il processo di migrazione tramite la replica di lettura Aurora, gli aggiornamenti apportati all'istanza database RDS for PostgreSQL di origine vengono replicati in modo asincrono nella replica di lettura Aurora del cluster di replica. Il processo utilizza la funzionalità di replica dello streaming nativa di PostgreSQL che memorizza i segmenti WAL (write ahead log) sull'istanza di origine. Prima di iniziare questo processo di migrazione, assicurati che l'istanza abbia una capacità di archiviazione sufficiente controllando i valori per i parametri elencati nella tabella.

Parametro	Descrizione
`FreeStorageSpace`	Spazio di storage disponibile. Unità: byte
`OldestReplicationSlotLag`	Entità del ritardo per i dati WAL nella replica con il maggiore ritardo. Unità: megabyte
`RDSToAuroraPostgreSQLReplicaLag`	Periodo di tempo, in secondi, di ritardo di un cluster DB Aurora PostgreSQL rispetto all'istan za database RDS di origine.
`TransactionLogsDiskUsage`	Spazio su disco utilizzato dai log delle transazio ni. Unità: megabyte

Per ulteriori informazioni sul monitoraggio dell'istanza, consulta Monitoraggio nella Guida per l'utente di Amazon RDS.

Creazione di una replica di lettura Aurora

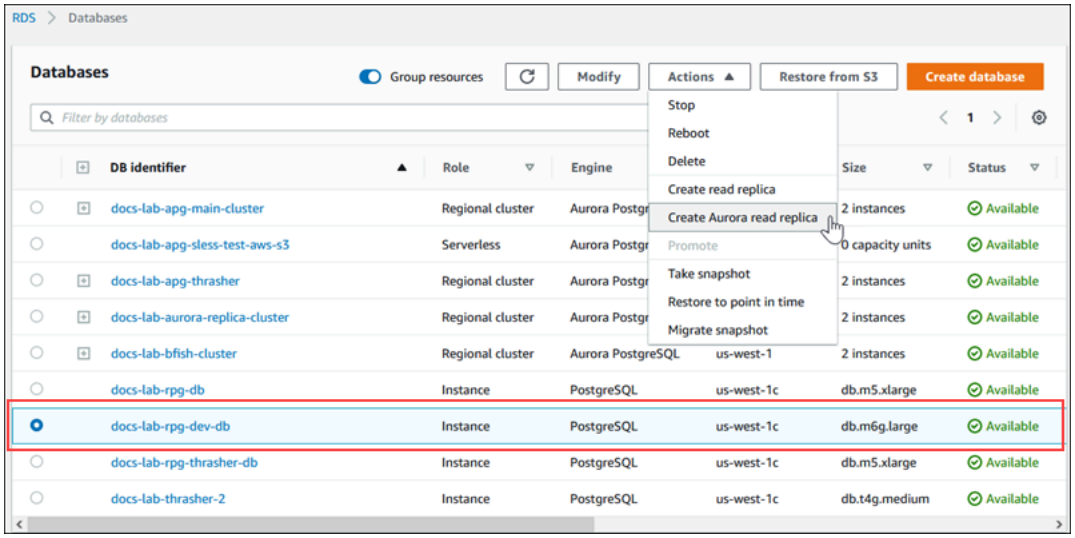
È possibile creare una replica di lettura Aurora per un'istanza DB RDS per PostgreSQL utilizzando o il. AWS Management Console AWS CLI L'opzione per creare una replica di lettura di Aurora

utilizzando AWS Management Console è disponibile solo se Regione AWS offre una versione Aurora PostgreSQL compatibile. Ovvero, è disponibile solo se è presente una versione di Aurora PostgreSQL che è la stessa della versione RDS for PostgreSQL o una versione secondaria superiore nella stessa famiglia di versioni principali.

Console

Per creare una replica di lettura Aurora da un'istanza database PostgreSQL di origine

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere l'istanza database RDS for PostgreSQL da usare come origine per la replica di lettura Aurora. Per Actions (Operazioni), scegli Create Aurora read replica (Crea replica di lettura Aurora). Se questa scelta non viene visualizzata, significa che una versione compatibile di Aurora PostgreSQL non è disponibile nella regione.



4. Nella pagina delle impostazioni per la creazione della replica di lettura Aurora è possibile configurare le proprietà per il cluster di database Aurora PostgreSQL come illustrato nella tabella riportata di seguito. Il cluster di database di replica viene creato da uno snapshot dell'istanza database di origine utilizzando lo stesso nome utente e password “master” dell'origine, quindi non è possibile modificarli in questo momento.

Opzione	Descrizione
DB instance class (Classe istanza database)	Scegliere una classe di istanza database che soddisfi i requisiti di elaborazione e di memoria per l'istanza primaria nel cluster di database. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.
Multi-AZ deployment (Implementazione Multi-AZ)	Non disponibile durante la migrazione
DB instance identifier (Identificatore istanze DB)	Inserire il nome da assegnare all'istanza database. Questo identificatore viene utilizzato nell'indirizzo dell'endpoint per l'istanza primaria del nuovo cluster di database. L'identificatore istanze database presenta i seguenti vincoli: • Deve contenere da 1 –a 63 caratteri alfanumerici o trattini. • Il primo carattere deve essere una lettera. • Non può terminare con un trattino o contenere due trattini consecutivi. • Deve essere unica per tutte le istanze DB di ogni AWS account, per ciascuno. Regione AWS
Virtual Private Cloud (VPC)	Scegliere il VPC che ospita il cluster DB. Scegliere Create a new VPC (Crea un nuovo VPC) per fare in modo che Amazon RDS crei un VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Opzione	Descrizione
DB subnet group (Gruppo di sottoreti DB)	Scegliere il gruppo di sottoreti di database da utilizzare per il cluster DB. Scegliere Create new DB subnet group (Crea nuovo gruppo di sottoreti DB) per fare in modo che Amazon RDS crei automaticamente un gruppo di sottoreti di database. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.
Public accessibility (Accesso pubblico)	Scegliere Sì per assegnare al cluster di database un indirizzo IP pubblico, altrimenti, selezionare No. Le istanze nel cluster di database possono essere una combinazione di istanze database pubbliche e private. Per ulteriori informazioni su come non consentire l'accesso pubblico per le istanze, consulta Nascondere cluster database in un VPC da Internet.
Availability zone (Zona di disponibilità)	Stabilire se si desidera specificare una zona di disponibilità particolare. Per ulteriori informazioni sulle zone di disponibilità, consultare Regioni e zone di disponibilità.
Gruppi di sicurezza VPC	Scegliere uno o più gruppi di sicurezza VPC per proteggere l'accesso di rete al cluster DB. Scegliere Create a new VPC security group (Crea nuovo gruppo di sicurezza VPC) per fare in modo che Amazon RDS crei un gruppo di sicurezza VPC. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.
Database port (Porta del database)	Specifica la porta per applicazioni e utilità da utilizzare per accedere al database. I cluster DB PostgreSQL Aurora utilizzano per impostazione predefinita la porta predefinita PostgreSQL, 5432. I firewall presso alcune aziende bloccano le connessioni a questa porta. Se il firewall dell'azienda blocca la porta predefinita, scegliere un'altra porta per il nuovo cluster di database.

Opzione	Descrizione
DB parameter group (Gruppo di parametri database)	Scegliere un gruppo di parametri del database per il cluster di database di Aurora PostgreSQL. Puoi utilizzare il gruppo parametri del database predefinito di Aurora oppure puoi creare il tuo gruppo parametri del database personalizzato. Per ulteriori informazioni sui gruppi di parametri database, consulta .
DB cluster parameter group (Gruppo di parametri del cluster di database)	Scegliere un gruppo di parametri del cluster di database per il cluster di database di Aurora PostgreSQL. Puoi utilizzare il gruppo di parametri del cluster di database predefinito di Aurora oppure puoi creare il tuo gruppo di parametri. Per ulteriori informazi oni sui gruppi di parametri del cluster di database, consulta .
Encryption (Crittografia)	Seleziona Enable Encryption (Abilita crittografia) affinché il nuovo cluster di database Aurora sia crittografato mentre è inattivo. Se si sceglie Enable encryption (Abilita crittografia), scegliere una chiave KMS come valore AWS KMS key.
Priority (Priorità)	Scegliere una priorità di failover per il cluster di database. Se non specifichi alcun valore, l'imposta zione predefinita è tier-1. Questa priorità determina l'ordine di promozione delle repliche di Aurora durante il recupero da un errore dell'istanza principale. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.
Backup retention period (Periodo di retention dei backup)	Scegliere l'intervallo di tempo, da 1 –a 35 giorni, nel quale Aurora conserverà le copie di backup del database. Le copie di Backup possono essere utilizzat e per point-in-time i ripristini (PITR) del database fino al secondo.

Opzione	Descrizione
Enhanced Monitoring (Monitoraggio avanzato)	Scegliere Enable enhanced monitoring (Abilita monitoraggio avanzato) per abilitare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster DB. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
Monitoring Role (Ruolo monitoraggio)	Disponibile solo se hai scelto Enable enhanced monitoring (Abilita monitoraggio avanzato). Il ruolo AWS Identity and Access Management (IAM) da utilizzare per il monitoraggio avanzato. Per ulteriori informazioni, consulta Configurare e abilitare il monitoraggio avanzato.
Granularity (Granularità)	Disponibile solo se hai scelto Enable enhanced monitoring (Abilita monitoraggio avanzato). Impostare l'intervallo, in secondi, tra le operazioni di raccolta dei parametri per il cluster DB.
Auto minor version upgrade (Aggiornamento automatico della versione secondaria)	Scegliere Yes (Sì) per abilitare il cluster DB Aurora PostgreSQL in modo che riceva automaticamente gli aggiornamenti delle versioni minori del motore di database PostgreSQL non appena diventano disponibili. L'opzione Auto minor version upgrade (Aggiornamento automatico della versione secondaria) si applica solo agli aggiornamenti alle versioni secondarie del motore PostgreSQL per il cluster di database Aurora PostgreSQL. Non riguarda le patch normali applicate per mantenere la stabilità del sistema.
Maintenance window (Finestra di manutenzione)	Scegliere l'intervallo temporale settimanale durante il quale può avvenire la manutenzione dei sistemi.

5. Scegli Create read replica (Crea replica di lettura).

AWS CLI

Per creare una replica di lettura Aurora da un'istanza DB RDS for PostgreSQL di origine utilizzando il AWS CLI, è innanzitutto necessario utilizzare il comando CLI create-db-clusterper creare un cluster Aurora DB vuoto. Una volta creato il cluster di database, dovrai creare l'istanza primaria per il cluster di database utilizzando il comando create-db-instance. L'istanza primaria è la prima istanza creata in un cluster di database Aurora. In questo caso, viene creata inizialmente come replica di lettura Aurora dell'istanza database RDS for PostgreSQL. Al termine del processo, la migrazione dell'istanza database RDS for PostgreSQL in un cluster di database Aurora PostgreSQL è completa.

Non è necessario specificare l'account utente principale (in genere, `postgres`), la password o il nome del database. La replica di lettura Aurora li ottiene automaticamente dall'istanza DB RDS di origine per PostgreSQL che identifichi quando richiami i comandi. AWS CLI

È necessario specificare la versione del motore da utilizzare per il cluster di database Aurora PostgreSQL e l'istanza database. La versione specificata deve corrispondere all'istanza database RDS for PostgreSQL di origine. Se l'istanza database RDS for PostgreSQL di origine è crittografata, è necessario specificare anche la crittografia per l'istanza primaria del cluster di database Aurora PostgreSQL. La migrazione di un'istanza crittografata in un cluster di database Aurora non crittografato non è supportata.

Gli esempi seguenti mostrano come creare un cluster di database Aurora PostgreSQL denominato `my-new-aurora-cluster`, che utilizza un'istanza database RDS di origine non crittografata. In primo luogo, crei il cluster di database Aurora PostgreSQL chiamando il comando CLI `create-db-cluster`. L'esempio mostra come utilizzare il parametro facoltativo `--storage-encrypted` per specificare che il cluster di database deve essere crittografato. Poiché il database di origine non è crittografato, è necessario specificare una chiave utilizzando `--kms-key-id`. Per ulteriori informazioni sui parametri obbligatori e facoltativi, vedi l'elenco dell'esempio seguente.

Per, o: Linux macOS Unix

```
aws rds create-db-cluster  \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-subnet-group-name my-db-subnet
    --vpc-security-group-ids sg-11111111
    --engine aurora-postgresql \
    --engine-version same-as-your-rds-instance-version \
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db \
    --storage-encrypted \
```

```
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Per Windows:

```
aws rds create-db-cluster  ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-subnet-group-name my-db-subnet ^
    --vpc-security-group-ids sg-11111111 ^
    --engine aurora-postgresql ^
    --engine-version same-as-your-rds-instance-version ^
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db ^
    --storage-encrypted ^
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Nell'elenco seguente è possibile trovare ulteriori informazioni su alcune delle opzioni mostrate nell'esempio. Se non diversamente specificato, questi parametri sono obbligatori.

- `--db-cluster-identifier`. Assegna un nome al nuovo cluster di database Aurora PostgreSQL.
- `--db-subnet-group-name`. Crea il cluster di database Aurora PostgreSQL nella stessa sottorete database dell'istanza database di origine.
- `--vpc-security-group-ids`. Specifica il gruppo di sicurezza per il cluster di database Aurora PostgreSQL.
- `--engine-version`. Specifica la versione da utilizzare per il cluster di database Aurora PostgreSQL. Deve essere la stessa versione o una versione secondaria superiore rispetto alla versione utilizzata dall'istanza database RDS for PostgreSQL di origine.
- `--replication-source-identifier`. Identifica l'istanza database RDS for PostgreSQL utilizzando il suo Amazon Resource Name (ARN). Per ulteriori informazioni su Amazon RDS ARNs, consulta Amazon Relational Database Service (Amazon RDS) nel. Riferimenti generali di AWSdel tuo cluster DB.
- `--storage-encrypted` : Opzionale. Utilizzalo solo quando devi specificare la crittografia, come spiegato di seguito:
  - Utilizza questo parametro quando l'istanza database di origine ha un'archiviazione crittografata. Se non utilizzi questo parametro con un'istanza database di origine con archiviazione

crittografata, la chiamata al `create-db-cluster` restituisce un errore. Se desideri utilizzare una chiave diversa per il cluster di database Aurora PostgreSQL rispetto alla chiave utilizzata dall'istanza database di origine, è necessario specificare anche la `--kms-key-id`.

- Utilizzala se l'archiviazione dell'istanza database di origine non è crittografata, ma desideri che il cluster di database Aurora PostgreSQL lo sia. In tal caso, devi anche identificare la chiave di crittografia necessaria utilizzando il parametro `--kms-key-id`.

- `--kms-key-id` : Opzionale. Permette di specificare la chiave da utilizzare per la crittografia dell'archiviazione (`--storage-encrypted`), utilizzando l'ARN, l'ID, l'alias ARN della chiave o il relativo nome alias. Questo parametro è necessario solo nelle situazioni seguenti:
  - Per scegliere una chiave diversa per il cluster di database Aurora PostgreSQL rispetto a quella utilizzata dall'istanza database di origine.
  - Per creare un cluster crittografato da un'origine non crittografata. In questo caso, è necessario specificare la chiave che Aurora PostgreSQL deve utilizzare per la crittografia.

Una volta creato il cluster di database Aurora PostgreSQL, dovrai creare l'istanza principale utilizzando il comando CLI `create-db-instance`, come mostrato di seguito:

PerLinux, o: macOS Unix

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-instance-class db.x2g.16xlarge \
    --db-instance-identifier rpg-for-migration \
    --engine aurora-postgresql
```

Per Windows:

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-instance-class db.x2g.16xlarge ^
    --db-instance-identifier rpg-for-migration ^
    --engine aurora-postgresql
```

L'elenco seguente fornisce ulteriori informazioni su alcune delle opzioni mostrate nell'esempio.

- `--db-cluster-identifier`: specifica il nome del nuovo cluster di database Aurora PostgreSQL creato nel passaggio precedente con il comando `create-db-instance`.

- `--db-instance-class`: nome della classe dell'istanza database da utilizzare per l'istanza primaria, come `db.r4.xlarge`, `db.t4g.medium`, `db.x2g.16xlarge`, ecc. Per un elenco delle classi di istanza database disponibili, vedi Tipi di classi di istanza database.
- `--db-instance-identifier`: specifica il nome da assegnare all'istanza primaria.
- `--engine aurora-postgresql`: specifica il `aurora-postgresql` per il motore.

API RDS

Per creare una replica di lettura Aurora da un'istanza database RDS for PostgreSQL di origine, per prima cosa crea un nuovo cluster di database Aurora utilizzando l'operazione API RDS `CreateDBCluster`, che verrà utilizzato per la replica. Quando il cluster di database Aurora PostgreSQL è disponibile, utilizza `CreateDBInstance`per creare l'istanza primaria per il cluster di database Aurora.

Non è necessario specificare l'account utente principale (in genere, `postgres`), la password o il nome del database. La replica di lettura Aurora li ottiene automaticamente dall'istanza database RDS for PostgreSQL di origine specificata con `ReplicationSourceIdentifier`.

È necessario specificare la versione del motore da utilizzare per il cluster di database Aurora PostgreSQL e l'istanza database. La versione specificata deve corrispondere all'istanza database RDS for PostgreSQL di origine. Se l'istanza database RDS for PostgreSQL di origine è crittografata, è necessario specificare anche la crittografia per l'istanza primaria del cluster di database Aurora PostgreSQL. La migrazione di un'istanza crittografata in un cluster di database Aurora non crittografato non è supportata.

Per creare il cluster di database Aurora per la replica di lettura Aurora, utilizza l'operazione API RDS `CreateDBCluster` con i seguenti parametri:

- `DBClusterIdentifier`: nome del cluster di database da creare.
- `DBSubnetGroupName`: nome del gruppo di sottoreti database da associare al cluster di database.
- `Engine=aurora-postgresql`: nome del motore da utilizzare.
- `ReplicationSourceIdentifier`: l'Amazon Resource Name (ARN) per l'istanza database PostgreSQL di origine. Per ulteriori informazioni su Amazon RDS ARNs, consulta Amazon Relational Database Service (Amazon RDS) nel.Riferimenti generali di Amazon Web Services Se `ReplicationSourceIdentifier` riconosce la fonte come crittografata, Amazon RDS utilizza la tua chiave KMS di default, a meno che tu non specifichi una chiave diversa utilizzando l'opzione `KmsKeyId`.

- `VpcSecurityGroupIds`— L'elenco dei gruppi di sicurezza Amazon EC2 VPC da associare a questo cluster DB.
- `StorageEncrypted`: indica che il cluster di database è crittografato. Quando utilizzi questo parametro senza specificare anche il `ReplicationSourceIdentifier`, Amazon RDS utilizza la chiave KMS di default.
- `KmsKeyId`: la chiave per un cluster crittografato. Permette di specificare la chiave da utilizzare per la crittografia dell'archiviazione utilizzando l'ARN, l'ID, l'alias ARN della chiave o il relativo nome alias.

Per ulteriori informazioni, consulta `CreateDBCluster` nella Guida di riferimento delle API di Amazon RDS.

Quando il cluster di database Aurora è disponibile, crea un'istanza primaria per il cluster di database utilizzando l'operazione API RDS `CreateDBInstance` con i seguenti parametri:

- `DBClusterIdentifier`: nome del cluster di database.
- `DBInstanceClass`: nome della classe dell'istanza database da utilizzare per l'istanza primaria.
- `DBInstanceIdentifier`: nome dell'istanza primaria.
- `Engine=aurora-postgresql`: nome del motore da utilizzare.

Per ulteriori informazioni, consulta `CreateDBInstance` nella Guida di riferimento delle API di Amazon RDS.

Promozione di una replica di lettura Aurora

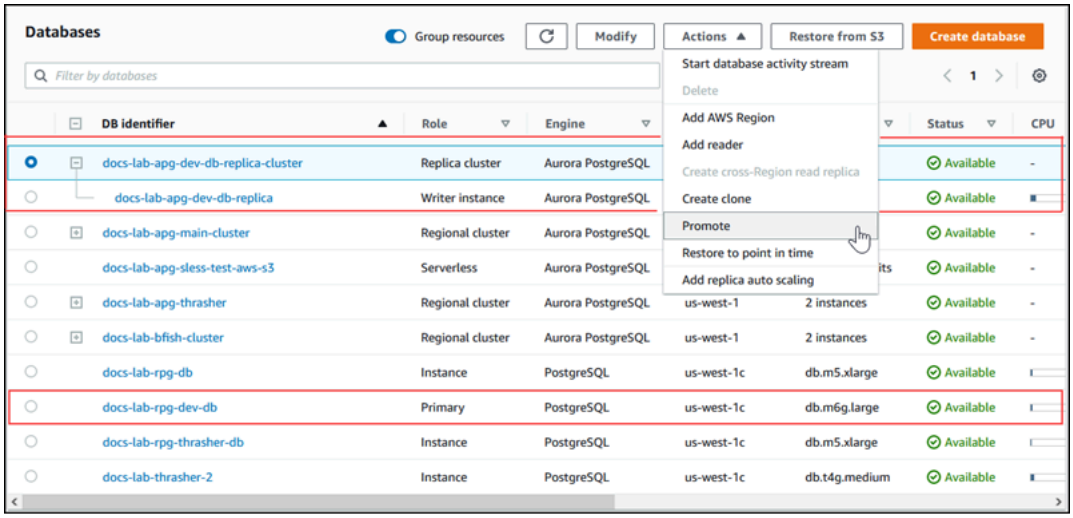
La migrazione ad Aurora PostgreSQL non è completa fino a quando non si promuove il cluster di replica, quindi non eliminare ancora l'istanza database RDS for PostgreSQL di origine.

Prima di promuovere il cluster di replica, assicurati che l'istanza database RDS for PostgreSQL non abbia transazioni in corso o altre attività di scrittura sul database. Quando il ritardo di replica sulla replica di lettura Aurora è zero (0), è possibile promuovere il cluster di replica. Per ulteriori informazioni sul monitoraggio del ritardo di replica, consulta Monitoraggio della replica Aurora PostgreSQL. e Parametri a livello di istanza per Amazon Aurora.

Console

Come promuovere una replica di lettura Aurora a cluster di database Aurora

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegli il cluster di replica.



4. In Actions (Operazioni), seleziona Promote (Promuovi). L'operazione può richiedere alcuni minuti e comportare tempi di inattività.

Al termine del processo, il cluster di replica Aurora è un cluster di database regionale Aurora PostgreSQL, con un'istanza di scrittura contenente i dati dall'istanza database RDS for PostgreSQL.

AWS CLI

Per promuovere una replica di lettura Aurora in un cluster DB autonomo, usa il comando. promote-read-replica-db-cluster AWS CLI

Example

PerLinux, o: macOS Unix

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Per Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

API RDS

Per promuovere una replica di lettura Aurora in un cluster DB autonomo, utilizza l'operazione API RDS. PromoteReadReplicaDBCluster

Dopo aver promosso il cluster di replica, puoi verificare che la promozione sia stata completata controllando il registro eventi, come segue.

Per confermare che il cluster di replica Aurora sia stato promosso

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione seleziona Events (Eventi).
3. Nella pagina Events (Eventi), trova il nome del cluster nell'elenco Source (Origine). Ogni evento ha origine, tipo, ora e messaggio. Puoi vedere tutti gli eventi che si sono verificati nella tua Regione AWS per il tuo account. Una promozione avvenuta con successo genera il seguente messaggio.

   ```
   Promoted Read Replica cluster to a stand-alone database cluster.
   ```

Al completamento della promozione, l'istanza database RDS for PostgreSQL di origine e il cluster di replica Aurora vengono scollegati. Puoi indirizzare le applicazioni client all'endpoint per la replica di lettura Aurora. Per ulteriori informazioni sugli endpoint Aurora, consulta Connessioni endpoint Amazon Aurora. A questo punto, puoi eliminare l'istanza database.

Ottimizzazione delle prestazioni delle query in Aurora PostgreSQL

L'ottimizzazione delle prestazioni delle query è fondamentale perché aiuta i database a funzionare in modo più rapido ed efficiente utilizzando meno risorse, il che si traduce in una migliore esperienza utente e in una riduzione dei costi operativi. Amazon Aurora PostgreSQL offre diverse funzionalità per ottimizzare le prestazioni delle query per i carichi di lavoro PostgreSQL.

Argomenti

- Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora

- Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL
- Miglioramento delle prestazioni delle query utilizzando adaptive join

Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora

Con Letture ottimizzate per Aurora è possibile velocizzare l'elaborazione delle query per Aurora PostgreSQL. Un'istanza database Aurora PostgreSQL che utilizza Letture ottimizzate per Aurora offre una latenza delle query fino a 8 volte migliorata e risparmi sui costi fino al 30% per applicazioni con set di dati di grandi dimensioni, che superano la capacità di memoria di un'istanza database.

Argomenti

- Panoramica di Letture ottimizzate per Aurora in PostgreSQL
- Utilizzo di Letture ottimizzate per Aurora
- Casi d'uso per letture ottimizzate per Aurora
- Monitoraggio delle istanze database che utilizzano Letture ottimizzate per Aurora
- Best practice per Letture ottimizzate per Aurora

Panoramica di Letture ottimizzate per Aurora in PostgreSQL

Aurora Optimized Reads è disponibile per impostazione predefinita quando si crea un cluster DB che utilizza istanze R6gd basate su Graviton e R6id basate su Intel con storage express () in memoria non volatile. NVMe È disponibile a partire dalle versioni di PostgreSQL seguenti:

- 16.1 e tutte le versioni successive
- 15.4 e versioni successive
- 14.9 e versioni successive

Letture ottimizzate per Aurora supporta due funzionalità: cache a più livelli e oggetti temporanei.

Cache a più livelli abilitata per Letture ottimizzate: con una cache a più livelli puoi ottenere una capacità di caching dell'istanza database fino a 5 volte superiore la memoria dell'istanza. Ciò assicura che la cache contenga automaticamente i dati più recenti e coerenti dal punto di vista transazionale, sollevando le applicazioni dall'onere di gestire la valuta dei dati delle soluzioni di caching basate

su set di risultati esterni. Offre una latenza fino a 8 volte migliore per le query che in precedenza recuperavano dati dall'archiviazione di Aurora.

In Aurora, il valore per `shared_buffers` nel gruppo di parametri predefinito è in genere impostato su circa il 75% della memoria disponibile. Tuttavia, per i tipi di istanza r6gd e r6id, Aurora ridurrà `shared_buffers` lo spazio del 4,5% per ospitare i metadati per la cache delle letture ottimizzate.

Oggetti temporanei ottimizzati per la lettura: utilizzando oggetti temporanei, è possibile ottenere un'elaborazione delle query più rapida posizionando i file temporanei generati da PostgreSQL nella memoria locale. NVMe Ciò riduce il traffico verso Elastic Block Storage (EBS) attraverso la rete. Offre una latenza e un throughput fino a 2 volte migliori per le query avanzate che ordinano, uniscono o uniscono grandi volumi di dati che non rientrano nella capacità di memoria disponibile su un'istanza DB.

In un cluster Aurora I/O ottimizzato, Optimized Reads utilizza sia la cache a più livelli che gli oggetti temporanei sullo storage. NVMe Con la funzionalità di cache a più livelli abilitata per Letture ottimizzate, Aurora alloca il doppio della memoria dell'istanza per gli oggetti temporanei, circa il 10% dell'archiviazione per le operazioni interne e l'archiviazione rimanente come cache a più livelli. In un cluster Aurora standard, Letture ottimizzate utilizza solo oggetti temporanei.

I cluster Aurora I/O Optimized consentono di ridimensionare lo spazio allocato per gli oggetti temporanei abilitati alle letture ottimizzate utilizzando il parametro dinamico a livello di istanza. `aurora_temp_space_size` Questa funzionalità di ridimensionamento è disponibile nelle seguenti versioni di PostgreSQL:

- 16.8 e tutte le versioni successive
- 15.12 e versioni successive 15
- 14.17 e versioni successive 14

Con questo parametro, è possibile ridimensionare la capacità da 2 a 6 volte la memoria dell'istanza senza richiedere il riavvio del motore di database. Quando si espande lo spazio temporaneo degli oggetti, la modifica ha effetto immediato, indipendentemente dai carichi di lavoro simultanei. Tuttavia, quando si riduce lo spazio, la regolazione viene completata solo dopo che c'è spazio inutilizzato sufficiente negli oggetti temporanei per soddisfare la nuova richiesta di dimensione. Dopo aver ridimensionato gli oggetti temporanei abilitati per Optimized Reads, la cache a più livelli si adatta automaticamente per utilizzare tutto lo spazio disponibile.

Motore	Configurazione dell'archiviazione del cluster	Oggetti temporanei abilitati per Letture ottimizzate	Cache a più livelli abilitata per Letture ottimizzate	Versioni supportate
Amazon Aurora edizione compatibile con PostgreSQL	Standard	Sì	No	Aurora PostgreSQL versione 16.1 e tutte le versioni successive, 15.4 e successive, versione 14.9 e successive
	Ottimizzato per I/O	Sì	Sì	

> **Note**
>
> Il passaggio tra cluster ottimizzati per IO e standard su una classe di istanze DB NVMe basata comporta un riavvio immediato del motore di database.

In Aurora PostgreSQL, usa il `temp_tablespaces` parametro per configurare lo spazio della tabella in cui vengono archiviati gli oggetti temporanei.

Per verificare se gli oggetti temporanei sono configurati, usa il seguente comando:

```
postgres=> show temp_tablespaces;
temp_tablespaces
---------------------
aurora_temp_tablespace
(1 row)
```

`aurora_temp_tablespace`È un tablespace configurato da Aurora che punta all'archiviazione locale. NVMe Non puoi modificare questo parametro o tornare all'archiviazione Amazon EBS.

Per verificare se la cache per le letture ottimizzate è attiva, usa il seguente comando:

```
postgres=> show shared_preload_libraries;
                shared_preload_libraries
-----------------------------------------------------
rdsutils,pg_stat_statements,aurora_optimized_reads_cache
```

Utilizzo di Letture ottimizzate per Aurora

Quando si esegue il provisioning di un'istanza DB Aurora PostgreSQL con una delle istanze DB NVMe basate, l'istanza DB utilizza automaticamente Aurora Optimized Reads.

Per attivare Letture ottimizzate per Aurora, procedi in uno dei seguenti modi:

- Crea un cluster Aurora PostgreSQL DB utilizzando una delle classi di istanze DB basate. NVMe Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.
- Modifica un cluster Aurora PostgreSQL DB esistente per utilizzare una delle classi di istanze DB basate. NVMe Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Aurora Optimized Reads è disponibile Regioni AWS ovunque siano supportate una o più classi di istanze DB con archiviazione NVMe SSD locale. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.

Per tornare a un'istanza Aurora di lettura non ottimizzata, modifica la classe di istanza DB dell'istanza Aurora con la classe di istanza simile senza spazio di archiviazione NVMe effimero per i carichi di lavoro del database. Ad esempio, se la classe di istanza database corrente è db.r6gd.4xlarge, scegli db.r6g.4xlarge per tornare indietro. Per ulteriori informazioni, consulta la pagina relativa alla modifica di un'istanza database Aurora.

Casi d'uso per letture ottimizzate per Aurora

Cache a più livelli abilitata per Letture ottimizzate

Di seguito sono riportati alcuni casi d'uso in cui è possibile trarre vantaggio dalla funzionalità Letture ottimizzate con cache a più livelli:

- Applicazioni su scala Internet come elaborazione dei pagamenti, fatturazione, e-commerce con prestazioni rigorose. SLAs
- Dashboard per le segnalazioni in tempo reale che eseguono centinaia di point query per la raccolta di metriche e dati.

- Applicazioni di intelligenza artificiale generativa con l'estensione pgvector per la ricerca di vicini esatti o più prossimi tra milioni di incorporamenti vettoriali.

Oggetti temporanei abilitati per Letture ottimizzate

Di seguito sono riportati alcuni casi d'uso in cui è possibile trarre vantaggio dalla funzionalità Letture ottimizzate con oggetti temporanei:

- Query analitiche che includono Common Table Expressions (CTEs), tabelle derivate e operazioni di raggruppamento.
- Repliche di lettura che gestiscono le query non ottimizzate per un'applicazione.
- Query di reporting on demand o dinamiche con operazioni complesse come GROUP BY e ORDER BY che non sempre possono utilizzare indici appropriati.
- `CREATE INDEX`o `REINDEX` operazioni di ordinamento.
- Altri carichi di lavoro che utilizzano tabelle temporanee interne.

Monitoraggio delle istanze database che utilizzano Letture ottimizzate per Aurora

Per monitorare le query che utilizzano la cache a più livelli abilitata per Letture ottimizzate utilizza il comando EXPLAIN, come mostrato nell'esempio seguente:

```
Postgres=> EXPLAIN (ANALYZE, BUFFERS) SELECT c FROM sbtest15 WHERE id=100000000


QUERY PLAN
------------------------------------------------------------------------------------
 Index Scan using sbtest15_pkey on sbtest15  (cost=0.57..8.59 rows=1 width=121) (actual
 time=0.287..0.288 rows=1 loops=1)
   Index Cond: (id = 100000000)
   Buffers: shared hit=3 read=2 aurora_orcache_hit=2
   I/O Timings: shared/local read=0.264
 Planning:
   Buffers: shared hit=33 read=6 aurora_orcache_hit=6
   I/O Timings: shared/local read=0.607
 Planning Time: 0.929 ms
 Execution Time: 0.303 ms
(9 rows)
Time: 2.028 ms
```

> **Note**
>
> `aurora_orcache_hite` `aurora_storage_read` i campi nella `Buffers` sezione del piano di spiegazione vengono visualizzati solo quando è attivata l'opzione Lettura ottimizzata e i relativi valori sono maggiori di zero. Il campo di lettura è il totale dei `aurora_storage_read` campi `aurora_orcache_hit` e.

È possibile monitorare le istanze DB che utilizzano Aurora Optimized Reads utilizzando le seguenti metriche: CloudWatch

- `AuroraOptimizedReadsCacheHitRatio`
- `FreeEphemeralStorage`
- `ReadIOPSEphemeralStorage`
- `ReadLatencyEphemeralStorage`
- `ReadThroughputEphemeralStorage`
- `WriteIOPSEphemeralStorage`
- `WriteLatencyEphemeralStorage`
- `WriteThroughputEphemeralStorage`

Queste metriche forniscono dati sullo spazio di archiviazione dell'archivio dell'istanza, sulle operazioni IOPS e sulla velocità di trasmissione effettiva disponibili. Per ulteriori informazioni su questi parametri, consulta Parametri a livello di istanza per Amazon Aurora.

È inoltre possibile utilizzare l'`pg_proctab`estensione per monitorare l'archiviazione. NVMe

```
postgres=>select * from pg_diskusage();

major | minor |       devname       | reads_completed | reads_merged | sectors_read |
 readtime | writes_completed | writes_merged | sectors_written | writetime | current_io
 | iotime  | totaliotime
------+-------+---------------------+-----------------+--------------+--------------
+----------+------------------+---------------+-----------------+-----------
+------------+---------+-------------
```

```
      |       | rdstemp              |           23264 |            0 |       191450 |
  11670 |          1750892 |             0 |        24540576 |    819350 |          0 |
 3847580 |      831020
      |       | rdsephemeralstorage |           23271 |            0 |       193098 |
   2620 |           114961 |             0 |        13845120 |    130770 |          0 |
  215010 |      133410
(2 rows)
```

Best practice per Letture ottimizzate per Aurora

Usa le seguenti best practice per Letture ottimizzate per Aurora:

- Monitora lo spazio di archiviazione disponibile sull'instance store con la CloudWatch metrica`FreeEphemeralStorage`. Se l'instance store sta raggiungendo il limite a causa del carico di lavoro sull'istanza DB, ottimizza la concorrenza e le query che utilizzano pesantemente oggetti temporanei o modificalo per utilizzare una classe di istanze DB più grande.
- Monitora la CloudWatch metrica per il tasso di accesso alla cache di Optimized Reads. Operazioni come VACUUM permettono di modificare blocchi numerosi molto rapidamente. Ciò può causare un calo temporaneo della percentuale di riscontri. Utilizza l'estensione `pg_prewarm` per caricare dati nella cache del buffer che consente ad Aurora di scrivere in modo proattivo alcuni di questi blocchi nella cache delle Letture ottimizzate.
- È possibile abilitare Gestione della cache del cluster (CCM) per preparare la cache del buffer e la cache a più livelli su un lettore di livello 0, che verrà utilizzato come destinazione di failover. Quando la funzionalità CCM è abilitata, la cache del buffer viene scansionata periodicamente per scrivere pagine idonee all'espulsione nella cache a più livelli. Per ulteriori informazioni sulla CCM, consulta Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL.

Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL

Una sottoquery correlata fa riferimento alle colonne della tabella dalla query esterna. Viene valutata una volta per ogni riga restituita dalla query esterna. Nell'esempio seguente, la sottoquery fa riferimento a una colonna dalla tabella ot. Questa tabella non è inclusa nella clausola FROM della subquery, ma è referenziata nella clausola FROM della query esterna. Se la tabella ot ha 1 milione di righe, la sottoquery deve essere valutata 1 milione di volte.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

> **Note**
>
> La trasformazione delle subquery e la cache delle subquery sono disponibili in Aurora PostgreSQL a partire dalla versione 16.8, mentre Babelfish per Aurora PostgreSQL supporta queste funzionalità dalla 4.2.0.

Miglioramento delle prestazioni delle query PostgreSQL di Aurora utilizzando la trasformazione delle subquery

Aurora PostgreSQL può accelerare le sottoquery correlate trasformandole in join esterni equivalenti. Questa ottimizzazione si applica ai seguenti due tipi di sottoquery correlate:

- Sottoquery che restituiscono un singolo valore aggregato e vengono visualizzate nell'elenco SELECT.

```
SELECT ot.a, ot.b, (SELECT AVG(it.b) FROM it WHERE it.a = ot.a) FROM ot;
```

- Sottoquery che restituiscono un singolo valore aggregato e compaiono in una clausola WHERE.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

Abilitazione della trasformazione nella sottoquery

Per abilitare la trasformazione di sottoquery correlate in join esterni equivalenti, impostate il parametro su. `apg_enable_correlated_scalar_transform` `ON` Il valore predefinito di questo parametro è `OFF`.

È possibile modificare il cluster o il gruppo di parametri dell'istanza per impostare i parametri. Per ulteriori informazioni, consulta .

In alternativa, è possibile configurare l'impostazione solo per la sessione corrente con il seguente comando:

```
SET apg_enable_correlated_scalar_transform TO ON;
```

Verifica della trasformazione

Utilizzate il comando EXPLAIN per verificare se la sottoquery correlata è stata trasformata in un outer join nel piano di query.

Quando la trasformazione è abilitata, la parte di subquery correlata applicabile verrà trasformata in outer join. Per esempio:

```
postgres=> CREATE TABLE ot (a INT, b INT);
CREATE TABLE
postgres=> CREATE TABLE it (a INT, b INT);
CREATE TABLE

postgres=> SET apg_enable_correlated_scalar_transform TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);

                        QUERY PLAN
-----------------------------------------------------------
 Hash Join
   Hash Cond: (ot.a = apg_scalar_subquery.scalar_output)
   Join Filter: ((ot.b)::numeric < apg_scalar_subquery.avg)
   ->  Seq Scan on ot
   ->  Hash
         ->  Subquery Scan on apg_scalar_subquery
               ->  HashAggregate
                     Group Key: it.a
                     ->  Seq Scan on it
```

La stessa query non viene trasformata quando viene ruotato il parametro GUC. `OFF` Il piano non avrà un outer join ma un sottopiano.

```
postgres=> SET apg_enable_correlated_scalar_transform TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);
                QUERY PLAN
---------------------------------------
 Seq Scan on ot
   Filter: ((b)::numeric < (SubPlan 1))
   SubPlan 1
```

```
     ->  Aggregate
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Limitazioni

- La sottoquery deve trovarsi nell'elenco SELECT o in una delle condizioni della clausola where. Altrimenti, non verrà trasformata.
- La sottoquery deve restituire una funzione aggregata. Le funzioni di aggregazione definite dall'utente non sono supportate per la trasformazione.
- Una sottoquery la cui espressione restituita non è una semplice funzione aggregata non verrà trasformata.
- La condizione correlata nelle clausole WHERE della subquery dovrebbe essere un semplice riferimento di colonna. Altrimenti, non verrà trasformato.
- La condizione correlata nella subquery in cui le clausole devono essere un semplice predicato di uguaglianza.
- La sottoquery non può contenere né una clausola HAVING né una clausola GROUP BY.
- La clausola where nella sottoquery può contenere uno o più predicati combinati con AND.

> **Note**
>
> L'impatto della trasformazione sulle prestazioni varia a seconda dello schema, dei dati e del carico di lavoro. L'esecuzione di sottoquery correlata alla trasformazione può migliorare significativamente le prestazioni all'aumentare del numero di righe prodotte dalla query esterna. Si consiglia vivamente di testare questa funzionalità in un ambiente non di produzione con lo schema, i dati e il carico di lavoro effettivi prima di abilitarla in un ambiente di produzione.

Utilizzo della cache delle subquery per migliorare le prestazioni delle query PostgreSQL di Aurora

Aurora PostgreSQL supporta la cache delle subquery per archiviare i risultati delle sottoquery correlate. Questa funzionalità salta le ripetute esecuzioni di sottoquery correlate quando i risultati delle subquery sono già presenti nella cache.

Comprendere la cache delle subquery

Il nodo Memoize di PostgreSQL è la parte fondamentale della cache delle subquery. Il nodo Memoize mantiene una tabella hash nella cache locale per mappare i valori dei parametri di input alle righe dei risultati delle query. Il limite di memoria per la tabella hash è il prodotto di work_mem e hash_mem_multiplier. Per ulteriori informazioni, consulta Consumo di risorse.

Durante l'esecuzione delle query, la cache delle subquery utilizza il Cache Hit Rate (CHR) per stimare se la cache sta migliorando le prestazioni delle query e per decidere in fase di esecuzione della query se continuare a utilizzare la cache. CHR è il rapporto tra il numero di accessi alla cache e il numero totale di richieste. Ad esempio, se una sottoquery correlata deve essere eseguita 100 volte e 70 di tali risultati di esecuzione possono essere recuperati dalla cache, il CHR è 0,7.

Per ogni numero di errori nella cache di apg_subquery_cache_check_interval, il vantaggio della cache delle subquery viene valutato controllando se il CHR è maggiore di apg_subquery_cache_hit_rate_threshold. In caso contrario, la cache verrà eliminata dalla memoria e l'esecuzione della query tornerà alla riesecuzione della sottoquery originale, non memorizzata nella cache.

Parametri che controllano il comportamento della cache delle subquery

La tabella seguente elenca i parametri che controllano il comportamento della cache delle subquery.

Parametro	Descrizione	Default	Consentito
apg_enable_subquery_cache	Abilita l'uso della cache per sottoquery scalari correlate.	OFF	ON, OFF
apg_subquery_cache_check_interval	Imposta la frequenza, in numero di errori nella cache, per valutare il tasso di successo della cache delle subquery.	500	0–2147483647
apg_subquery_cache_hit_rate_threshold	Imposta la soglia per il tasso di successo	0.3	0,0—1,0

Parametro	Descrizione	Default	Consentito
	della cache delle subquery.		

> **Note**
>
> - Valori maggiori di `apg_subquery_cache_check_interval` possono migliorare l'accuratezza della stima dei vantaggi della cache basata su CHR, ma aumenteranno il sovraccarico della cache, poiché CHR non verrà valutato finché la tabella della cache non avrà delle righe. `apg_subquery_cache_check_interval`
> - Valori più elevati di `apg_subquery_cache_hit_rate_threshold` inclinazione verso l'abbandono della cache delle sottoquery e il ritorno alla riesecuzione della sottoquery originale non memorizzata nella cache.

È possibile modificare il gruppo di parametri del cluster o dell'istanza per impostare i parametri. Per ulteriori informazioni, consulta .

In alternativa, è possibile configurare l'impostazione solo per la sessione corrente con il seguente comando:

```
SET apg_enable_subquery_cache TO ON;
```

Attivazione della cache delle subquery in Aurora PostgreSQL

Quando la cache delle subquery è abilitata, Aurora PostgreSQL applica la cache per salvare i risultati delle subquery. Il piano di interrogazione avrà quindi un nodo Memoize sotto. SubPlan

Ad esempio, la seguente sequenza di comandi mostra il piano di esecuzione delle query stimato di una semplice sottoquery correlata senza cache delle subquery.

```
postgres=> SET apg_enable_subquery_cache TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
```

```
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Seq Scan on it
           Filter: (a = ot.a)
```

Dopo l'attivazione`apg_enable_subquery_cache`, il piano di query conterrà un nodo Memoize sotto il SubPlan nodo, a indicare che la subquery prevede di utilizzare la cache.

```
postgres=> SET apg_enable_subquery_cache TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Memoize
           Cache Key: ot.a
           Cache Mode: binary
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Il piano di esecuzione delle query effettivo contiene maggiori dettagli sulla cache delle sottoquery, inclusi gli accessi alla cache e gli errori nella cache. L'output seguente mostra l'effettivo piano di esecuzione delle query della query di esempio precedente dopo l'inserimento di alcuni valori nelle tabelle.

```
postgres=> EXPLAIN (COSTS FALSE, TIMING FALSE, ANALYZE TRUE) SELECT ot.a, ot.b FROM ot
 WHERE ot.b < (SELECT it.b FROM it WHERE it.a = ot.a);
            QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on ot (actual rows=2 loops=1)
   Filter: (b < (SubPlan 1))
   Rows Removed by Filter: 8
   SubPlan 1
     ->  Memoize (actual rows=0 loops=10)
           Cache Key: ot.a
           Cache Mode: binary
```

```
          Hits: 4  Misses: 6  Evictions: 0  Overflows: 0  Memory Usage: 1kB
          ->  Seq Scan on it (actual rows=0 loops=6)
                Filter: (a = ot.a)
                Rows Removed by Filter: 4
```

Il numero totale di accessi alla cache è 4 e il numero totale di accessi mancati alla cache è 6. Se il numero totale di accessi e mancati è inferiore al numero di loop nel nodo Memoize, significa che la valutazione CHR non ha avuto esito positivo e che la cache è stata ripulita e abbandonata a un certo punto. L'esecuzione della sottoquery è quindi tornata alla riesecuzione originale non memorizzata nella cache.

Limitazioni

La cache delle subquery non supporta determinati modelli di sottoquery correlate. Questi tipi di query verranno eseguiti senza cache, anche se la cache delle subquery è attivata:

- IN/EXISTS/ANY/ALLsottoquery correlate
- Sottoquery correlate contenenti funzioni non deterministiche.
- Sottoquery correlate che fanno riferimento a colonne esterne della tabella con tipi di dati che non supportano operazioni di hashing o uguaglianza.

Miglioramento delle prestazioni delle query utilizzando adaptive join

Panoramica

Adaptive join è una funzionalità di anteprima di Aurora PostgreSQL 17.4 che aiuta a migliorare le prestazioni delle query. Questa funzionalità è disabilitata per impostazione predefinita, ma è possibile abilitarla utilizzando i parametri di Global User Configuration (GUC). Poiché si tratta di una funzionalità di anteprima, i valori dei parametri predefiniti potrebbero cambiare. Se abilitato, l'adaptive join aiuta a ottimizzare le prestazioni delle query passando dinamicamente da un nested loop join a un hash join in fase di esecuzione. Questo passaggio si verifica quando l'ottimizzatore PostgreSQL ha scelto erroneamente un loop join annidato a causa di stime di cardinalità imprecise.

Configurazione del join adattivo

È possibile controllare il join adattivo utilizzando questi tre parametri GUC:

Parametri di configurazione del join adattivo

Parametro GUC	Descrizione	Opzioni predefinite e di configurazione
apg_adaptive_join_crossover_multiplier	Questo moltiplicatore funziona con il punto di incrocio delle righe per determinare quando passare da un loop annidato a un hash join. Il punto di incrocio delle righe è il punto in cui l'ottimizzatore SQL stima che le operazioni di nidificato loop e hash join abbiano lo stesso costo. Un valore di moltiplicatore più elevato riduce la probabilità che il join adattivo passi a un hash join.	Controlla se Adaptive Join è abilitato • Valore predefinito: -1 (disabilitato) • Intervallo valido: da -1 a DBL_MAX • Per abilitare: Impostare su >= 1
apg_adaptive_join_cost_threshold	Questo parametro imposta una soglia minima di costo delle query. Adaptive Join si disattiva automaticamente per le query al di sotto di questa soglia. In questo modo si evita un sovraccarico prestazionale nelle query semplici in cui il costo della pianificazione di un join adattivo potrebbe superare i vantaggi del passaggio dal nested loop all'hash join.	Imposta la soglia di costo minima per la query • Valore predefinito: 100 • Intervallo valido: da 0 a DBL_MAX
apg_enable_parameterized_adaptive_join	Se abilitato, questo parametro estende la funzionalità di adaptive join ai loop join annidati parametrizzati. Per impostazione predefinita, l'adaptive join funziona solo con loop join annidati non parametrizzati, poiché è più probabile che questi traggano vantaggio dal passaggio all'hash join. I loop join annidati parametrizzati in genere offrono	Controlla il comportamento adattivo dei loop join annidati • Valore predefinito: false • Valori validi: true/false • Quando falso: funziona solo con loop join annidati non parametrizzati

Parametro GUC	Descrizione	Opzioni predefinite e di configurazione
	prestazioni migliori, rendendo meno importante il passaggio all'hash join.	• Quando è vero: funziona con loop join annidati parametrizzati e non parametrizzati **Note** Deve essere prima abilitato `apg_adaptive_join_crossover_multiplier`

Utilizzo di tabelle non registrate in Aurora PostgreSQL

Amazon Aurora PostgreSQL supporta tabelle non registrate che sono resistenti agli arresti anomali e mantengono l'integrità dei dati anche dopo errori o failover delle istanze di scrittura. In PostgreSQL standard, le tabelle non registrate ignorano il Write-Ahead Log (WAL) durante le operazioni di scrittura, con conseguente velocità di scrittura più elevate. Tuttavia, ciò comporta una riduzione della durabilità, in quanto le tabelle non registrate non sono a prova di crash e possono perdere dati dopo un guasto del sistema o un arresto non corretto. Queste tabelle non registrate vengono troncate automaticamente dopo un arresto anomalo o un arresto non corretto. Inoltre, i loro contenuti e indici non vengono replicati sui server di standby.

Al contrario, Aurora PostgreSQL gestisce le tabelle non registrate in modo diverso grazie alla sua architettura di archiviazione distribuita. Questo perché il sistema di storage di Aurora non si basa sul tradizionale WAL PostgreSQL per la durabilità. Tuttavia, i vantaggi in termini di prestazioni tipicamente associati alle tabelle non registrate in PostgreSQL standard potrebbero non essere così significativi in Aurora. Ciò è dovuto all'architettura di storage distribuito di Aurora, che può introdurre un sovraccarico aggiuntivo rispetto allo storage locale utilizzato in PostgreSQL standard.

Quando utilizzi tabelle non registrate in Aurora PostgreSQL, considera quanto segue:

- È possibile accedere alle tabelle non registrate solo dal nodo writer nel cluster Aurora DB.
- I nodi Reader possono accedere alle tabelle non registrate solo quando vengono promossi allo stato di scrittore.
- Quando si tenta di accedere a tabelle non registrate da un nodo di lettura, si verificherà il seguente errore:

```
cannot access temporary or unlogged relations during recovery.
```

Creazione di tabelle non registrate

Per creare una tabella non registrata in Aurora PostgreSQL, aggiungi la parola chiave UNLOGGED nell'istruzione CREATE TABLE:

```
CREATE UNLOGGED TABLE staging_sales_data (
    transaction_id bigint,
    customer_id bigint,
    product_id bigint,
    transaction_date date,
    amount NUMERIC
);
```

Conversione di tabelle non registrate in tabelle registrate

Quando è necessario riconvertire una tabella non registrata in una tabella registrata, è possibile utilizzare il seguente comando:

```
ALTER TABLE table_name SET LOGGED;
```

Questa operazione riscrive l'intera tabella e la blocca in modo esclusivo fino al completamento dell'operazione. Per le tabelle di grandi dimensioni, ciò può comportare tempi di inattività significativi.

Tabelle non registrate e replica logica

Le tabelle non registrate generalmente non sono incluse nella replica logica perché la replica logica si basa sul WAL per acquisire e trasferire le modifiche. Per impostazione predefinita, le modifiche alle

tabelle non registrate non vengono registrate da WAL ed escluse dal flusso di replica, il che le rende inadatte ai casi d'uso in cui è richiesta la replica logica. Tuttavia, Aurora PostgreSQL fornisce un parametro chiamato `rds.logically_replicate_unlogged_tables` che consente di controllare questo comportamento:

- Quando `rds.logically_replicate_unlogged_tables` è impostato su 0 (off), le tabelle non registrate vengono escluse dalla replica logica.
- Quando `rds.logically_replicate_unlogged_tables` è impostato su 1 (acceso), le tabelle non registrate vengono incluse nella replica logica.

> **Note**
>
> In Aurora PostgreSQL, il `rds.logically_replicate_unlogged_tables` parametro è impostato per impostazione predefinita su 1 (attivo) nelle versioni 14 e precedenti e su 0 (disattivato) nelle versioni 15 e successive.

Utilizzo dell'autovacuum di PostgreSQL su Aurora PostgreSQL

Consigliamo vivamente di usare la caratteristica di autovacuum per mantenere l'integrità dell'istanza database di PostgreSQL. La funzione di autovaacuum automatizza l'esecuzione del comando VACUUM e ANALYZE. Verifica la presenza di tabelle con un numero elevato di tuple inserite, aggiornate o eliminate. Dopo questa verifica, recupera lo storage rimuovendo i dati obsoleti o le tuple da database PostgreSQL.

Per impostazione predefinita, l'autovacuum è attivato per le istanze utilizzando uno qualsiasi dei gruppi di parametri PostgreSQL DB predefiniti. Per impostazione predefinita vengono impostati anche altri parametri di configurazione associati alla caratteristica di autovacuum. Poiché questi valori di default sono in qualche modo generici, è possibile trarre vantaggio dalla regolazione di alcuni parametri associati alla caratteristica di autovacuum per il carico di lavoro specifico.

Di seguito, puoi trovare ulteriori informazioni sull'autovacuum e su come ottimizzare alcuni dei suoi parametri sull'istanza DB .

Argomenti

- Allocazione di memoria per il vacuum
- Riduzione della probabilità che si verifichi il wraparound dell'ID delle transazioni

- Determinare se le tabelle nel database devono essere sottoposte a vacuum
- Determinare quali tabelle sono attualmente idonee per l'Autovacuum
- Determinare se l'Auotvacuum è attualmente in esecuzione e per quanto tempo
- Esecuzione di un congelamento manuale del vacuum
- Indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione
- Gestione di autovacuum con indici di grandi dimensioni
- Altri parametri che influenzano l'autovacuum
- Impostazione dei parametri di autovacuum a livello tabella
- Registrazione delle attività di autovacuum e vacuum
- Comprensione del comportamento dell'autovacuum con database non validi
- Identifica e risolvi i vacuoblocker aggressivi in

Allocazione di memoria per il vacuum

Uno dei parametri più importanti che influenzano le prestazioni dell'autovacuum è il parametro `autovacuum_work_mem`. Nelle versioni 14 e `autovacuum_work_mem` precedenti di Aurora PostgreSQL, il parametro è impostato su -1, a indicare che al suo posto viene utilizzata l'impostazione di. `maintenance_work_mem` Per tutte le altre versioni, `autovacuum_work_mem` è determinato da GREATEST ({/32768}, 65536). DBInstance ClassMemory

Le operazioni manuali di aspirazione utilizzano sempre l'`maintenance_work_mem`impostazione, con un'impostazione predefinita di GREATEST ({DBInstanceClassMemory/63963136 *1024}, 65536), e può anche essere regolata a livello di sessione utilizzando il comando per operazioni manuali più mirate. `SET VACUUM`

`autovacuum_work_mem`Determina che la memoria di autovacuum contenga gli identificatori delle tuple morte () per gli indici di aspirazione. `pg_stat_all_tables.n_dead_tup`

Quando esegui calcoli per determinare il `autovacuum_work_mem` valore del parametro, tieni presente quanto segue:

- Se impostate il parametro su un valore troppo basso, il processo di sottovuoto potrebbe dover scansionare la tabella più volte per completare il suo lavoro. Queste scansioni multiple possono avere un impatto negativo sulle prestazioni Per istanze più grandi, l'impostazione `maintenance_work_mem` o `autovacuum_work_mem` su almeno 1 GB può migliorare le prestazioni delle tabelle di aspirazione con un numero elevato di tuple morte. Tuttavia, nelle

versioni 16 e precedenti di PostgreSQL, l'utilizzo della memoria di vacuum è limitato a 1 GB, il che è sufficiente per elaborare circa 179 milioni di tuple morte in un unico passaggio. Se una tabella ha più tuple morte di questa, vacuum dovrà effettuare più passaggi tra gli indici della tabella, aumentando notevolmente il tempo necessario. A partire dalla versione 17 di PostgreSQL, non c'è un limite di 1 GB e autovacuum può elaborare più di 179 milioni di tuple utilizzando alberi radix.

Un identificatore di tupla ha una dimensione di 6 byte. Per stimare la memoria necessaria per svuotare un indice di una tabella, esegui una query `pg_stat_all_tables.n_dead_tup` per trovare il numero di tuple morte, quindi moltiplica questo numero per 6 per determinare la memoria necessaria per svuotare l'indice in un unico passaggio. È possibile utilizzare la seguente interrogazione:

```
SELECT
    relname AS table_name,
    n_dead_tup,
    pg_size_pretty(n_dead_tup * 6) AS estimated_memory
FROM
    pg_stat_all_tables
WHERE
    relname = 'name_of_the_table';
```

- Il parametro `autovacuum_work_mem` funziona insieme al parametro `autovacuum_max_workers`. Ogni lavoratore `autovacuum_max_workers` può utilizzare la memoria allocata. Se si dispone di molte tabelle di piccole dimensioni, assegna più `autovacuum_max_workers` e meno `autovacuum_work_mem`. Se disponi di tabelle di grandi dimensioni (più grandi di 100 GB), alloca più memoria e meno processi di lavoro. Si deve avere abbastanza memoria allocata affinché si abbia esito positivo sulle tabelle più grandi. Pertanto, assicuratevi che la combinazione di processi di lavoro e memoria sia uguale alla memoria totale che desiderate allocare.

Riduzione della probabilità che si verifichi il wraparound dell'ID delle transazioni

In alcuni casi, le impostazioni del gruppo di parametri correlate all'autovacuum potrebbero non essere abbastanza aggressive da prevenire il wraparound dell'ID delle transazioni. Per risolvere questo problema, fornisce un meccanismo che adatta automaticamente i valori dei parametri autovacuum. L'autovacuum adattivo è una funzionalità di RDS per PostgreSQL Aurora . Una spiegazione dettagliata di Wraparound della transazione si trova nella documentazione di PostgreSQL.

L'autovacuum adattivo è attivato per impostazione predefinita per RDS per le istanze dinamico impostato su ON. `rds.adaptive_autovacuum` Si consiglia di tenere questa opzione attivata. Tuttavia, per disattivare l'ottimizzazione adattiva del parametro di autovacuum, impostare il parametro `rds.adaptive_autovacuum` su 0 o OFF.

Il wraparound degli ID delle transazioni è ancora possibile anche quando Amazon ottimizza i parametri dell'autovacuum. Ti invitiamo a implementare un CloudWatch allarme Amazon per il wraparound degli ID delle transazioni. Per ulteriori informazioni, consulta il post Implementazione di un sistema di avviso rapido per il wraparound degli ID delle transazioni in RDS per PostgreSQL sul blog del database. AWS

Con l'ottimizzazione adattiva dei parametri dell'autovacuum attivata, Amazon RDS inizia a regolare i parametri dell'autovacuum quando la CloudWatch metrica `MaximumUsedTransactionIDs` raggiunge il valore del parametro o 500.000.000, a seconda di quale sia il maggiore. `autovacuum_freeze_max_age`

Amazon RDS continua ad adattare i parametri per l'autovacuum se una tabella continua a tendere verso i wraparound dell'ID della transazione. Ognuno di questi aggiustamenti dedica più risorse all'autovacuum per evitare il wraparound. Amazon RDS aggiorna i seguenti parametri correlati all'autovacuum:

- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`
- autovacuum_naptime

RDS modifica questi parametri solo se il nuovo valore rende l'autovacuum più aggressivo. I parametri vengono modificati nella memoria sull'istanza database. I valori nel gruppo di parametri non vengono modificati. Per visualizzare le impostazioni in memoria correnti, utilizzare il comando SQL PostgreSQL SHOW.

Quando Amazon RDS modifica uno qualsiasi dei parametri autovacuum, genera un evento per l'istanza database interessata. Questo evento è visibile su AWS Management Console e tramite l'API Amazon RDS. Dopo che la `MaximumUsedTransactionIDs` CloudWatch metrica è tornata al di sotto della soglia, Amazon RDS ripristina i parametri relativi all'autovacuum in memoria ai valori specificati nel gruppo di parametri. Quindi genera un altro evento corrispondente a questa modifica.

Determinare se le tabelle nel database devono essere sottoposte a vacuum

È possibile utilizzare la seguente query per mostrare il numero di transazioni non bloccate in un database. La `datfrozenxid` colonna della `pg_database` riga di un database è il limite inferiore della normale transazione che IDs appare in quel database. Questa colonna è il minimo dei valori `relfrozenxid` per tabella all'interno del database.

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

Ad esempio, i risultati dell'esecuzione della query precedente potrebbero essere i seguenti.

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

Quando l'età di un database raggiunge i 2 miliardi di transazioni IDs, si verifica l'aggiustamento dell'ID della transazione (XID) e il database diventa di sola lettura. Puoi utilizzare questa query per produrre un parametro ed eseguirla alcune volte al giorno. Per impostazione predefinita, l'autovacuum è impostato per mantenere l'età delle transazioni a non più di 200,000,000 (`autovacuum_freeze_max_age`).

Un esempio di strategia di monitoraggio potrebbe avere questo aspetto:

- Impostare il valore `autovacuum_freeze_max_age` su 200 milioni di transazioni.
- Se una tabella raggiunge 500 milioni di transazioni non bloccate, viene attivato un allarme di bassa gravità. Questo non è un valore irragionevole, ma può indicare che l'autovacuum non riesce a mantenere il passo.
- Se una tabella invecchia a un miliardo, questo dovrebbe essere trattato come un allarme per cui intervenire. In generale, si desidera mantenere le età più vicine a `autovacuum_freeze_max_age` per motivi di prestazioni. Si consiglia di investigare utilizzando le raccomandazioni che seguono.
- Se una tabella raggiunge 1,5 milioni di transazioni senza vacuum, viene attivato un allarme a gravità elevata. A seconda della velocità con cui il database utilizza la transazione IDs, questo

allarme può indicare che il tempo necessario per eseguire l'autovacuum sta per scadere. In questo caso, consigliamo di risolvere il problema immediatamente.

Se una tabella superando costantemente queste soglie, modifica ulteriormente i parametri dell'autovacuum. Per impostazione predefinita, l'utilizzo manuale di VACUUM (che ha disabilitato i ritardi basati sui costi) è più aggressivo dell'autovacuum predefinito, ma è anche più intrusivo per il sistema nel suo complesso.

Consigliamo quanto segue:

- Attiva un meccanismo di monitoraggio in modo da essere consapevole dell'età delle transazioni più vecchie.

  Per informazioni sulla creazione di un processo che ti avvisa in merito al wraparound degli ID delle transazioni, consulta il post del AWS Database Blog Implementa an early warning system for Transaction ID wraparound in Amazon RDS for PostgreSQL.
- Per le tabelle più occupate, eseguire regolarmente un congelamento manuale del vacuum durante una finestra di manutenzione, oltre a fare affidamento sull'autovacuum. Per informazioni sull'esecuzione di un congelamento manuale del vacuum, consulta Esecuzione di un congelamento manuale del vacuum.

Determinare quali tabelle sono attualmente idonee per l'Autovacuum

Spesso, una o due tabelle hanno bisogno del vacuum. Le tabelle il cui valore `relfrozenxid` sia maggiore del numero di transazioni in `autovacuum_freeze_max_age` sono sempre destinate all'autovacuum. Altrimenti, se il numero di tuple reso obsoleto dall'ultimo VACUUM supera la soglia del vacuum, la tabella viene sottoposta a vacuum.

La autovacuum threshold (soglia di autovacuum) viene definita come:

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

dove l'`vacuum base threshold`è`autovacuum_vacuum_threshold`, l'`vacuum scale factor`è `autovacuum_vacuum_scale_factor` e il `number of tuples` è`pg_class.reltuples`.

Mentre sei connesso al database, esegui la seguente query per visualizzare un elenco di tabelle che Autovacuum vede come idonee per il vacuuming.

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Determinare se l'Auotvacuum è attualmente in esecuzione e per quanto tempo

Se è necessario procedere manualmente con il vacuum in una tabella, devi determinare se l'autovacuum è attualmente in esecuzione. In tal caso, potrebbe essere necessario regolare i parametri per farlo eseguire in modo più efficiente oppure disattivare l'autovacuum temporaneamente in modo da poter eseguire manualmente il VACUUM.

Utilizzare la seguente query per determinare se l'autovacuum è in esecuzione, da quanto tempo è in esecuzione e se è in attesa su un'altra sessione.

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

Dopo l'esecuzione della query, si dovrebbe visualizzare un output simile a quello riportato di seguito.

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-------------------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | active |            | 00:00:00                |
 SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query+
         |          |       |        |            |                         | FROM
 pg_stat_activity
           +
         |          |       |        |            |                         | WHERE
 query like '%VACUUM%'
          +
         |          |       |        |            |                         | ORDER BY
 xact_start;
       +
```

Diversi problemi possono provocare sessioni di autovacuum di lunga esecuzione (che durano più giorni). Il problema più comune è che il valore del parametro `maintenance_work_mem` è impostato come troppo basso per la dimensione della tabella o la frequenza degli aggiornamenti.

Consigliamo di utilizzare la seguente formula per impostare il valore del parametro `maintenance_work_mem`.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

Le sessioni autovacuum a breve esecuzioni possono anche indicare dei problemi:

- Può indicare che non ci sono abbastanza `autovacuum_max_workers` per il carico di lavoro. In questo caso, sarà necessario indicare il numero di lavoratori.
- Può indicare che esiste un danneggiamento dell'indice (l'autovacuum si blocca e si riavvia sulla stessa relazione ma non ci sono progressi). In questo caso, esegui un `vacuum freeze verbose` *`table`* manuale per vedere la causa esatta.

Esecuzione di un congelamento manuale del vacuum

Si potrebbe voler eseguire un vacuum manuale su una tabella che ha già un processo di vacuum in esecuzione. Ciò è utile se hai identificato una tabella con un'età che si avvicina a 2 miliardi (o al di sopra di qualsiasi soglia monitorata).

I seguenti passaggi sono delle linea guida, con diverse varianti del processo. Ad esempio, durante la verifica, supporre che il valore del parametro `maintenance_work_mem` sia stato impostato come troppo piccolo e che sia necessario agire immediatamente su una tabella. Tuttavia, probabilmente al momento non desideri che l'istanza non venga recapitata. Utilizzando le query delle sezioni precedenti, si determina quale tabella rappresenta il problema e si nota una sessione autovacuum a lunga esecuzione. Si sa che è necessario cambiare l'impostazione del parametro `maintenance_work_mem`, ma è necessario anche agire immediatamente ed eseguire il vacuum della tabella in questione. La procedura seguente mostra cosa fare in questa situazione.

Per eseguire manualmente un congelamento del vacuum

1. Aprire due sessioni nel database che contiene la tabella che si desidera sottoporre a vacuum. Per la seconda sessione, utilizzare "screen" o un'altra utility che mantiene la sessione se la connessione viene interrotta.
2. Nella sessione uno, ottieni l'ID di processo (PID) della sessione di autovacuum in esecuzione sulla tabella.

   Eseguire la query seguente per ottenere il PID della sessione di autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   ```

```
xact_start;
```

3. Nella sessione due, calcoli la quantità di memoria necessaria per questa operazione. In questo esempio, stabiliamo che è possibile permettersi di utilizzare fino a 2 GB di memoria per questa operazione, pertanto abbiamo impostato `maintenance_work_mem` per la sessione corrente su 2 GB.

```
SET maintenance_work_mem='2 GB';
SET
```

4. Nella sessione 2, inviare un comando `vacuum freeze verbose` per la tabella. L'impostazione di verbose è utile perché, anche se al momento non vi è alcun rapporto sullo stato di avanzamento in PostgreSQL, è possibile visualizzare l'attività.

```
\timing on
Timing is on.
vacuum freeze verbose pgbench_branches;
```

```
INFO:  vacuuming "public.pgbench_branches"
INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. Nella prima sessione, se l'autovacuum bloccava la sessione di aspirazione, `pg_stat_activity` indica che l'Tattesa riguarda la sessione di aspirazione. In questo caso, terminate il processo di aspirazione automatica come segue.

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Alcune versioni precedenti di Aurora non possono terminare un processo di autovacuum utilizzando il comando precedente e falliscono con il seguente errore:. `ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227` Per trovare le versioni di PostgreSQL a cui è stata applicata la patch, cerca il seguente bullet negli aggiornamenti di PostgreSQL di Amazon Aurora:
>
> ```
> Allow rds_superuser to terminate backends which are not explicitly
>  associated with a role
> ```

In questo momento, inizia la sessione. Autovacuum si riavvia immediatamente perché questa tabella è probabilmente la più alta nella sua lista di lavori.

6. Avvia il comando `vacuum freeze verbose` nella sessione due, quindi termina il processo di autovacuum nella sessione uno.

Indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione

Se un indice diventa corrotto, l'autovacuum continua a elaborare la tabella e avrà esito negativo. Setenti di eseguire un vacuum manuale in questa situazione, riceverai un messaggio di errore come il seguente.

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

Quando l'indice è corrotto e l'autovacuum sta tentando l'esecuzione sulla tabella, ci sarà una contesa con una sessione di autovacuum già in esecuzione. Quando si immette un comando REINDEX, si richiede un blocco esclusivo sulla tabella. Le operazioni in scrittura sono bloccate e anche quelle in lettura che utilizzano l'indice specifico.

Per indicizzare di nuovo una tabella quando l'autovacuum è in esecuzione sulla tabella

1. Apri due sessioni nel database che contiene la tabella da sottoporre a vacuum. Per la seconda sessione, utilizzare "screen" o un'altra utility che mantiene la sessione se la connessione viene interrotta.
2. Nella sessione numero uno, ottenere il PID della sessione di autovacuum in esecuzione sulla tabella.

   Eseguire la query seguente per ottenere il PID della sessione di autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```

3. Nella sessione due, rilasciare il comando di reindicizzazione.

   ```
   \timing on
   Timing is on.
   reindex index pgbench_branches_test_index;
   REINDEX
     Time: 9.966 ms
   ```

4. Nella sessione uno, se l'autovacuum bloccava il processo, in `pg_stat_activity` vedi che l'attesa è "T" per la sessione di vacuum. In questo caso, termina il processo di autovacuum.

   ```
   SELECT pg_terminate_backend('the_pid');
   ```

   In questo momento, inizia la sessione. È importante notare che l'autovacuum si riavvia immediatamente poiché questa tabella è probabilmente la più alta nella lista di lavori.
5. Avvia il comando nella sessione due, quindi termina il processo di autovacuum nella sessione 1.

Gestione di autovacuum con indici di grandi dimensioni

Come parte del funzionamento, autovacuum esegue diverse fasi di vacuum mentre viene eseguito su una tabella. Prima che la tabella venga pulita, tutti i suoi indici vengono prima sottoposti al vacuum. Quando si rimuovono più indici di grandi dimensioni, questa fase richiede una notevole quantità di tempo e risorse. Pertanto, come best practice, assicurati di controllare il numero di indici in una tabella ed eliminare gli indici non utilizzati.

Per questo processo, controlla innanzitutto la dimensione complessiva degli indici. Quindi, determina se ci sono indici potenzialmente inutilizzati da rimuovere come mostrato negli esempi seguenti.

Per verificare la dimensione della tabella e dei relativi indici

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
pg_size_pretty
11 GB
(1 row)
```

In questo esempio, la dimensione degli indici è maggiore della tabella. Questa differenza può causare problemi di prestazioni perché gli indici sono aumentati in dimensioni o inutilizzati, il che influisce sull'autovacuum e sulle operazioni di inserimento.

Per verificare la presenza di indici non utilizzati

Utilizzando la visualizzazione pg_stat_user_indexes, è possibile verificare la frequenza con cui viene utilizzato un indice con la colonna `idx_scan`. Nell'esempio seguente, gli indici non utilizzati hanno `idx_scan` con il valore `0`.

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
-------+------------+------------+------------------+-----------------------+----------
+--------------+--------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
```

```
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
public      | pgbench_accounts | index_d               | 0
public      | pgbench_accounts | index_e               | 0
public      | pgbench_accounts | index_c               | 0
public      | pgbench_accounts | index_a               | 0
(7 rows)
```

> **Note**
>
> Queste statistiche sono incrementali dal momento in cui vengono ripristinate. Supponi di avere un indice utilizzato solo alla fine di un trimestre lavorativo o solo per un report specifico. È possibile che questo indice non sia stato utilizzato da quando le statistiche sono state ripristinate. Per ulteriori informazioni, consulta Funzioni statistiche. Gli indici utilizzati per garantire l'univocità non vengono sottoposti ad analisi e non devono essere identificati come indici non utilizzati. Per identificare gli indici non utilizzati, è necessario avere una conoscenza approfondita dell'applicazione e delle relative query.

Per verificare quando le statistiche sono state ripristinate l'ultima volta per un database, usa `pg_stat_database`

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
```

```
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

Vacuum di una tabella il più rapidamente possibile

RDS per PostgreSQL 12 e versioni successive

Se sono presenti troppi indici in una tabella di grandi dimensioni, l'istanza database potrebbe essere vicina al wraparound dell'ID di transazione (XID), ovvero quando il contatore XID arriva a zero. Se non controllata, questa situazione potrebbe causare la perdita di dati. Tuttavia, è possibile eseguire rapidamente il vacuum della tabella senza ripulire gli indici. In RDS per PostgreSQL 12 e versioni successive, puoi usare VACUUM con la clausola `INDEX_CLEANUP`.

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

Se è già in esecuzione una sessione di autovacuum, è necessario interromperla per iniziare il VACUUM manuale. Per informazioni sull'esecuzione di un congelamento manuale del vacuum, consulta Esecuzione di un congelamento manuale del vacuum.

> **Note**
>
> Saltare la pulizia dell'indice causa regolarmente un aumento dell'indice, che riduce le prestazioni di scansione. L'indice mantiene le righe morte e la tabella mantiene i puntatori delle linee morte. Di conseguenza, `pg_stat_all_tables.n_dead_tup` aumenta fino a quando non viene eseguito l'autovacuum o un VACUUM manuale con pulizia dell'indice. È consigliabile utilizzare questa procedura solo per evitare che l'ID della transazione venga alterato.

RDS per PostgreSQL 11 e versioni precedenti

Tuttavia, in RDS per PostgreSQL 11 e versioni precedenti, l'unico modo per eseguire il vacuum più rapidamente è riducendo il numero di indici su una tabella. L'eliminazione di un indice può influire sui piani di query. Ti consigliamo di eliminare prima gli indici inutilizzati, quindi quelli che hanno il wraparound XID molto vicino. Una volta completato il processo di vacuum, è possibile ricreare questi indici.

Altri parametri che influenzano l'autovacuum

La query seguente mostra i valori di alcuni dei parametri che influenzano direttamente l'autovacuum e il suo comportamento. I parametri di autovacuum vengono descritti in forma completa nella documentazione di PostgreSQL.

```
SELECT name, setting, unit, short_desc
FROM pg_settings
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

Mentre questi riguardano tutti l'autovacuum, alcuni dei più importanti sono:

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

Impostazione dei parametri di autovacuum a livello tabella

Puoi impostare i parametri di archiviazione correlati all'autovacuum a livello di tabella, che può essere meglio di alterare il comportamento dell'intero database. Per le tabelle di grandi dimensioni, potrebbe essere necessario regolare impostazioni aggressive e si potrebbe non desiderare di eseguire l'autovacuum in questo modo per tutte le tabelle.

La query seguente mostra quali tabelle attualmente dispongono di opzioni a livello di tabella.

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

Un esempio in cui ciò potrebbe essere utile è per tabelle che sono molto più grandi rispetto al resto delle tabelle. Supponi di disporre di una tabella da 300 GB e di altre 30 tabelle da meno di un GB. Se disponi di una tabella da 300 GB e di altre 30 tabelle da meno di 1 GB, puoi impostare alcuni parametri specifici per la tabella di grandi dimensioni in modo da non alterare il comportamento dell'intero sistema.

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

In questo modo si disattiva il ritardo dell'autovacuum basato sul costo per questa tabella a spese di un maggiore utilizzo delle risorse sul sistema. Normalmente, l'autovacuum si ferma per `autovacuum_vacuum_cost_delay` ogni volta che viene raggiunto `autovacuum_cost_limit`. Per ulteriori dettagli, consulta la documentazione di PostegreSQL relativa al vacuuming basato sul costo.

Registrazione delle attività di autovacuum e vacuum

Le informazioni sulle attività dell'autovacuum vengono inviate a `postgresql.log` in base al livello specificato nel parametro `rds.force_autovacuum_logging_level`. Di seguito sono riportati i valori consentiti per questo parametro e le versioni di PostgreSQL per le quali tale valore è l'impostazione predefinita:

- `disabled` (PostgreSQL 10, PostgreSQL 9.6)
- `debug5`, `debug4`, `debug3`, `debug2`, `debug1`
- `info` (PostgreSQL 12, PostgreSQL 11)

- `notice`
- `warning` (PostgreSQL 13 e versioni successive)
- `error`, log, `fatal`, `panic`

`rds.force_autovacuum_logging_level` funziona con il parametro `log_autovacuum_min_duration`. Il valore del parametro `log_autovacuum_min_duration` è la soglia (in millisecondi) al di sopra della quale vengono registrate le azioni dell'autovacuum. Un ambiente di `-1` non registra nulla, mentre un'impostazione di 0 registra tutte le azioni. Come con `rds.force_autovacuum_logging_level`, i valori predefiniti per `log_autovacuum_min_duration` dipendono dalla versione, come segue:

- `10000 ms` – PostgreSQL 14, PostgreSQL 13, PostgreSQL 12 e PostgreSQL 11
- `(empty)` – Nessun valore predefinito per PostgreSQL 10 e PostgreSQL 9.6

Consigliamo di impostare `rds.force_autovacuum_logging_level` su `WARNING`. Consigliamo anche di impostare `log_autovacuum_min_duration` su un valore compreso tra 1000 e 5000. Un'impostazione di 5000 registri di attività che richiede più di 5000 millisecondi. Qualsiasi impostazione diversa da -1 registra anche i messaggi se l'azione dell'autovacuum viene ignorata a causa di un blocco in conflitto o di relazioni interrotte simultaneamente. Per ulteriori informazioni, consulta la pagina relativa al vacuum automatico nella documentazione di PostgreSQL.

Per risolvere i problemi, è possibile modificare il parametro `rds.force_autovacuum_logging_level` in uno dei livelli di debug, da `debug1` fino a `debug5` per le informazioni più dettagliate. Si consiglia di utilizzare le impostazioni di debug per brevi periodi di tempo e solo per la risoluzione dei problemi. Per ulteriori informazioni, consulta Quando registrare nella documentazione di PostgreSQL.

> **Note**
>
> PostgreSQL consente all'account `rds_superuser` di visualizzare le sessioni di autovacuum in `pg_stat_activity`. Ad esempio, è possibile identificare e terminare una sessione di autovacuum che blocca l'esecuzione di un comando o l'esecuzione più lenta di un comando vacuum emesso manualmente.

Comprensione del comportamento dell'autovacuum con database non validi

`-2`Viene introdotto un nuovo valore nella `datconnlimit` colonna del `pg_database` catalogo per indicare come non validi i database che sono stati interrotti durante l'operazione DROP DATABASE.

- 15.4 e tutte le versioni successive
- 14.9 e versioni successive
- 13.12 e versioni successive
- 12.16 e versioni successive
- 11.21 e versioni successive

I database non validi non influiscono sulla capacità di autovacuum di bloccare le funzionalità dei database validi. Autovacuum ignora i database non validi. Di conseguenza, le normali operazioni di vacuum continueranno a funzionare correttamente ed efficacemente per tutti i database validi nell'ambiente PostgreSQL.

Argomenti

- Monitoraggio dell'ID della transazione
- Regolazione della query di monitoraggio
- Risoluzione del problema relativo al database non valido

Monitoraggio dell'ID della transazione

La `age(datfrozenxid)` funzione viene comunemente utilizzata per monitorare l'età dell'ID di transazione (XID) dei database per evitare che l'ID delle transazioni si verifichi.

Poiché i database non validi vengono esclusi dall'autovacuum, il relativo contatore di ID di transazione (XID) può raggiungere il valore massimo di`2 billion`, completare e continuare questo ciclo all'infinito. `- 2 billion` Una tipica query per monitorare il wraparound dell'ID di transazione potrebbe essere simile a:

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

Tuttavia, con l'introduzione del valore -2 per`datconnlimit`, i database non validi possono alterare i risultati di questa query. Poiché questi database non sono validi e non devono essere sottoposti a

regolari controlli di manutenzione, possono causare falsi positivi, inducendo a credere che il valore `age(datfrozenxid)` sia superiore a quello reale.

Regolazione della query di monitoraggio

Per garantire un monitoraggio accurato, è necessario modificare la query di monitoraggio per escludere database non validi. Segui questa query consigliata:

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
WHERE
    datconnlimit <> -2;
```

Questa query garantisce che nel `age(datfrozenxid)` calcolo vengano presi in considerazione solo i database validi, fornendo un riflesso fedele dell'età degli ID delle transazioni nell'ambiente PostgreSQL.

Risoluzione del problema relativo al database non valido

Quando si tenta di connettersi a un database non valido, è possibile che venga visualizzato un messaggio di errore simile al seguente:

```
postgres=> \c db1
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

Inoltre, se il `log_min_messages` parametro è impostato su `DEBUG2` o superiore, è possibile che vengano visualizzate le seguenti voci di registro che indicano che il processo autovacuum sta ignorando il database non valido:

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
```

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

Per risolvere il problema, segui le istruzioni `HINT` fornite durante il tentativo di connessione. Connettiti a qualsiasi database valido utilizzando il tuo account master RDS o un account di database con il `rds_superuser` ruolo ed elimina i database non validi.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

Identifica e risolvi i vacuoblocker aggressivi in

In PostgreSQL, l'vacuuming è fondamentale per garantire la salute del database in quanto recupera lo spazio di archiviazione e previene problemi di wraparound degli ID delle transazioni. Tuttavia, ci sono momenti in cui è possibile impedire che l'vacuuming funzioni come desiderato, il che può comportare un peggioramento delle prestazioni, un aumento dello storage e persino influire sulla disponibilità dell'istanza DB a causa della configurazione degli ID delle transazioni. Pertanto, l'identificazione e la risoluzione di questi problemi sono essenziali per garantire prestazioni e disponibilità ottimali del database. Leggi Understanding autovacuum in ambienti Amazon RDS for PostgreSQL per saperne di più sull'autovacuum.

La `postgres_get_av_diag()` funzione aiuta a identificare i problemi che impediscono o ritardano l'avanzamento del vuoto aggressivo. Vengono forniti suggerimenti, che possono includere comandi per risolvere il problema laddove sia identificabile o linee guida per ulteriori diagnosi laddove il problema non sia identificabile. I vacuobloccanti aggressivi vengono segnalati quando l'età supera la soglia dell'autovuoto adattivo di RDS, pari a 500 milioni di transazioni. IDs

Qual è l'età dell'ID della transazione?

La `age()` funzione per la transazione IDs calcola il numero di transazioni avvenute dal più vecchio ID di transazione non bloccato per un database (`pg_database.datfrozenxid`) o una tabella (`pg_class.relfrozenxid`). Questo valore indica l'attività del database dall'ultima operazione aggressiva di vacuum ed evidenzia il probabile carico di lavoro per i prossimi processi VACUUM.

Cos'è un vuoto aggressivo?

Un'operazione VACUUM aggressiva esegue una scansione completa di tutte le pagine all'interno di una tabella, comprese quelle normalmente saltate durante la normale procedura. VACUUMs Questa scansione approfondita mira a «congelare» le transazioni che IDs si avvicinano alla loro età massima, prevenendo efficacemente una situazione nota come transazione ID wraparound.

Per `postgres_get_av_diag()` segnalare i bloccanti, è necessario che il blocco abbia almeno 500 milioni di transazioni.

Argomenti

- Installazione di strumenti di monitoraggio e diagnostica dell'autovacuum in
-
- Risoluzione dei vacuoblocker identificabili in
-
- Risoluzione dei problemi di prestazioni del vuoto in
- Spiegazione dei messaggi NOTICE in

Installazione di strumenti di monitoraggio e diagnostica dell'autovacuum in

La `postgres_get_av_diag()` funzione è attualmente disponibile nelle seguenti versioni :

- 17.4 e versioni successive (17)
- 16.7 e versioni successive 16
- 15.11 e versioni successive 15
- 14.16 e versioni successive 14
- 13.19 e versioni successive 13

Per utilizzarla`postgres_get_av_diag()`, crea l'`rds_tools`estensione.

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

Verifica che l'estensione sia installata.

```
postgres=> \dx rds_tools
             List of installed extensions
```

```
   Name    | Version |  Schema   |                    Description
 ----------+---------+-----------
+-------------------------------------------------------------
 rds_tools |   1.9   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

Verificate che la funzione sia stata creata.

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

La query deve essere eseguita nel database con l'ID di transazione più vecchio per ottenere risultati accurati. Per ulteriori informazioni sull'utilizzo del database con l'ID di transazione più vecchio, vedi Non connesso al database con l'età dell'ID di transazione più vecchio

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
        rds_tools.postgres_get_av_diag ()
    ORDER BY
        autovacuum_lagging_by DESC) q;
```

La `postgres_get_av_diag()` funzione restituisce una tabella con le seguenti informazioni:

bloccante

Speciifica la categoria di attività del database che blocca il vuoto.

- Dichiarazione attiva
- Inattivo in transazione
- Transazione preparata
- Slot di replica logica
- Istanze Reader
- Tabelle temporanee

database

Specificate il nome del database, se applicabile e supportato. Si tratta del database in cui l'attività è in corso e blocca o bloccherà l'autovacuum. Questo è il database a cui devi connetterti e agire.

blocker_identifier

Speciifica l'identificatore dell'attività che sta bloccando o bloccherà l'autovacuum. L'identificatore può essere un ID di processo insieme a un'istruzione SQL, una transazione preparata, un indirizzo IP di una replica letta e il nome dello slot di replica, logico o fisico.

wait_event

Speciifica l'evento di di attesa della sessione di blocco ed è applicabile ai seguenti bloccanti:

- Dichiarazione attiva
- Inattivo in transazione

autovacum_lagging_by

Specificate il numero di transazioni che autovacuum è in ritardo rispetto al volume di lavoro arretrato per categoria.

suggerimento

Specificate i suggerimenti per risolvere il blocco. Queste istruzioni includono il nome del database in cui esiste l'attività, ove applicabile, l'ID di processo (PID) della sessione, ove applicabile, e l'azione da intraprendere.

azione_suggerita

Suggerisce l'azione da intraprendere per risolvere il blocco.

Risoluzione dei vacuoblocker identificabili in

Autovacuum esegue vuoti aggressivi e riduce l'età della transazione portandola al di sotto della soglia specificata dal parametro dell'istanza RDS. IDs `autovacuum_freeze_max_age` Puoi tenere traccia di questa età utilizzando il CloudWatch parametro `MaximumUsedTransactionIDs` Amazon.

Per trovare l'impostazione di `autovacuum_freeze_max_age` (che ha un valore predefinito di 200 milioni di transazioni IDs) per la tua istanza Amazon RDS, puoi utilizzare la seguente query:

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
    name = 'autovacuum_freeze_max_age';
```

Tieni presente che verifica la presenza di bloccanti del vuoto aggressivi `postgres_get_av_diag()` solo quando l'età supera la soglia di autovuoto adattivo di Amazon RDS di 500 milioni di transazioni. IDs `postgres_get_av_diag()`Per rilevare i bloccanti, il blocker deve avere almeno 500 milioni di transazioni.

La `postgres_get_av_diag()` funzione identifica i seguenti tipi di bloccanti:

Argomenti

- Dichiarazione attiva
- Inattivo in transazione
- Transazione preparata
- Slot di replica logica
- Istanze Reader
- Tabelle temporanee

Dichiarazione attiva

In PostgreSQL, un'istruzione attiva è un'istruzione SQL attualmente eseguita dal database.
Ciò include domande, transazioni o qualsiasi operazione in corso. Durante il monitoraggio tramite`pg_stat_activity`, la colonna dello stato indica che il processo con il PID corrispondente è attivo.

La `postgres_get_av_diag()` funzione visualizza un output simile al seguente quando identifica un'istruzione che è un'istruzione attiva.

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

Azione consigliata

Seguendo le indicazioni riportate nella `suggestion` colonna, l'utente può connettersi al database in cui è presente l'istruzione attiva e, come specificato nella `suggested_action` colonna, è consigliabile esaminare attentamente l'opzione per terminare la sessione. Se l'interruzione è sicura, è possibile utilizzare la `pg_terminate_backend()` funzione per terminare la sessione. Questa azione può essere eseguita da un amministratore (come l'account principale RDS) o da un utente con il privilegio richiesto. `pg_terminate_backend()`

> **Warning**
>
> Una sessione terminata annullerà (`ROLLBACK`) le modifiche apportate. A seconda delle tue esigenze, potresti voler eseguire nuovamente l'istruzione. Tuttavia, si consiglia di eseguire questa operazione solo dopo che il processo di aspirazione automatica ha terminato l'aggressiva operazione di aspirazione.

Inattivo in transazione

Un'istruzione di transazione inattiva si riferisce a qualsiasi sessione che ha aperto una transazione esplicita (ad esempio emettendo una `BEGIN` dichiarazione), eseguito del lavoro e ora è in attesa che il client esegua altro lavoro o segnali la fine della transazione emettendo un `COMMITROLLBACK`, o `END` (il che risulterebbe implicito). `COMMIT`

La `postgres_get_av_diag()` funzione visualizza un output simile al seguente quando identifica un'istruzione come bloccante. `idle in transaction`

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

Azione consigliata

Come indicato nella `suggestion` colonna, è possibile connettersi al database in cui è presente la sessione di transazione inattiva e terminare la sessione utilizzando la `pg_terminate_backend()` funzione. L'utente può essere l'utente amministratore (account principale RDS) o un utente con il privilegio. `pg_terminate_backend()`

> **Warning**
>
> Una sessione terminata annullerà (`ROLLBACK`) le modifiche apportate. A seconda delle tue esigenze, potresti voler eseguire nuovamente l'istruzione. Tuttavia, si consiglia di eseguire questa operazione solo dopo che il processo di aspirazione automatica ha terminato l'aggressiva operazione di aspirazione.

Transazione preparata

PostgreSQL consente transazioni che fanno parte di una strategia di commit in due fasi chiamata transazioni preparate. Queste sono abilitate impostando il `max_prepared_transactions` parametro su un valore diverso da zero. Le transazioni preparate sono progettate per garantire che una transazione sia duratura e rimanga disponibile anche dopo arresti anomali del database, riavvii o disconnessioni del client. Analogamente alle normali transazioni, a esse viene assegnato un ID di transazione e possono influire sull'autovacuum. Se lasciato in uno stato preparato, l'autovacuum non può bloccarsi e può comportare la modifica dell'ID della transazione.

Quando le transazioni vengono lasciate preparate a tempo indeterminato senza essere risolte da un gestore delle transazioni, diventano transazioni preparate orfane. L'unico modo per risolvere questo problema è eseguire il commit o il rollback della transazione utilizzando rispettivamente i `ROLLBACK PREPARED` comandi `COMMIT PREPARED` or.

> **Note**
>
> Tieni presente che un backup eseguito durante una transazione preparata conterrà ancora quella transazione dopo il ripristino. Consulta le seguenti informazioni su come individuare e chiudere tali transazioni.

La `postgres_get_av_diag()` funzione visualizza il seguente output quando identifica un blocker che è una transazione preparata.

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

Azione suggerita

Come indicato nella colonna dei suggerimenti, connettiti al database in cui si trova la transazione preparata. In base alla `suggested_action` colonna, esamina attentamente se eseguire una delle due `COMMIT` o`ROLLBACK`, quindi, l'azione appropriata.

Per monitorare le transazioni preparate in generale, PostgreSQL offre una visualizzazione del catalogo chiamata. `pg_prepared_xacts` È possibile utilizzare la seguente query per trovare le transazioni preparate.

```
SELECT
    gid,
    prepared,
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

Slot di replica logica

Lo scopo di uno slot di replica è conservare le modifiche non utilizzate fino a quando non vengono replicate su un server di destinazione. Per ulteriori informazioni, consulta Replica logica di PostgreSQL.

Esistono due tipi di slot di replica logica.

Slot di replica logica inattivi

Quando la replica viene terminata, i log delle transazioni non utilizzati non possono essere rimossi e lo slot di replica diventa inattivo. Sebbene uno slot di replica logica inattivo non sia attualmente utilizzato da un abbonato, rimane sul server, il che comporta la conservazione dei file WAL e impedisce la rimozione dei vecchi registri delle transazioni. Ciò può aumentare l'utilizzo del disco e in particolare impedire ad autovacuum di ripulire le tabelle interne del catalogo, poiché il sistema deve impedire che le informazioni LSN vengano sovrascritte. Se non viene risolto, ciò può comportare un aumento del volume del catalogo, un peggioramento delle prestazioni e un aumento del rischio di un vuoto incompleto, con conseguenti potenziali interruzioni delle transazioni.

Slot di replica logica attivi ma lenti

A volte la rimozione delle tuple morte del catalogo viene ritardata a causa del peggioramento delle prestazioni della replica logica. Questo ritardo nella replica rallenta l'aggiornamento del catalogo `catalog_xmin` e può portare a un aumento esponenziale del catalogo e a un vuoto avvolgente.

La `postgres_get_av_diag()` funzione visualizza un output simile al seguente quando trova uno slot di replica logico da bloccare.

```
blocker               | Logical replication slot
database              | my_database
blocker_identifier    | slot1
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

Azione consigliata

Per risolvere questo problema, controllate la configurazione di replica per individuare eventuali problemi con lo schema o i dati di destinazione che potrebbero interrompere il processo di applicazione. I motivi più comuni sono i seguenti:

- Colonne mancanti
- Tipo di dati incompatibile
- Mancata corrispondenza dei dati
- Tabella mancante

Se il problema è legato a problemi di infrastruttura:

- Problemi di rete: come posso risolvere i problemi con un database Amazon RDS in uno stato di rete incompatibile? .
- Il database o l'istanza DB non sono disponibili per i seguenti motivi:
  - L'istanza di replica ha esaurito lo storage: consulta lo storage delle istanze database di Amazon RDS esaurito per informazioni sull'aggiunta di storage.
  - Parametri incompatibili: recensione Come posso correggere un'istanza DB di Amazon RDS bloccata nello stato di parametri incompatibili? per ulteriori informazioni su come risolvere il problema.

Se l'istanza è attiva o esterna alla AWS rete AWS EC2, consulta l'amministratore per risolvere i problemi relativi alla disponibilità o all'infrastruttura.

Eliminazione dello slot inattivo

> **Warning**
>
> Attenzione: prima di eliminare uno slot di replica, assicuratevi attentamente che non sia in corso alcuna replica, che sia inattivo e che si trovi in uno stato irreversibile. L'eliminazione prematura di uno slot potrebbe interrompere la replica o causare la perdita di dati.

Dopo aver verificato che lo slot di replica non è più necessario, eliminalo per consentire il proseguimento dell'autovacuum. Questa condizione `active = 'f'` garantisce che venga eliminato solo uno slot inattivo.

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

Istanze Reader

Quando l'impostazione hot_standby_feedback è abilitata, impedisce che autovacuum sull'istanza writer rimuova le righe morte che potrebbero essere ancora necessarie per le query in esecuzione sull'istanza reader. Questo comportamento è necessario perché le query in esecuzione sull'istanza reader (applicabile anche alle istanze di lettura in Aurora Global Database) richiedono che tali righe rimangano disponibili sull'istanza writer, prevenendo conflitti e annullamenti delle query.

> **Note**
>
> `hot_standby_feedback`è abilitato per impostazione predefinita e non modificabile in Aurora PostgreSQL.

La `postgres_get_av_diag()` funzione visualizza un output simile al seguente quando trova una replica di lettura con uno slot di replica fisico come blocco.

```
blocker               | Oldest query running on aurora reader
database              | Not applicable
blocker_identifier    | my-aurora-reader-2
wait_event            | Not applicable
autovacuum_lagging_by | 540,122,859
suggestion            | Run the following query on the reader "my-aurora-reader-2" to
 find the long running query:

                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 523476310;

                      | Review carefully and you may consider terminating the query on
 reader using suggested_action.
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 523476310;","

                      | [OR]


                      | ","Delete the reader if not needed"}
```

Come consigliato nella `suggested_action` colonna, esamina attentamente queste opzioni per sbloccare l'autovacuum.

- Termina la query: seguendo le indicazioni nella colonna dei suggerimenti, puoi connetterti alla replica di lettura, come specificato nella colonna suggested_action. È consigliabile esaminare attentamente l'opzione per terminare la sessione. Se l'interruzione è ritenuta sicura, è possibile utilizzare la funzione per terminare la sessione. `pg_terminate_backend()` Questa azione può essere eseguita da un amministratore (come l'account principale RDS) o da un utente con il privilegio pg_terminate_backend () richiesto.

  È possibile eseguire il seguente comando SQL sulla replica di lettura per terminare la query che impedisce all'aspirapolvere sul primario di ripulire le vecchie righe. Il valore di `backend_xmin` è riportato nell'output della funzione:

  ```
  SELECT
      pg_terminate_backend(pid)
  FROM
      pg_catalog.pg_stat_activity
  WHERE
      backend_xmin::text::bigint = backend_xmin;
  ```

- Eliminare le istanze del lettore se non è più necessaria: se l'istanza del lettore non è più necessaria, è possibile eliminarla. Ciò rimuoverà il sovraccarico di replica associato e consentirà all'istanza principale di riciclare i log delle transazioni senza essere ostacolato dall'istanza.

Tabelle temporanee

Le tabelle temporanee, create utilizzando la `TEMPORARY` parola chiave, risiedono nello schema temporaneo, ad esempio pg_temp_xxx, e sono accessibili solo alla sessione che le ha create. Le tabelle temporanee vengono eliminate al termine della sessione. Tuttavia, queste tabelle sono invisibili al processo di autovacuum di PostgreSQL e devono essere cancellate manualmente dalla sessione che le ha create. Il tentativo di aspirare la tabella temporanea da un'altra sessione non ha alcun effetto.

In circostanze insolite, una tabella temporanea esiste senza che una sessione attiva ne sia proprietaria. Se la sessione proprietaria termina inaspettatamente a causa di un arresto anomalo irreversibile, di un problema di rete o di un evento simile, è possibile che la tabella temporanea non venga ripulita, lasciandola come tabella «orfana». Quando il processo autovacuum di PostgreSQL rileva una tabella temporanea orfana, registra il seguente messaggio:

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

La `postgres_get_av_diag()` funzione visualizza un output simile al seguente quando identifica una tabella temporanea come bloccante. Affinché la funzione mostri correttamente l'output relativo alle tabelle temporanee, deve essere eseguita all'interno dello stesso database in cui esistono tali tabelle.

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

Azione consigliata

Segui le istruzioni fornite nella `suggestion` colonna dell'output per identificare e rimuovere la tabella temporanea che impedisce l'esecuzione di autovacuum. Utilizzate il comando seguente per eliminare la tabella temporanea riportata da. `postgres_get_av_diag()` Sostituisci il nome della tabella in base all'output fornito dalla `postgres_get_av_diag()` funzione.

```
DROP TABLE my_temp_schema.my_temp_table;
```

La seguente query può essere utilizzata per identificare le tabelle temporanee:

```
SELECT
    oid,
    relname,
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

Questa sezione esplora altri motivi che possono impedire che l'aspirapolvere compia progressi. Questi problemi non sono attualmente identificabili direttamente dalla funzione. `postgres_get_av_diag()`

Argomenti

- Incoerenza dell'indice
- Tasso di transazione eccezionalmente elevato

Incoerenza dell'indice

Un indice logicamente incoerente può impedire che l'autovacuum compia progressi. I seguenti errori o errori simili vengono registrati durante la fase di vuoto dell'indice o quando si accede all'indice tramite istruzioni SQL.

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

Guida

Ricostruisci l'indice o salta gli indici utilizzando il manuale. `INDEX_CLEANUP VACUUM FREEZE`

- Utilizzo dell'opzione CONCURRENTLY — Prima della versione 12 di PostgreSQL, la ricostruzione di un indice richiedeva un blocco tabella esclusivo, che limitava l'accesso alla tabella. Con PostgreSQL versione 12 e versioni successive, l'opzione CONCURRENTLY consente il blocco a livello di riga, migliorando significativamente la disponibilità della tabella. Di seguito è riportato il comando:

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  Anche se CONCURRENTLY comporta meno interruzioni, può essere più lenta sui tavoli occupati. Se possibile, valuta la possibilità di creare l'indice durante i periodi di traffico ridotto. Per ulteriori informazioni, vedere REINDEX nella documentazione di PostgreSQL.
- Utilizzo dell'opzione INDEX_CLEANUP FALSE: se gli indici sono grandi e si stima che richiedano molto tempo per essere completati, puoi sbloccare l'autovacuum eseguendo un VACUUM FREEZE

manuale escludendo gli indici. Questa funzionalità è disponibile nella versione 12 di PostgreSQL e nelle versioni successive.

Bypassare gli indici ti consentirà di saltare il processo di vacuità dell'indice incoerente e di mitigare il problema dell'involucro. Tuttavia, ciò non risolverà il problema sottostante relativo alla pagina non valida. Per risolvere completamente il problema relativo alla pagina non valida, dovrai comunque ricostruire l'indice.

Tasso di transazione eccezionalmente elevato

In PostgreSQL, tassi di transazione elevati possono influire in modo significativo sulle prestazioni di autovacuum, portando a una pulizia più lenta delle tuple morte e a un aumento del rischio di avvolgimento dell'ID delle transazioni. È possibile monitorare il tasso di transazione misurando la differenza tra due periodi di tempo, in genere al secondo. `max(age(datfrozenxid))` Inoltre, puoi utilizzare le seguenti metriche dei contatori di RDS Performance Insights per misurare il tasso di transazione (la somma di xact_commit e xact_rollback) che è il numero totale di transazioni.

Contatore	Tipo	Unità	Parametro
xact_commit	Transazioni	Commit al secondo	db.Transactions.xact_commit
xact_rollback	Transazioni	Rollback al secondo	db.Transactions.xact_rollback

Un aumento rapido indica un carico di transazioni elevato, che può sovraccaricare l'autovacuum, con conseguenti ingrossamenti, conflitti tra blocchi e potenziali problemi di prestazioni. Ciò può influire negativamente sul processo di autovacuum in un paio di modi:

- Attività delle tabelle: la tabella specifica oggetto di cancellazione potrebbe registrare un volume elevato di transazioni, con conseguenti ritardi.
- Risorse di sistema L'intero sistema potrebbe essere sovraccarico, rendendo difficile per Autovacuum l'accesso alle risorse necessarie per funzionare in modo efficiente.

Considerate le seguenti strategie per consentire all'autovacuum di funzionare in modo più efficace e di tenere il passo con le sue attività:

1. Se possibile, riduci il tasso di transazione. Prendi in considerazione la possibilità di raggruppare o raggruppare transazioni simili, ove possibile.
2. Scegli come target le tabelle aggiornate di frequente con `VACUUM FREEZE` operazioni manuali ogni notte, settimanalmente o bisettimanalmente durante le ore non di punta.
3. Valuta la possibilità di ampliare la classe di istanza per allocare più risorse di sistema per gestire l'elevato volume di transazioni e l'autovacuum.

Risoluzione dei problemi di prestazioni del vuoto in

Questa sezione illustra i fattori che spesso contribuiscono a rallentare le prestazioni del vuoto e come risolvere questi problemi.

Argomenti

- Aspirare indici di grandi dimensioni
- Troppe tabelle o database da archiviare
- È in funzione un aspirapolvere aggressivo (per evitare che si avvolga)

Aspirare indici di grandi dimensioni

VACUUM opera attraverso fasi sequenziali: inizializzazione, scansione degli heap, vacuazione degli indici e degli heap, pulizia degli indici, troncamento dell'heap e pulizia finale. Durante la scansione dell'heap, il processo elimina le pagine, le deframmenta e le congela. Dopo aver completato la scansione dell'heap, VACUUM pulisce gli indici, restituisce le pagine vuote al sistema operativo ed esegue le attività di pulizia finali come la pulizia della mappa dello spazio libero e l'aggiornamento delle statistiche.

L'aspirazione dell'indice può richiedere più passaggi quando `maintenance_work_mem` (o) è insufficiente per elaborare l'indice. `autovacuum_work_mem` In PostgreSQL 16 e versioni precedenti, un limite di memoria di 1 GB per l'archiviazione di IDs tuple non funzionanti spesso imponeva passaggi multipli su indici di grandi dimensioni. `TidStore`Presenta PostgreSQL 17, che alloca la memoria in modo dinamico anziché utilizzare un array a allocazione singola. Ciò rimuove il vincolo di 1 GB, utilizza la memoria in modo più efficiente e riduce la necessità di più scansioni dell'indice per ogni indice.

Gli indici di grandi dimensioni possono comunque richiedere più passaggi in PostgreSQL 17 se la memoria disponibile non è in grado di supportare l'elaborazione dell'intero indice

contemporaneamente. In genere, gli indici più grandi contengono più tuple morte che richiedono più passaggi.

Rilevamento di operazioni di aspirazione lente

La `postgres_get_av_diag()` funzione è in grado di rilevare quando le operazioni di aspirazione funzionano lentamente a causa di una memoria insufficiente. Per ulteriori informazioni su questa funzione, vedereInstallazione di strumenti di monitoraggio e diagnostica dell'autovacuum in .

La `postgres_get_av_diag()` funzione emette i seguenti avvisi quando la memoria disponibile non è sufficiente per completare lo svuotamento dell'indice in un solo passaggio.

`rds_tools`1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
       memory. The suggested value is an estimate based on the current number of dead
 tuples for the table being vacuumed, which might not fully reflect the latest state.
 Additionally, review the possibility of manual
       vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
 TRUE) table_name;). For more information, see

       Working with PostgreSQL autovacuum in the Amazon Aurora User Guide.
```

> **Note**
>
> La `postgres_get_av_diag()` funzione si basa sulla stima della quantità di memoria richiesta `pg_stat_all_tables.n_dead_tup` per l'aspirazione degli indici.

Quando la `postgres_get_av_diag()` funzione identifica un'operazione di vuoto lenta che richiede più scansioni dell'indice perché insufficienteautovacuum_work_mem, genera il seguente messaggio:

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
```

```
        For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

Guida

È possibile applicare le seguenti soluzioni alternative utilizzando il manuale `VACUUM FREEZE` per accelerare il congelamento della tabella.

Aumenta la memoria per l'aspirazione

Come suggerito dalla `postgres_get_av_diag()` funzione, è consigliabile aumentare il `autovacuum_work_mem` parametro per risolvere potenziali vincoli di memoria a livello di istanza. Sebbene `autovacuum_work_mem` sia un parametro dinamico, è importante notare che affinché la nuova impostazione di memoria abbia effetto, il demone autovacuum deve riavviare i suoi worker. Per eseguire questa operazione:

1. Conferma che la nuova impostazione sia attiva.
2. Termina i processi che attualmente eseguono autovacuum.

Questo approccio garantisce che l'allocazione di memoria modificata venga applicata alle nuove operazioni di autovacuum.

Per risultati più immediati, prendi in considerazione l'esecuzione manuale di un'`VACUUM FREEZE`operazione con un'`maintenance_work_mem`impostazione maggiore all'interno della sessione:

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Se utilizzi Amazon RDS e ritieni di aver bisogno di memoria aggiuntiva per supportare valori più elevati`autovacuum_work_mem`, `maintenance_work_mem` oppure prendi in considerazione l'aggiornamento a una classe di istanze con più memoria. Ciò può fornire le risorse necessarie per migliorare le operazioni di aspirazione manuali e automatiche, con conseguente miglioramento delle prestazioni complessive del vuoto e del database.

Disattiva INDEX_CLEANUP

Il manuale `VACUUM` in PostgreSQL versione 12 e successive consente di saltare la fase di pulizia dell'indice, mentre l'autovacuum di emergenza in PostgreSQL versione 14 e successive lo fa automaticamente in base al parametro. `vacuum_failsafe_age`

> **Warning**
>
> Saltare la pulizia dell'indice può causare un aumento esponenziale dell'indice e influire negativamente sulle prestazioni delle query. Per ovviare a questo problema, prendete in considerazione la possibilità di reindicizzare o cancellare gli indici interessati durante una finestra di manutenzione.

Per ulteriori indicazioni sulla gestione di indici di grandi dimensioni, consulta la documentazione su. Gestione di autovacuum con indici di grandi dimensioni

Aspirazione a indice parallelo

A partire da PostgreSQL 13, gli indici possono essere aspirati e puliti in parallelo per impostazione predefinita utilizzando il metodo `VACUUM` manuale, con un processo vacuum worker assegnato a ciascun indice. Tuttavia, per consentire a PostgreSQL di determinare se un'operazione sotto vuoto è idonea per l'esecuzione parallela, devono essere soddisfatti criteri specifici:

- Devono essere presenti almeno due indici.
- Il `max_parallel_maintenance_workers` parametro deve essere impostato almeno su 2.
- La dimensione dell'indice deve superare il `min_parallel_index_scan_size` limite, che per impostazione predefinita è 512 KB.

Puoi modificare l'`max_parallel_maintenance_workers`impostazione in base al numero di v CPUs disponibili sulla tua istanza Amazon RDS e al numero di indici sulla tabella per ottimizzare i tempi di consegna dell'aspirapolvere.

Per ulteriori informazioni, consulta Parallel vacuuming in Amazon RDS for PostgreSQL e Amazon Aurora PostgreSQL.

Troppe tabelle o database da archiviare

Come indicato nella documentazione The Autovacuum Daemon di PostgreSQL, il demone autovacuum opera attraverso più processi. Ciò include un programma di avvio automatico persistente responsabile dell'avvio dei processi di lavoro di autovacuum per ogni database all'interno del sistema. Il programma di avvio pianifica l'avvio di questi worker all'incirca ogni secondo per database. `autovacuum_naptime`

Con i database 'N', un nuovo lavoratore inizia all'incirca ogni [`autovacuum_naptime`/N secondi]. Tuttavia, il numero totale di lavoratori simultanei è limitato dall'`autovacuum_max_workers`impostazione. Se il numero di database o tabelle che richiedono l'aspirazione supera questo limite, il database o la tabella successiva verrà elaborata non appena un lavoratore sarà disponibile.

Quando è necessario pulire contemporaneamente molte tabelle o database di grandi dimensioni, tutti gli aspiratori automatici disponibili possono rimanere occupati per un periodo prolungato, ritardando la manutenzione di altre tabelle e database. In ambienti con tassi di transazione elevati, questo collo di bottiglia può aggravarsi rapidamente e potenzialmente portare a problemi di vuoto all'interno dell'istanza Amazon RDS.

Quando `postgres_get_av_diag()` rileva un numero elevato di tabelle o database, fornisce la seguente raccomandazione:

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

Guida

Aumenta autovacuum_max_workers

Per velocizzare l'aspirazione, consigliamo di regolare il parametro per consentire l'utilizzo simultaneo di più aspiratori. `autovacuum_max_workers` Se i problemi di prestazioni persistono, prendi in considerazione la possibilità di scalare la tua istanza Amazon RDS a una classe con più CPUs v, che può migliorare ulteriormente le capacità di elaborazione parallela.

È in funzione un aspirapolvere aggressivo (per evitare che si avvolga)

L'età del database (MaximumUsedTransactionIDs) in PostgreSQL diminuisce solo quando un vuoto aggressivo (per evitare il wraparound) viene completato con successo. Fino alla fine di questo vuoto, l'età continuerà ad aumentare a seconda del tasso di transazione.

La `postgres_get_av_diag()` funzione genera quanto segue `NOTICE` quando rileva un vuoto aggressivo. Tuttavia, attiva questa uscita solo dopo che il vuoto è rimasto attivo per almeno due minuti.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Per ulteriori informazioni sull'aspirapolvere aggressivo, vedere Quando un aspirapolvere aggressivo è già in funzione.

È possibile verificare se è in corso un aspirapolvere aggressivo con la seguente interrogazione:

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
    pg_stat_activity a
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

Puoi determinare se si tratta di un vuoto aggressivo (per evitare che si avvolga) controllando la colonna di interrogazione nell'output. La frase «per prevenire l'avvolgimento» indica che si tratta di un vuoto aggressivo.

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

Ad esempio, supponiamo di avere un blocker all'età di 1 miliardo di anni di età delle transazioni e una tabella che richieda un vuoto aggressivo per evitare che la transazione si concluda alla stessa età delle transazioni. Inoltre, esiste un altro blocco all'età di 750 milioni di anni di età delle transazioni. Dopo aver eliminato il blocco all'età di 1 miliardo di anni, l'età delle transazioni non scenderà immediatamente a 750 milioni. Rimarrà elevata fino al completamento della tabella che richiede un vuoto aggressivo o fino al completamento di qualsiasi transazione con un fatturato superiore a 750 milioni. Durante questo periodo, l'età delle transazioni del cluster PostgreSQL continuerà a crescere. Una volta completato il processo di sottovuoto, l'età delle transazioni scenderà a 750 milioni, ma ricomincerà ad aumentare fino al termine di un'ulteriore fase di sottovuoto. Questo ciclo continuerà finché persistono queste condizioni, fino a quando l'età delle transazioni non scenderà al livello configurato per l'istanza Amazon RDS, specificato da. `autovacuum_freeze_max_age`

Spiegazione dei messaggi NOTICE in

La funzione fornisce i seguenti messaggi NOTICE: `postgres_get_av_diag()`

Quando l'età non ha ancora raggiunto la soglia di monitoraggio

La soglia di monitoraggio per `postgres_get_av_diag()` identificare i bloccanti è di 500 milioni di transazioni per impostazione predefinita. Se `postgres_get_av_diag()` genera il seguente AVVISO, indica che l'età della transazione non ha ancora raggiunto questa soglia.

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

Non connesso al database con l'età dell'ID di transazione più vecchio

La `postgres_get_av_diag()` funzione fornisce l'output più preciso quando è connessa al database con l'ID di transazione più vecchio. Nel tuo caso, il database con l'ID di transazione più vecchio riportato `postgres_get_av_diag()` sarà diverso da «my_database». Se non sei connesso al database corretto, viene generato il seguente AVVISO:

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

La connessione al database con la data di transazione più vecchia è importante per i seguenti motivi:

- Identificazione dei blocchi temporanei delle tabelle: poiché i metadati per le tabelle temporanee sono specifici di ogni database, in genere si trovano nel database in cui vengono creati. Tuttavia, se una tabella temporanea è il blocco principale e si trova nel database con la transazione più vecchia, ciò potrebbe essere fuorviante. La connessione al database corretto garantisce l'identificazione accurata del blocco temporaneo delle tabelle.
- Diagnosi dei vuoti lenti: i metadati degli indici e le informazioni sul conteggio delle tabelle sono specifici del database e sono necessari per diagnosticare i problemi relativi al vuoto lento.

Il database con la transazione più vecchia per età si trova su un database rdsadmin o template0

In alcuni casi, il `template0` database `rdsadmin` o può essere identificato come il database con l'ID di transazione più vecchio (età). In tal caso, `postgres_get_av_diag()` emetterà il seguente AVVISO:

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

Verifica che il blocco elencato non provenga da nessuno di questi due database. Se viene segnalata la presenza del blocco in uno dei due `rdsadmintemplate0`, contatta l'assistenza poiché questi database non sono accessibili agli utenti e richiedono un intervento.

È altamente improbabile che il `template0` database `rdsadmin` o il database contenga un blocco principale.

Quando è già in funzione un aspirapolvere aggressivo

La `postgres_get_av_diag()` funzione è progettata per segnalare quando è in corso un processo di aspirazione aggressivo, ma attiva questa uscita solo dopo che il vuoto è rimasto attivo per almeno 1 minuto. Questo ritardo intenzionale aiuta a ridurre le possibilità di falsi positivi. Attendendo, la funzione assicura che vengano segnalati solo i vuoti efficaci e significativi, permettendo un monitoraggio più accurato e affidabile dell'attività del vuoto.

La `postgres_get_av_diag()` funzione genera il seguente AVVISO quando rileva uno o più aspirapolvere aggressivi in corso.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Come indicato nell'AVVISO, continuate a monitorare le prestazioni del vuoto. Per ulteriori informazioni sul vuoto aggressivo, vedere È in funzione un aspirapolvere aggressivo (per evitare che si avvolga)

Quando l'aspirapolvere automatico è spento

La `postgres_get_av_diag()` funzione genera il seguente AVVISO se l'autovacuum è disabilitato sull'istanza del database:

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

Autovacuum è una funzionalità fondamentale dell' del database. Rimuove automaticamente le vecchie versioni di riga, recupera spazio di archiviazione e previene il sovraccarico delle tabelle, contribuendo a mantenere tabelle e indici efficienti per prestazioni ottimali. Inoltre, protegge dall'alterazione dell'ID delle transazioni, che può interrompere le transazioni sulla tua istanza Amazon RDS. La disabilitazione dell'autovacuum può portare a un calo a lungo termine delle prestazioni e della stabilità del database. Ti consigliamo di tenerlo sempre acceso. Per ulteriori informazioni, consulta Understanding autovacuum in .

> **Note**
>
> La disattivazione dell'aspirapolvere automatico non blocca gli aspirapolvere aggressivi. Questi problemi continueranno a verificarsi quando i tavoli raggiungeranno la soglia. `autovacuum_freeze_max_age`

Il numero di transazioni rimanenti è estremamente basso

La `postgres_get_av_diag()` funzione genera il seguente AVVISO quando è imminente un aspirapolvere avvolgente. Questo AVVISO viene emesso quando l'istanza Amazon RDS è a 100 milioni di transazioni dal potenziale rifiuto di nuove transazioni.

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

È necessaria un'azione immediata per evitare tempi di inattività del database. È necessario monitorare attentamente le operazioni di svuotamento e prendere in considerazione l'avvio manuale di un file `VACUUM FREEZE` sul database interessato per evitare errori nelle transazioni.

Utilizzo di Babelfish per Aurora PostgreSQL

Babelfish per Aurora PostgreSQL estende il cluster database Aurora PostgreSQL con la possibilità di accettare connessioni al database dai client SQL Server. Con Babelfish, le applicazioni create originariamente per SQL Server possono utilizzare direttamente Aurora PostgreSQL con poche modifiche al codice rispetto a una migrazione tradizionale e senza modificare i driver del database. Per ulteriori informazioni sulla migrazione, consulta Migrazione di un database SQL Server a Babelfish per Aurora Postgre SQL.

Babelfish fornisce un endpoint aggiuntivo per un cluster di database Aurora PostgreSQL che consente di comprendere il protocollo wire-level di SQL Server e le istruzioni SQL Server comunemente utilizzate. Le applicazioni client che utilizzano il protocollo wire Tabular Data Stream (TDS) possono connettersi in maniera nativa alla porta del listener TDS su Aurora PostgreSQL. Per ulteriori informazioni su TDS, consulta [MS-TDS]: Tabular Data Stream Protocol sul sito Web di Microsoft.

> **Note**
>
> Babelfish per Aurora PostgreSQL supporta versioni TDS da 7.1 a 7.4.

Babelfish fornisce inoltre l'accesso ai dati utilizzando la connessione PostgreSQL. Per impostazione predefinita, entrambi i dialetti SQL supportati da Babelfish sono disponibili tramite i protocolli wire nativi presso le seguenti porte:

- Dialetto SQL Server (T-SQL), i client si connettono alla porta 1433.
- Dialetto PostgreSQL (PL/pgSQL), i client si connettono alla porta 5432.

Babelfish esegue il linguaggio Transact-SQL (T-SQL) con alcune differenze. Per ulteriori informazioni, consulta Differenze tra Babelfish per Aurora PostgreSQL e SQL Server.

Nelle seguenti sezioni sono disponibili informazioni sulla configurazione e l'utilizzo di un cluster database Babelfish per Aurora PostgreSQL.

Argomenti

- Limitazioni di Babelfish
- Comprendere l'architettura e la configurazione di Babelfish

- Creazione di un cluster Babelfish per Aurora Postgre DB SQL
- Migrazione di un database SQL Server a Babelfish per Aurora Postgre SQL
- Autenticazione del database con Babelfish per Aurora Postgre SQL
- Connessione a un cluster database Babelfish
- Utilizzo di Babelfish
- Risoluzione dei problemi relativi a Babelfish
- Disattivazione di Babelfish
- Gestione degli aggiornamenti di versione di Babelfish per Aurora Postgre SQL
- Babelfish per Aurora Postgree SQL

Limitazioni di Babelfish

Le seguenti limitazioni si applicano attualmente a Babelfish per Aurora PostgreSQL:

- Babelfish attualmente non supporta le seguenti funzionalità di Aurora:
  - AWS Identity and Access Management
  - Flussi di attività di database (DAS)
  - API dati RDS con Aurora PostgreSQL Serverless v2 e provisioning
  - Server proxy per RDS con RDS per SQL Server
  - SCRAM (Salted Challenge Response Authentication Mechanism)
  - Editor della query
- Babelfish non fornisce il seguente supporto API per i driver client:
  - Le richieste API con gli attributi di connessione correlati a Microsoft Distributed Transaction Coordinator (MSDTC) non sono supportate. Queste includono le chiamate XA effettuate dalla classe nel driver JDBC del server SQL. SQLServer XAResource
- Babelfish attualmente non supporta le seguenti estensioni Aurora PostgreSQL:
  - `bloom`
  - `btree_gin`
  - `btree_gist`
  - `citext`
  - `cube`
  - `hstore`
  - `hypopg`
  - Replica logica con `pglogical`
  - `ltree`
  - `pgcrypto`
  - Gestione del piano di interrogazione tramite `apg_plan_mgmt`

  Per ulteriori informazioni sulle estensioni PostgreSQL, consulta. Utilizzo di estensioni e wrapper di dati esterni
- Il driver jTDS open source progettato come alternativa al driver Microsoft JDBC non è supportato.

Comprendere l'architettura e la configurazione di Babelfish

La gestione del cluster database Aurora PostgreSQL-Compatible Edition che esegue Babelfish è simile a quella di qualsiasi cluster database Aurora. Ovvero, è possibile sfruttare la scalabilità, l'elevata disponibilità con il supporto per il failover e la replica integrata forniti da un cluster database Aurora. Per ulteriori informazioni su queste funzionalità, consulta Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora, Elevata disponibilità di Amazon Aurora e Replica con Amazon Aurora. Hai anche accesso a molti altri AWS strumenti e utilità, tra cui:

- Amazon CloudWatch è un servizio di monitoraggio e osservabilità che ti fornisce dati e approfondimenti utilizzabili. Per ulteriori informazioni, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.
- Performance Insights è una funzionalità di ottimizzazione e monitoraggio delle prestazioni del database che consente di valutare rapidamente il carico sul database. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- I database globali di Aurora si estendono su più database Regioni AWS, abilitando letture globali a bassa latenza e fornendo un ripristino rapido da rare interruzioni che potrebbero interessare un intero sistema. Regione AWS Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.
- L'applicazione automatica delle patch software mantiene il database aggiornato sulle ultime patch up-to-date di sicurezza e funzionalità non appena disponibili.
- Gli eventi Amazon RDS comunicano via e-mail o SMS eventi importanti del database, ad esempio un failover automatico. Per ulteriori informazioni, consulta .

Di seguito sono fornite ulteriori informazioni sull'architettura Babelfish e su come i database SQL Server migrati vengono gestiti da Babelfish. Quando crei il cluster database Babelfish, devi prendere alcune decisioni in anticipo su database singoli o multipli, regole di confronto e altri dettagli.

Argomenti

- Architettura babelfish
- Impostazioni del gruppo di parametri del cluster database per Babelfish
- Comprendere le collazioni in Babelfish per Aurora Postgre SQL
- Gestione degli errori di Babelfish con escape hatch

Architettura babelfish

Quando crei un cluster Aurora PostgreSQL con Babelfish attivato, Aurora fornisce il cluster un database PostgreSQL denominato `babelfish_db`. Questo database è dove risiedono tutti gli oggetti e le strutture di SQL Server migrati.

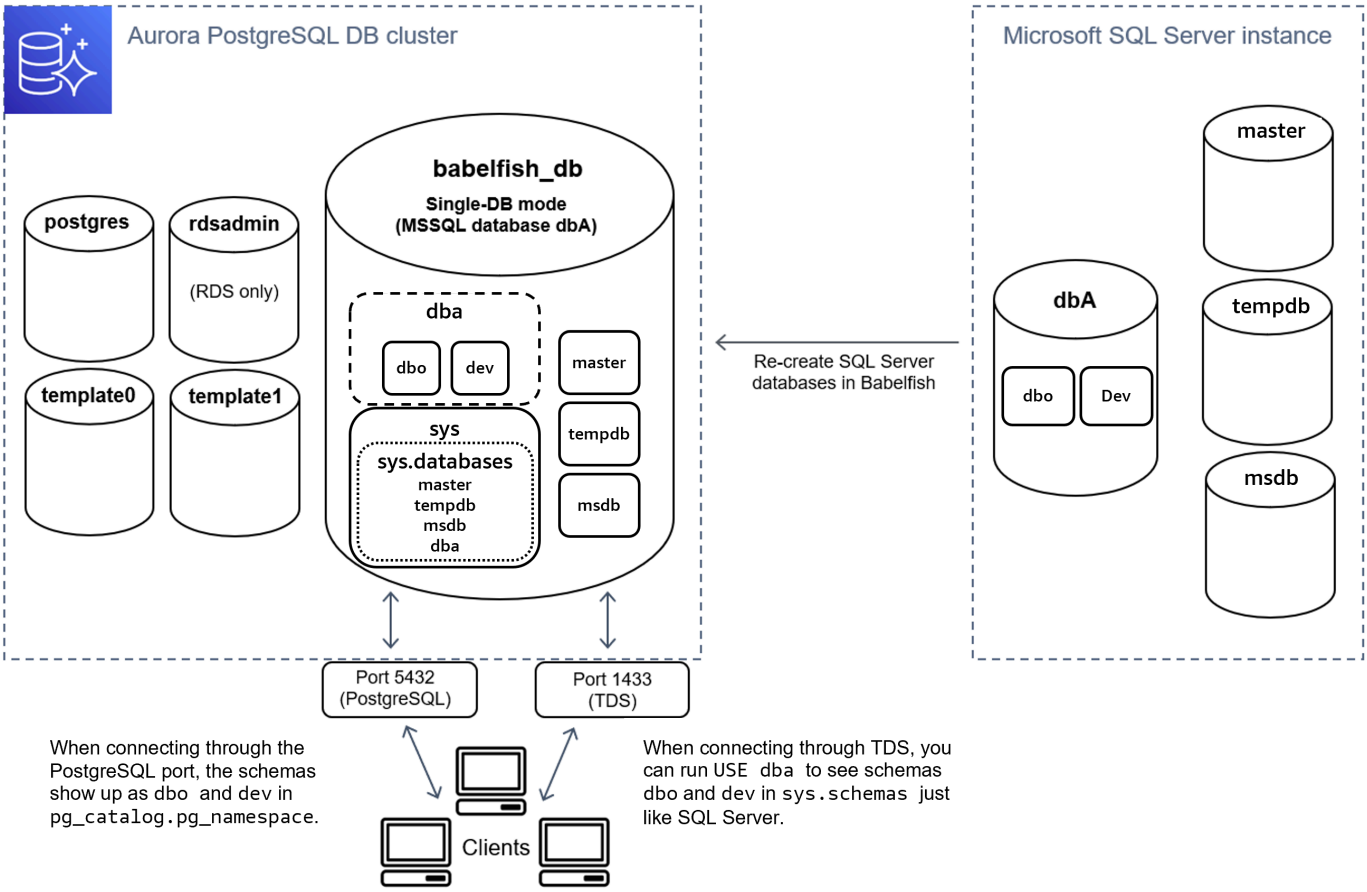
> **Note**
>
> In un cluster Aurora PostgreSQL, il nome del database `babelfish_db` è riservato a Babelfish. La creazione del proprio database "babelfish_db" su un cluster database Babelfish impedisce ad Aurora di eseguire correttamente il provisioning di Babelfish.

Quando ci si connette alla porta TDS, la sessione viene inserita nel database `babelfish_db`. Da T-SQL, la struttura sembra simile alla connessione a un'istanza di SQL Server. Puoi visualizzare i database `master`, `msdb` e `tempdb` e il catalogo `sys.databases`. È possibile creare database utente aggiuntivi e passare da un database all'altro con l'istruzione USE. Quando crei un database utente di SQL Server, viene appiattito nel Database PostgreSQL `babelfish_db`. Il database mantiene la sintassi e la semantica tra database uguali o simili a quelli forniti da SQL Server.

Utilizzo di Babelfish con un singolo database o più database

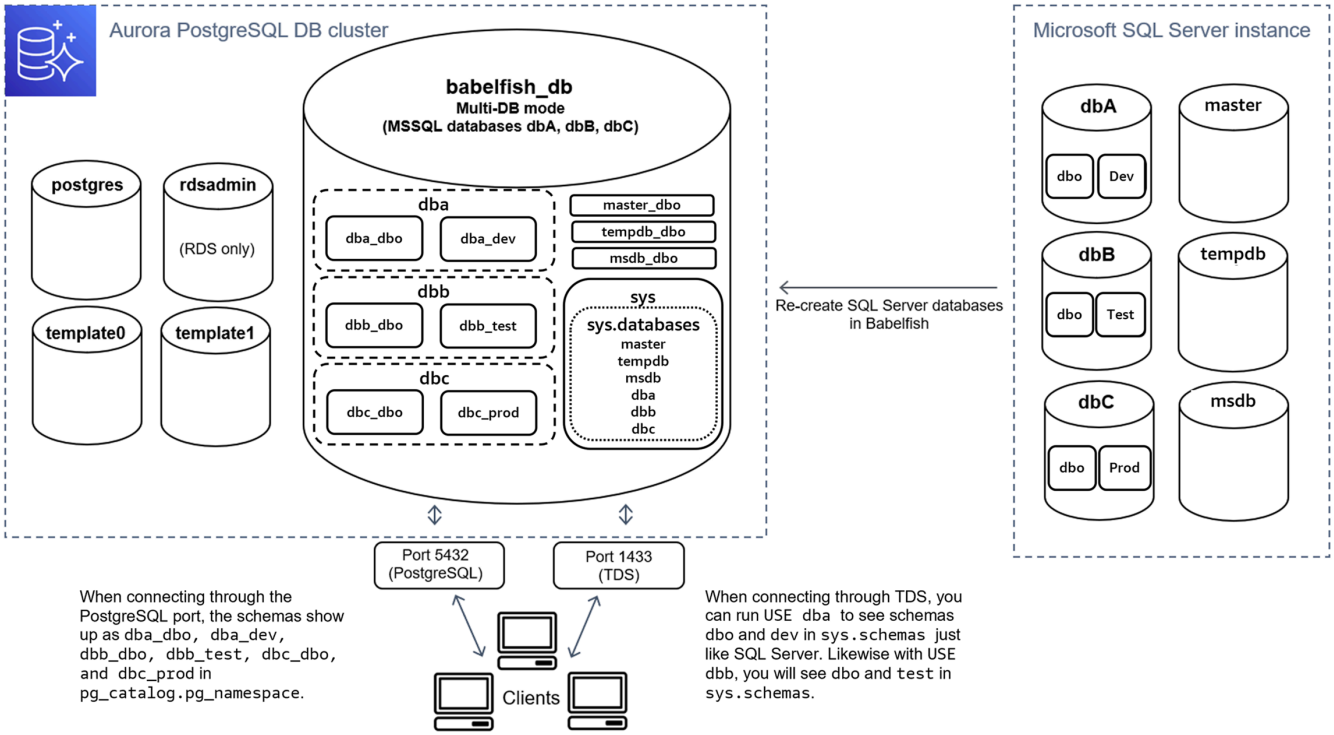
Quando si crea un cluster Aurora PostgreSQL da utilizzare con Babelfish, è possibile scegliere tra l'utilizzo di un singolo database SQL Server sul proprio database SQL Server o più database SQL Server insieme. La tua scelta influisce sul modo in cui i nomi degli schemi di SQL Server all'interno del database `babelfish_db` vengono visualizzati da Aurora PostgreSQL. La modalità di migrazione è memorizzata nel parametro `migration_mode`. Non è necessario modificare questo parametro dopo aver creato il cluster poiché si potrebbe perdere l'accesso a tutti gli oggetti SQL creati in precedenza.

In modalità database singolo, i nomi dello schema del database SQL Server rimangono uguali nel database `babelfish_db` di PostgreSQL. Se scegli di migrare un solo singolo database, i nomi dello schema del database dell'utente migrato possono essere usati come riferimento in PostgreSQL con gli stessi nomi utilizzati in SQL Server. Ad esempio, gli schemi `dbo` e `smith` risiedono all'interno del database `dbA` .



Quando ci si connette tramite TDS, è possibile eseguire `USE dba` per vedere gli schemi dbo e dev da T-SQL, come si farebbe in SQL Server. I nomi dello schema invariati sono visibili anche da PostgreSQL.

In modalità di database multipli, i nomi dello schema dei database utente diventano `dbname_schemaname` se l'accesso viene eseguito da PostgreSQL. I nomi dello schema rimangono gli stessi se l'accesso viene eseguito da T-SQL.



Come mostrato nell'immagine, la modalità database multipli e la modalità database singolo sono le stesse di SQL Server durante la connessione tramite la porta TDS e l'utilizzo di T-SQL. Ad esempio, `USE dbA` elenca schemi `dbo` e `dev` proprio come in SQL Server. I nomi dello schema mappati, come `dba_dbo` e `dba_dev`, sono visibili da PostgreSQL.

Ogni database contiene ancora i tuoi schemi. Il nome di ogni database è preceduto al nome dello schema di SQL Server, utilizzando un trattino di sottolineatura come delimitatore, ad esempio:

- `dba` contiene `dba_dbo` e `dba_dev`.
- `dbb` contiene `dbb_dbo` e `dbb_test`.
- `dbc` contiene `dbc_dbo` e `dbc_prod`.

All'interno del database`babelfish_db`, l'utente T-SQL deve ancora eseguire `USE dbname` per modificare il contesto del database, in modo che l'aspetto rimanga simile a SQL Server.

Scegliere una modalità di migrazione

Ciascuna modalità di migrazione presenta vantaggi e svantaggi. Scegli la modalità di migrazione in base al numero di database utente e ai piani di migrazione. Dopo aver creato un cluster da utilizzare con Babelfish, non devi modificare la modalità di migrazione poiché potresti perdere l'accesso a tutti

gli oggetti SQL creati in precedenza. Quando scegli una modalità di migrazione, considera i requisiti dei database utente e dei client.

Quando crei un cluster da utilizzare con Babelfish, Aurora PostgreSQL crea i database di sistema, `master` e `tempdb`. Se sono stati creati o modificati oggetti nei database di sistema (`master` o `tempdb`), assicurati di ricreare questi oggetti nel nuovo cluster. A differenza di SQL Server, Babelfish non si reinizializza `tempdb` dopo il riavvio di un cluster.

Utilizzare la modalità di migrazione di database singolo nei seguenti casi:

- Se si sta eseguendo la migrazione di un singolo database SQL Server. In modalità database singolo, se l'accesso viene eseguito da PostgreSQL i nomi degli schemi migrati sono identici ai nomi dello schema originale di SQL Server. Ciò riduce le modifiche al codice per le query SQL esistenti se desideri ottimizzarle per l'esecuzione con una connessione PostgreSQL.
- Se il tuo obiettivo finale è una migrazione completa alla nativa Aurora PostgreSQL. Prima di eseguire la migrazione, consolidare gli schemi in un unico schema (`dbo`) e quindi migrare in un singolo cluster per ridurre le modifiche richieste.

Utilizzare la modalità di migrazione di più database nei seguenti casi:

- Se desideri l'esperienza SQL Server predefinita con più database utente nella stessa istanza.
- Se è necessario migrare insieme più database dell'utente.

Impostazioni del gruppo di parametri del cluster database per Babelfish

Quando crei un cluster database Aurora PostgreSQL e scegli Turn on Babelfish (Attiva Babelfish), un gruppo di parametri del cluster database viene creato automaticamente se scegli Create new (Crea nuovo). Questo gruppo di parametri del cluster database si basa sul gruppo di parametri del cluster database Aurora PostgreSQL per la versione di Aurora PostgreSQL scelta per l'installazione, ad esempio Aurora PostgreSQL versione 14. Viene chiamato utilizzando il seguente modello generale:

```
custom-aurora-postgresql14-babelfish-compat-3
```

Durante il processo di creazione del cluster puoi modificare le seguenti impostazioni, ma alcune di queste non possono essere modificate dopo che sono state memorizzate nel gruppo di parametri personalizzati, quindi scegli attentamente:

- Database singolo o database multipli
- Impostazioni locali delle regole di confronto predefinite
- Nome della regola di confronto
- DB parameter group (Gruppo di parametri database)

Per utilizzare un gruppo di parametri esistente del cluster database Aurora PostgreSQL versione 13 o successive, modifica il gruppo e imposta il parametro `babelfish_status` su on. Specificare tutte le opzioni Babelfish prima di creare il cluster Aurora PostgreSQL. Per ulteriori informazioni, consulta .

I seguenti parametri controllano le preferenze di Babelfish. Salvo laddove diversamente indicato nella descrizione, i parametri sono modificabili. Il valore predefinito è incluso nella descrizione. Per visualizzare i valori consentiti per qualsiasi parametro, procedi come segue:

> **Note**
>
> Quando si associa un nuovo gruppo parametri del database a un'istanza database, i parametri statici e dinamici modificati vengono applicati solo dopo il riavvio dell'istanza database. Tuttavia, se modifichi i parametri dinamici nel gruppo di parametri database associato all'istanza database, tali modifiche vengono applicate immediatamente senza eseguire il riavvio.

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel menu di navigazione scegli Parameter groups (Gruppi di parametri).
3. Nell'elenco, scegli il gruppo di parametri del cluster database `default.aurora-postgresql14`.
4. Inserisci il nome di un parametro nel campo di ricerca. Ad esempio, inserisci `babelfishpg_tsql.default_locale` nel campo di ricerca per visualizzare questo parametro e il suo valore predefinito e le impostazioni consentite.

> **Note**
>
> I database globali PostgreSQL di Babelfish for Aurora funzionano nelle aree secondarie solo se i seguenti parametri sono attivati in tali aree.

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tds.tds_default_numeric_scale	Imposta la scala predefinita di tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno. (Predefinito: 8) (Consentito: 0–38)	dynamic	true
babelfishpg_tds.tds_default_numeric_precision	Un valore intero che imposta la precision e predefinita del tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno. (Predefinito: 38) (Consentito: 1–38)	dynamic	true

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tds.tds_default_packet_size	Un valore intero che imposta la dimensione predefinita del pacchetto per la connessione di client SQL Server. (Predefinito: 4096) (Consentito: 512–32767)	dynamic	true
babelfishpg_tds.tds_default_protocol_version	Un valore intero che imposta una versione del protocollo TDS predefinita per la connessione dei client. (Predefinito: DEFAULT) (Consentito: TDSv7.0, TDSv7.1, TDSv7.1.1, TDSv7.2, TDSv7.3A, TDSv7.3B, TDSv7.4, DEFAULT)	dynamic	true
babelfishpg_tds.default_server_name	Una stringa che identifica il nome predefinito del server Babelfish. (Predefinito: Microsoft SQL Server) (Consentito: null)	dynamic	true

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tds.tds_debug_log_level	Un valore intero che imposta il livello di registrazione in TDS; 0 disattiva la registrazione. (Predefinito: 1) (Consentito: 0, 1, 2, 3)	dynamic	true
babelfishpg_tds.listen_address	Una stringa che imposta il nome host o l'indirizzo IP o gli indirizzi su cui ascoltare TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: *) (Consentito: null)	–	false
babelfishpg_tds.port	Un valore intero che imposta la porta TCP utilizzata per le richieste nella sintassi di SQL Server. (Predefinito: 1433) (Consentito: 1–65535)	static	true

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tds.tds_ssl_encrypt	Un valore booleano che attiva o disattiva la crittografia (0) (1) per i dati che attraversano la porta del listener TDS. Per informazioni dettagliate sull'utilizzo di SSL per le connessioni client, consulta Impostazioni SSL Babelfish e connessioni client. (Predefinito: 0) (Consentito: 0, 1)	dynamic	true
babelfishpg_tds.tds_ssl_max_protocol_version	Una stringa che specifica la versione del protocollo SSL/TLS più alta da utilizzare per la sessione TDS. (Predefinito: 'TLSv1.2') (Consentito: 'TLSv1', 'TLSv1.1', 'TLSv1.2')	dynamic	true

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tds.tds_ssl_min_protocol_version	Una stringa che specifica la versione del protocollo SSL/TLS minima da utilizzare per la sessione TDS. (Impostazione predefinita: 'TLSv1.2' da Aurora PostgreSQL versione 16, 'TLSv1' per versioni precedenti a Aurora PostgreSQL versione 16) (consentito: 'TLSv1', 'TLSv1.1', 'TLSv1.2')	dynamic	true
babelfishpg_tds.unix_socket_directories	Una stringa che identifica la directory socket Unix server TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: /tmp) (Consentito: null)	–	false

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tds.unix_socket_group	Una stringa che identifica il gruppo socket Unix server TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: rdsdb) (Consentito: null)	–	false

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tsql.default_locale	Una stringa che specifica le impostazioni locali predefinite utilizzate per le regole di confronto Babelfish. Le impostazioni locali predefinite sono le uniche e non includono eventuali qualificatori. Impostare questo parametro quando si esegue il provisioning di un cluster Babelfish DB. Dopo aver eseguito il provisioning del cluster database, le modifiche apportate a questo parametro vengono ignorate. (Predefinito: en_US) (Consentito: consulta tabelle)	static	true

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tsql.migration_mode	Un elenco non modificabile che specifica il supporto per database a un solo utente o più utenti. Impostare questo parametro quando si esegue il provisioning di un cluster Babelfish DB. Dopo aver eseguito il provisioning del cluster database, non è possibile modificare il valore di questo parametro. (Impostazione predefinita: multi-db da Aurora PostgreSQL versione 16, single-db per versioni precedenti a Aurora PostgreSQL versione 16) (consentito: single-db, multi-db, null)	static	true

Parametro	Descrizione	Applica tipo	È modificabile
babelfishpg_tsql.server_collation_name	Una stringa che specifica il nome della regola di confronto utilizzata per le azioni a livello di server. Impostare questo parametro quando si esegue il provisioning di un cluster Babelfish DB. Dopo aver eseguito il provisioning del cluster di database, non modificare il valore di questo parametro. (Predefinito: bbf_unicode_general_ci_as) (Consentito: consulta tabelle)	static	true
babelfishpg_tsql.versione	Una stringa che imposta l'output della variabile @@VERSION. Non modificare questo valore per i cluster Aurora PostgreSQL DB. (Predefinito: null) (Consentito: default)	dynamic	true

Parametro	Descrizione	Applica tipo	È modificabile
rds.babelfish_status	Una stringa che imposta lo stato della funzionalità Babelfish. Quando questo parametro è impostato su `datatypesonly` , Babelfish è disattivato ma i tipi di dati di SQL Server sono ancora disponibili. (Predefinito: off) (Consentito: on, off, datatypesonly)	static	true
unix_socket_permissions	Un valore intero che imposta le autorizzazioni socket Unix server TDS. Questo parametro non può essere modificato dopo la creazione del cluster database Babelfish. (Predefinito: 0700) (Consentito: 0–511)	–	false

Impostazioni SSL Babelfish e connessioni client

Quando un client si connette alla porta TDS (default `1433`), Babelfish confronta l'impostazione Secure Sockets Layer (SSL) inviata durante l'handshake del client con l'impostazione del parametro SSL Babelfish (`tds_ssl_encrypt`). Babelfish determina quindi se è consentita una connessione. Se è consentita una connessione, il comportamento di crittografia viene applicato o meno, a seconda delle impostazioni dei parametri e del supporto per la crittografia offerto dal client.

La tabella seguente mostra come si comporta Babelfish per ogni combinazione.

Impostazione del client dell'app	Impostazione SSL Babelfish	Connessione consentita?	Valore restituito al client
ENCRYPT_OFF	tds_ssl_encrypt=0	Consentito, il pacchetto di accesso è crittografato	ENCRYPT_OFF
ENCRYPT_OFF	tds_ssl_encrypt=1	Consentito, l'intera connessione è crittografata	ENCRYPT_REQ
ENCRYPT_ON	tds_ssl_encrypt=0	Consentito, l'intera connessione è crittografata	ENCRYPT_ON
ENCRYPT_ON	tds_ssl_encrypt=1	Consentito, l'intera connessione è	ENCRYPT_ON

Impostazione del client dell'app	Impostazione SSL Babelfish	Connessione consentita?	Valore restituito al client
		crittografata	
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	Sì	ENCRYPT_NOT_SUP
ENCRYPT_NOT_SUP	tds_ssl_encrypt=1	No, connessione chiusa	ENCRYPT_REQ
ENCRYPT_REQ	tds_ssl_encrypt=0	Consentito, l'intera connessione è crittografata	ENCRYPT_ON
ENCRYPT_REQ	tds_ssl_encrypt=1	Consentito, l'intera connessione è crittografata	ENCRYPT_ON
ENCRYPT_CLIENT_CERT	tds_ssl_encrypt=0	No, connessione chiusa	Non supportato.
ENCRYPT_CLIENT_CERT	tds_ssl_encrypt=1	No, connessione chiusa	Non supportato.

Comprendere le collazioni in Babelfish per Aurora Postgre SQL

Quando crei un cluster Aurora Postgre SQL DB con Babelfish, scegli una collazione per i tuoi dati. Una regola di confronto specifica i modelli di ordinamento e bit che producono il testo o i caratteri in una determinata lingua scritta. Una regola di confronto include regole che confrontano i dati per un determinato set di modelli di bit. La regola di confronto è correlata alla localizzazione. Le diverse impostazioni locali influenzano la mappatura dei caratteri, l'ordinamento e simili. Gli attributi della regola di confronto si riflettono nei nomi di varie regole di confronto. Per ulteriori informazioni sugli attributi, consulta Babelfish collation attributes table.

Babelfish associa le regole di confronto del SQL Server a regole di confronto comparabili fornite da Babelfish. Babelfish predefinisce le raccolte Unicode con raccolte di stringhe culturalmente sensibili e ordini di ordinamento. Babelfish fornisce anche un modo per tradurre le regole di confronto presenti nel tuo Server DB nelle regole di confronto Babelfish più simili. SQL Vengono fornite raccolte specifice per le diverse lingue e regioni.

Alcune raccolte specificano una pagina codice corrispondente a una codifica lato client. Babelfish traduce automaticamente dalla codifica del server alla codifica client a seconda della raccolta di ciascuna colonna di output.

Babelfish supporta le regole di confronto elencate nella Babelfish supported collations table. Babelfish associa le regole di confronto del Server a regole di confronto comparabili fornite da Babelfish. SQL

Babelfish utilizza la versione 153.80 della libreria di confronto International Components for Unicode (). ICU Per ulteriori informazioni sulle regole di confronto, vedete ICU Collation nella documentazione. ICU Per saperne di più su SQL Postgre e le regole di confronto, consulta Collation Support nella documentazione di Postgre. SQL

Argomenti

- Parametri del cluster database che controllano regole di confronto e impostazioni locali
- Collazioni deterministiche e non deterministiche in Babelfish
- Le regole di confronto sono supportate a livello di database in Babelfish
- Collazioni di server e oggetti in Babelfish
- Comportamento di confronto predefinito in Babelfish
- Gestione delle raccolte
- Limitazioni di regole di confronto e differenze di comportamento

Parametri del cluster database che controllano regole di confronto e impostazioni locali

I seguenti parametri influiscono sul comportamento della regola di confronto.

babelfishpg_tsql.default_locale

Questo parametro specifica le impostazioni locali predefinite utilizzate dalla regola di confronto. Questo parametro viene utilizzato in combinazione con gli attributi elencati nella Babelfish collation attributes table per personalizzare le regole di confronto per una lingua e una regione specifici. Il valore predefinito per questo parametro è `en-US`.

Le impostazioni locali predefinite si applicano a tutti i nomi di confronto di Babelfish che iniziano con "BBF" e a tutte SQL le regole di confronto del Server mappate alle regole di confronto Babelfish. La modifica dell'impostazione di questo parametro su un cluster database Babelfish esistente non influenza le impostazioni locali delle regole di confronto esistenti. Per l'elenco delle regole di confronto, consulta Babelfish supported collations table.

babelfishpg_tsql.server_collation_name

Questo parametro specifica le regole di confronto predefinite per il server (istanza del cluster SQL DB Aurora Postgre) e il database. Il valore predefinito è `sql_latin1_general_cp1_ci_as`. Deve `server_collation_name` essere una `CI_AS` collazione perché in T-SQL, la collazione del server determina il modo in cui gli identificatori vengono confrontati.

Quando crei il tuo cluster database Babelfish, scegli il Collation name (Nome regola di confronto) dall'elenco selezionabile. Questi includono le regole di confronto elencate in Babelfish supported collations table. Non modificare il `server_collation_name` dopo la creazione del database Babelfish.

Le impostazioni che scegli quando crei il tuo cluster Babelfish for Aurora Postgre SQL DB vengono archiviate nel gruppo di parametri del cluster DB associato al cluster per questi parametri e ne impostano il comportamento di confronto.

Collazioni deterministiche e non deterministiche in Babelfish

Babelfish supporta le raccolte deterministiche e non deterministiche:

- Una regola di confronto deterministica valuta come uguali i caratteri con sequenze di byte identiche. Ciò significa che `x` e `X` non sono uguali in una regola di confronto deterministica. Le regole di confronto deterministiche possono essere con distinzione maiuscole/minuscole (CS) e con distinzione caratteri accentati/non accentati (AS).

- Una regola di confronto non deterministica non richiede una corrispondenza identica. Una raccolta non deterministica valuta x e X come uguale. Le collazioni non deterministiche sono insensibili alle maiuscole e minuscole (CI) o all'accento (AI), o entrambe.

Nella tabella seguente, puoi trovare alcune differenze di comportamento tra Babelfish e Postgre quando si utilizzano regole di confronto non deterministiche. SQL

Babelfish	Postgre SQL
Supporta la LIKE clausola per le regole di confronto CI_AS.	Non supporta la LIKE clausola sulle regole di confronto non deterministiche.
Supporta la LIKE clausola solo nelle seguenti regole di confronto AI della versione 4.2.0 di Babelfish: • bbf_unicode_cp1250_ci_ai • bbf_unicode_cp1250_cs_ai • bbf_unicode_cp1257_ci_ai • bbf_unicode_cp1257_cs_ai • bbf_unicode_cp1_ci_ai • bbf_unicode_cp1_cs_ai • estone_ci_ai • finlandese_svedese_ci_ai • frances_ci_ai • latin1_general_ci_ai • latin1_general_cs_ai • spagnolo_moderno_ci_ai • polacco_ci_ai • sql_latin1_general_cp1_ci_ai • sql_latin1_general_cp1_cs_ai • spagnolo_tradizionale_ci_ai	Non supporta la clausola sulle regole di confronto non deterministiche. LIKE

Per un elenco di altre limitazioni e differenze di comportamento per Babelfish rispetto a Server e Postgre, vedi. SQL SQL Limitazioni di regole di confronto e differenze di comportamento

Babelfish e SQL Server seguono una convenzione di denominazione per le regole di confronto che descrivono gli attributi di confronto, come mostrato nella tabella seguente.

Attributo	Descrizione
Intelligenza artificiale	Insensibile all'accento.
AS	sensibile agli accenti.
BIN2	BIN2richiede che i dati vengano ordinati in ordine di punti di codice. L'ordine dei punti di codice Unicode è lo stesso ordine dei caratteri per le UTF codifiche -8, UTF -16 e UCS -2. L'ordine dei punti di codice è una raccolta deterministica rapida.
CI	Senza distinzione tra maiuscole e minuscole
CS	Distinzione tra lettere maiuscole e minuscole
PREF	Per ordinare le lettere maiuscole prima delle lettere minuscole, utilizzate un confronto. PREF Se la raccolta è priva di distinzione tra maiuscole e minuscole, la versione maiuscola di una lettera viene ordinata prima della versione minuscola, se non esiste altra distinzione. La ICU libreria supporta la preferenza delle lettere maiuscole con`colCaseFirst=upper` , ma non per le regole di confronto CI_AS. PREFpuò essere applicato solo alle regole di confronto deterministiche. `CS_AS`

Le regole di confronto sono supportate a livello di database in Babelfish

Le seguenti regole di confronto sono supportate a livello di database in Babelfish:

- bbf_unicode_bin2
- bbf_unicode_cp1_ci_ai
- bbf_unicode_cp1_ci_as

- bbf_unicode_cp1250_ci_ai
- bbf_unicode_cp1250_ci_it
- bbf_unicode_cp1257_ci_ai
- bbf_unicode_cp1257_ci_as
- estone_ci_ai
- estone_ci_as
- finlandese_svedese_ci_ai
- finnish_swedish_ci_as
- frances_ci_ai
- french_ci_as
- latin1_general_bin2
- latin1_general_ci_ai
- latin1_general_ci_as
- latin1_general_90_bin2
- latin1_general_100_bin2
- latin1_general_140_bin2
- spagnolo_moderno_ci_ai
- modern_spanish_ci_as
- polacco_ci_ai
- polish_ci_as
- sql_latin1_general_cp1_ci_ai
- SQL_Latin1_General_CP1_CI_AS (default)
- sql_latin1_general_cp1250_ci_as
- sql_latin1_general_cp1251_ci_as
- sql_latin1_general_cp1257_ci_as
- spagnolo_tradizionale_ci_ai
- traditional_spanish_ci_as

> **Note**
>
> Per utilizzare un confronto diverso a livello di database, assicurati che corrisponda a quello a livello di server. Per ulteriori informazioni, consulta Collazioni di server e oggetti in Babelfish

Collazioni di server e oggetti in Babelfish

Utilizzare le raccolte seguenti come raccolta del server o come raccolta di oggetti.

ID raccolta	Note
bbf_unicode_general_ci_as	Supporta il confronto senza distinzione tra maiuscole e minuscole e l'operatore. LIKE
bbf_unicode_cp1_ci_as	Collazione non deterministica nota anche come. CP1252
CP125bbf_unicode_ 0_ci_as	Raccolta non deterministica utilizzata per rappresentare testi nelle lingue dell'Europa centrale e dell'Europa orientale che utilizzano lo script latino.
bbf_unicode_ CP1251 _ci_it	Raccolta non deterministica per le lingue che utilizzano lo script cirillico.
bbf_unicode_cp1253_ci_as	Raccolta non deterministica usata per rappresentare il greco moderno.
bbf_unicode_cp1254_ci_as	Raccolta non deterministica che supporta il turco.
bbf_unicode_cp1255_ci_as	Raccolta non deterministica che supporta l'ebraico.
bbf_unicode_cp1256_ci_as	Raccolta non deterministica utilizzata per scrivere lingue che utilizzano lo script arabo.

ID raccolta	Note
bbf_unicode_cp1257_ci_as	Raccolta non deterministica utilizzata per supportare le lingue estone, lettone e lituano.
bbf_unicode_cp1258_ci_as	Raccolta non deterministica utilizzata per scrivere caratteri vietnamiti.
bbf_unicode_cp874_ci_as	Raccolta non deterministica utilizzata per scrivere caratteri thailandesi.
sql_latin1_general_cp1250_ci_as	Codifica di caratteri a byte singolo non deterministico usata per rappresentare caratteri latini.
sql_latin1_general_cp1251_ci_as	Raccolta non deterministica che supporta i caratteri latini.
SQL_Latin1_General_CP1_CI_AS (default)	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1253_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1254_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1255_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1256_ci_as	Raccolta non deterministica che supporta i caratteri latini.

ID raccolta	Note
sql_latin1_general_cp1257_ci_as	Raccolta non deterministica che supporta i caratteri latini.
sql_latin1_general_cp1258_ci_as	Raccolta non deterministica che supporta i caratteri latini.
chinese_PRC_CI_AS	Collazione non deterministica che supporta il cinese (). PRC
cyrillic_general_ci_as	Raccolta non deterministica che supporta il cirillico.
finnish_swedish_ci_as	Raccolta non deterministica che supporta che supporta il finlandese.
french_ci_as	Raccolta non deterministica che supporta il francese.
japanese_ci_as	Regola di confronto non deterministica che supporta il giapponese. Supportata in Babelfish 2.1.0 e versioni successive.
korean_wansung_ci_as	Raccolta non deterministica che supporta il coreano (con ordinamento del dizionario).
latin1_general_ci_as	Raccolta non deterministica che supporta i caratteri latini.
modern_spanish_ci_as	Raccolta non deterministica che supporta lo spagnolo moderno.
polish_ci_as	Raccolta non deterministica che supporta il polacco.
thai_ci_as	Raccolta non deterministica che supporta il tailandese.

ID raccolta	Note
traditional_spanish_ci_as	Raccolta non deterministica che supporta lo spagnolo (classificazione tradizionale).
turkish_ci_as	Raccolta non deterministica che supporta il turco.
ukrainian_ci_as	Raccolta non deterministica che supporta l'ucraino.
vietnamese_ci_as	Raccolta non deterministica che supporta il vietnamita.

È possibile utilizzare le seguenti raccolte come raccolte tra oggetti.

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Arabo	Arabic_cs_as	Arabic_CI_AS Arabico_CI_AI
Alfabeto arabo	BBF_Unicode_ _CS_AS CP1256 BBF_Unicode_Pref_ CP1256 _CS_AS	BBF_Unicode_ CP1256 _CI_AI BBF_Unicode_ CP1256 _CS_AI
Binario	latino1_general_bin2 BBF_Unicode_ BIN2	–
Lingue dell'Europa centrale e dell'Europa orientale	BBF_Unicode_ 0_CS_AS CP125 BBF_Unicode_Pref_ CP125 0_CS_AS	BBF_Unicode_ CP125 0_CI_AI BBF_Unicode_ CP125 0_CS_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
che utilizzano caratteri latini		
Cinese	Cinese_ PRC _CS_AS	Cinese_ PRC _CI_AS Cinese_ PRC _CI_AI
Cyrillic_General	Cyrillic_general_cs_as	Cirillico generale_CI_AS cirillico_general_CI_AI
Scrittura cirillica	BBF_Unicode_ CP1251 _CS_AS BBF_Unicode_Pref_ CP1251 _CS_AS	BBF_Unicode_ CP1251 _CI_AI BBF_Unicode_ CP1251 _CS_AI
Estone	Estonian_cs_as	estone_CI_AS Estone_CI_AI
estone, lettone e lituano	BBFCP1257_Unicode_ _CS_AS BBF_Unicode_Pref_ CP1257 _CS_AS	BBF_Unicode_ CP1257 _CI_AI BBF_Unicode_ CP1257 _CS_AI
Finnish_Swedish	Finnish_Swedish_CS_AS	Finnish_Swedish_CI_AS Finlandese_svedese_CI_AI
Francese	French_CS_AS	French_CI_AS Francese_CI_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Greco	Greek_CS_AS	Greco_CI_AS Greco_CI_AI
Ebraico	BBF_Unicode_ _CS_AS CP1255 BBF_Unicode_Pref_ CP1255 _CS_AS Hebrew_CS_AS	BBF_Unicode_ CP1255 _CI_AI BBF_Unicode_ CP1255 _CS_AI Hebrew_CI_AS Ebraico_CI_AI
Giapponese (Babelfish 2.1.0 e versioni successive)	Japanese_CS_AS	Giapponese_CI_AI Japanese_CI_AS
Korean_Wamsung	Korean_Wamsung_CS_AS	coreano_Wamsung_CI_AS coreano_Wamsung_CI_AI
Caratteri latini per la tabella codici CP1252	latin1_general_cs_as BBF_Unicode_general_cs_as BBF_Unicode_general_pref_cs_AS BBFCP1_Unicode_Pref_ _CS_AS BBF_Unicode_ _CS_AS CP1	latin1_general_ci_as latin1_general_ci_ai latin1_general_cs_ai BBF_Unicode_generale_CI_AI BBF_Unicode_General_CS_AI BBFCP1_Unicode_ _CI_AI BBF_Unicode_ CP1 _CS_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Greco moderno	BBF_Unicode_ _CS_AS CP1253 BBF_Unicode_Pref_ CP1253 _CS_AS	BBF_Unicode_ CP1253 _CI_AI BBF_Unicode_ CP1253 _CS_AI
Modern_Spanish	Modern_Spanish_CS_AS	Modern_Spanish_CI_AS CI_AI in spagnolo moderno
Mongolo	Mongolian_CS_AS	CI_AS mongolo mongolo_CI_AI
Polacco	Polish_CS_AS	Polish_CI_AS Polacco_CI_AI
Thai	BBF_Unicode_ _CS_AS CP874 BBF_Unicode_Pref_ CP874 _CS_AS Thai_CS_AS	BBF_Unicode_ CP874 _CI_AI BBF_Unicode_ CP874 _CS_AI Thai_CI_AS, Thai_CI_AI
Tradition al_Spanish	Traditional_Spanish_CS_AS	CI_AS in spagnolo tradizionale Spagnolo_tradizionale_CI_AI
Turco	BBF_Unicode_ _CS_AS CP1254 BBF_Unicode_Pref_ CP1254 _CS_AS Turkish_CS_AS	BBF_Unicode_ CP1254 _CI_AI BBF_Unicode_ CP1254 _CS_AI Turkish_CI_AS, Turkish_CI_AI

Dialetto	Opzioni deterministiche	Opzioni non deterministiche
Ucraino	Ukranian_CS_AS	UCRAINO_CI_AS Ucraino_CI_AI
Vietnamita	BBF_Unicode_ _CS_AS CP1258 BBF_Unicode_Pref_ CP1258 _CS_AS Vietnamese_CS_AS	BBF_Unicode_ CP1258 _CI_AI BBF_Unicode_ CP1258 _CS_AI CI_AS vietnamita Vietnamita CI_AI

Comportamento di confronto predefinito in Babelfish

In precedenza, la regola di confronto predefinita dei tipi di dati di regola di confronto era `pg_catalog.default`. I tipi di dati e gli oggetti che dipendono da questi tipi di dati seguono la regola di confronto con distinzione tra maiuscole e minuscole. Questa condizione ha un potenziale impatto sugli SQL oggetti T del set di dati con regole di confronto senza distinzione tra maiuscole e minuscole. A partire da Babelfish 2.3.0, le regole di confronto predefinite per i tipi di dati collezionabili (eccetto TEXT eNTEXT) sono le stesse delle regole di confronto nel parametro. `babelfishpg_tsql.server_collation_name` Quando esegui l'aggiornamento a Babelfish 2.3.0, la regola di confronto predefinita viene selezionata automaticamente al momento della creazione del cluster database, senza alcun impatto visibile.

Gestione delle raccolte

La libreria ICU fornisce il tracciamento delle raccolte per garantire che gli indici che dipendono dalle raccolte possano essere reindicizzati quando diventa disponibile una nuova versione di ICU. Per verificare se il database corrente contiene regole di confronto che richiedono l'aggiornamento, puoi utilizzare la seguente query dopo aver effettuato la connessione utilizzando `psql` o `pgAdmin`:

```
SELECT pg_describe_object(refclassid, refobjid,
    refobjsubid) AS "Collation",
    pg_describe_object(classid, objid, objsubid) AS "Object"
    FROM pg_depend d JOIN pg_collation c ON refclassid = 'pg_collation'::regclass
    AND refobjid = c.oid WHERE c.collversion <> pg_collation_actual_version(c.oid)
```

```
    ORDER BY 1, 2;
```

Questa query restituisce un output del tipo seguente:

```
 Collation | Object
-----------+--------
(0 rows)
```

In questo esempio, non è necessario aggiornare regole di confronto.

Per ottenere un elenco delle regole di confronto predefinite nel database Babelfish, puoi utilizzare `psql` o `pgAdmin` con la seguente query:

```
SELECT * FROM pg_collation;
```

Le raccolte predefinite sono memorizzate nella tabella `sys.fn_helpcollations`. È possibile utilizzare il seguente comando per visualizzare informazioni su una raccolta (ad esempio i flag lcid, style e collate). Per ottenere un elenco di tutte le regole di confronto utilizzando `sqlcmd`, esegui la connessione alla porta T-SQL (1433, per impostazione predefinita) ed esegui la seguente query:

```
1> :setvar SQLCMDMAXVARTYPEWIDTH 40
2> :setvar SQLCMDMAXFIXEDTYPEWIDTH 40
3> SELECT * FROM fn_helpcollations()
4> GO
name                                    description
--------------------------------------- ----------------------------------------
arabic_cs_as                            Arabic, case-sensitive, accent-sensitive
arabic_ci_ai                            Arabic, case-insensitive, accent-insensi
arabic_ci_as                            Arabic, case-insensitive, accent-sensiti
bbf_unicode_bin2                        Unicode-General, case-sensitive, accent-
bbf_unicode_cp1250_ci_ai                Default locale, code page 1250, case-ins
bbf_unicode_cp1250_ci_as                Default locale, code page 1250, case-ins
bbf_unicode_cp1250_cs_ai                Default locale, code page 1250, case-sen
bbf_unicode_cp1250_cs_as                Default locale, code page 1250, case-sen
bbf_unicode_pref_cp1250_cs_as           Default locale, code page 1250, case-sen
bbf_unicode_cp1251_ci_ai                Default locale, code page 1251, case-ins
bbf_unicode_cp1251_ci_as                Default locale, code page 1251, case-ins
bbf_unicode_cp1254_ci_ai                Default locale, code page 1254, case-ins
...
(124 rows affected)
```

Le righe 1 e 2 mostrate nell'esempio restringono l'output solo ai fini della leggibilità della documentazione.

```
1> SELECT SERVERPROPERTY('COLLATION')
2> GO
serverproperty
--------------------------------------------------------------------------
sql_latin1_general_cp1_ci_as

(1 rows affected)
1>
```

Limitazioni di regole di confronto e differenze di comportamento

Babelfish utilizza la libreria ICU per il supporto delle raccolte. PostgreSQL è costruito con una versione specifica di ICU e può corrispondere al massimo a una versione di un confronto. Le variazioni tra le versioni sono inevitabili, così come le piccole variazioni nel tempo man mano che le lingue evolvono. Nella sezione seguente sono disponibili le limitazioni note e le variazioni di comportamento delle regole di confronto Babelfish:

- Indici e dipendenza tipo regola di confronto: un indice su un tipo definito dall'utente che dipende dalla libreria di regole di confronto ICU (International Components for Unicode) (la libreria utilizzata da Babelfish) non viene invalidato quando la versione della libreria viene modificata.
- Funzione COLLATIONPROPERTY: le proprietà delle regole di confronto sono implementate solo per le regole di confronto Babelfish BBF supportate. Per ulteriori informazioni, consulta la Babelfish supported collations table.
- Differenze regole di ordinamento Unicode: le regole di confronto SQL per SQL Server ordinano i dati con codifica Unicode (`nchar` e `nvarchar`) in modo diverso rispetto ai dati senza codifica Unicode (`char` e `varchar`). I database Babelfish sono sempre codificati UTF-8 e applicano sempre regole di ordinamento Unicode in modo coerente, a prescindere dal tipo di dati, pertanto l'ordinamento per `char` o `varchar` è uguale a quello per `nchar` o `nvarchar`.
- Regole di confronto secondarie uguali e comportamento di ordinamento: la regola di confronto predefinita Unicode secondaria uguale (`CI_AS`) ordina i segni di punteggiatura e altri caratteri non alfanumerici prima dei caratteri numerici e i caratteri numerici prima dei caratteri alfabetici. Tuttavia, l'ordine di punteggiatura e altri caratteri speciali sono diversi.
- Regole di confronto terziarie, soluzione alternativa per ORDER BY: le regole di confronto SQL, ad esempio `SQL_Latin1_General_Pref_CP1_CI_AS`, supportano la funzione `TERTIARY_WEIGHTS` e la capacità di ordinare stringhe che si confrontano allo stesso modo in una

regola di confronto `CI_AS` da ordinare prima in maiuscolo: `ABC`, `ABc`, `AbC`, `Abc`, `aBC`, `aBc`, `abC` e infine `abc`. Pertanto la funzione analitica `DENSE_RANK OVER (ORDER BY column)` valuta queste stringhe come aventi lo stesso rango, ma le ordina prima in maiuscolo all'interno di una partizione.

Puoi ottenere un risultato simile con Babelfish aggiungendo una clausola `COLLATE`alla clausola `ORDER BY` che specifica una raccolta terziaria `CS_AS` che specifica `@colCaseFirst=upper`. Tuttavia, il modificatore `colCaseFirst` si applica esclusivamente alle stringhe terziarie uguali (piuttosto che secondarie-uguali come una regola di confronto `CI_AS`). Pertanto, non è possibile emulare raccolte SQL terziarie utilizzando una singola relazione ICU.

Come soluzione alternativa, consigliamo di modificare le applicazioni che utilizzano la raccolta `SQL_Latin1_General_Pref_CP1_CI_AS` per utilizzare la raccolta `BBF_SQL_Latin1_General_CP1_CI_AS` prima di iniziare la raccolta. Quindi aggiungere `COLLATE BBF_SQL_Latin1_General_Pref_CP1_CS_AS`a qualsiasi clausola `ORDER BY` per questa colonna.

- Espansione di caratteri: un'espansione di caratteri tratta un singolo carattere come uguale a una sequenza di caratteri a livello primario. La regola di confronto `CI_AS` predefinita di SQL Server supporta l'espansione di caratteri. Le regole di confronto ICU supportano l'espansione di caratteri solo per regole di confronto senza distinzione caratteri accentati/non accentati.

  Quando è richiesta l'espansione dei caratteri, utilizzare una raccolta `AI` per confronti. Tuttavia, tali raccolte non sono attualmente supportate dall'operatore LIKE.

- Codifica char e varchar: quando le regole di confronto SQL vengono utilizzate per i tipi di dati `char` o `varchar`, l'ordinamento per i caratteri precedenti a ASCII 127 è determinato dalla pagina codice specifica per tale regola di confronto SQL. Per le regole di confronto SQL, le stringhe dichiarate come `char` o `varchar` potrebbero essere ordinate in modo diverso rispetto alle stringhe dichiarate come `nchar` o `nvarchar`.

  PostgreSQL codifica tutte le stringhe con la codifica del database, pertanto tutti i caratteri vengono convertiti in UTF-8 e ordinati utilizzando regole Unicode.

  Poiché le regole di confronto SQL ordinano i tipi di dati nchar e nvarchar utilizzando regole Unicode, Babelfish codifica tutte le stringhe sul server utilizzando UTF-8. Babelfish ordina le stringhe nchar e nvarchar nello stesso modo in cui ordina le stringhe char e varchar, usando regole Unicode.

- Caratteri supplementari: le funzioni SQL Server `NCHAR`, `UNICODE`, e `LEN` supportano i caratteri per i punti di codice al di fuori di Unicode Basic Multilingual Plane (BMP). Al contrario, le raccolte non SC utilizzano caratteri di coppia surrogata per gestire caratteri supplementari. Per i tipi di dati Unicode, SQL Server può rappresentare fino a 65.535 caratteri utilizzando UCS-2 o l'intervallo Unicode completo (1,114,114 caratteri) se si utilizzano caratteri supplementari.
- Regole di confronto con distinzione Kana (KS): una regola di confronto con distinzione Kana (KS) gestisce i caratteri Kana giapponesi `Hiragana` e `Katakana` in modo diverso. ICU supporta lo standard di raccolta giapponese`JIS X 4061`. Il modificatore di localizzazione `colhiraganaQ [on | off]` oramai obsoleto potrebbe fornire la stessa funzionalità delle raccolte KS. Tuttavia, le raccolte KS con lo stesso nome di SQL Server non sono attualmente supportate da Babelfish.
- Regole di confronto con distinzione della larghezza (WS): quando un carattere a byte singolo (mezza larghezza) e lo stesso carattere rappresentato come carattere a doppio byte (larghezza intera) vengono trattati in modo diverso, la regola di confronto viene chiamata con distinzione della larghezza (WS). Le raccolte WS con lo stesso nome di SQL Server non sono attualmente supportate da Babelfish.
- Regole di confronto con distinzione selettore di variazione (VSS): le regole di confronto con distinzione selettore di variazione (VSS) distinguono tra selettori variazione ideografica nelle regole di confronto giapponesi `Japanese_Bushu_Kakusu_140` e `Japanese_XJIS_140`. Una sequenza di varianti è costituita da un carattere base più un selettore di varianti aggiuntivo. Se non viene selezionata l'opzione `_VSS`, il selettore delle varianti non è considerato nel confronto.

  Le relazioni VSS attualmente non sono supportate da Babelfish.
- Regole di confronto BIN e BIN2: una regola di confronto BIN2 ordina i caratteri in base all'ordine dei punti di codice. L'ordine binario byte per byte di UTF-8 conserva l'ordine dei punti di codice Unicode, quindi è probabile che questo sia anche il confronto con le migliori raccolte. Se l'ordine dei punti di codice Unicode funziona per un'applicazione, prendere in considerazione l'utilizzo di una raccolta BIN2. Tuttavia, l'utilizzo di una raccolta BIN2 può comportare la visualizzazione dei dati sul client in un ordine culturalmente inaspettato. I nuovi mapping con caratteri minuscoli vengono aggiunti a Unicode man mano che il tempo progredisce, quindi la funzione `LOWER` potrebbe funzionare in modo diverso su diverse versioni di ICU. Questo è un caso speciale del problema di controllo delle versioni di raccolta più generale piuttosto che come qualcosa di specifico per la raccolta BIN2.

  Babelfish fornisce la raccolta `BBF_Latin1_General_BIN2` con la distribuzione Babelfish da collare nell'ordine dei punti di codice Unicode. In una raccolta BIN solo il primo carattere viene ordinato come wchar. I caratteri rimanenti vengono ordinati byte per byte, in modo efficace in

ordine di codice in base alla sua codifica. Questo approccio non segue le regole di confronto Unicode e non è supportato da Babelfish.

- Regole di confronto non deterministiche e limitazione CHARINDEX: per versioni di Babelfish precedenti alla versione 2.1.0, non è possibile utilizzare CHARINDEX con regole di confronto non deterministiche. Per impostazione predefinita, Babelfish utilizza una regola di confronto senza distinzione maiuscole/minuscole (non deterministica). L'utilizzo di CHARINDEX per le versioni precedenti di Babelfish genera il seguente errore di runtime:

```
nondeterministic collations are not supported for substring searches
```

> **Note**
>
> Questa limitazione e la soluzione alternativa si applicano solo a Babelfish versione 1.x (Aurora PostgreSQL versioni 13.x). Babelfish 2.1.0 e versioni successive non presentano questo problema.

Puoi risolvere il problema in uno dei seguenti modi:

- Converti esplicitamente l'espressione in un confronto con distinzione tra maiuscole e minuscole e trasforma entrambi gli argomenti applicando LOWER o UPPER. Ad esempio, `SELECT charindex('x', a) FROM t1` diventa ciò che è indicato qui di seguito:

```
SELECT charindex(LOWER('x'), LOWER(a COLLATE sql_latin1_general_cp1_cs_as)) FROM t1
```

- Crea una funzione SQL f_charindex e sostituisci le chiamate CHARINDEX con le chiamate alla seguente funzione:

```
CREATE function f_charindex(@s1 varchar(max), @s2 varchar(max)) RETURNS int
AS
BEGIN
declare @i int = 1
WHILE len(@s2) >= len(@s1)
BEGIN
  if LOWER(@s1) = LOWER(substring(@s2,1,len(@s1))) return @i
  set @i += 1
  set @s2 = substring(@s2,2,999999999)
END
return 0
END
```

```
go
```

Gestione degli errori di Babelfish con escape hatch

Babelfish imita il SQL comportamento per il flusso di controllo e lo stato delle transazioni ogni volta che è possibile. Quando Babelfish rileva un errore, restituisce un codice di errore simile al codice di errore del server. SQL Se Babelfish non riesce a mappare l'errore a un codice SQL Server, restituisce un codice di errore fisso (`33557097`) e intraprende azioni specifiche in base al tipo di errore, come segue:

- Per errori in fase di compilazione, Babelfish esegue il rollback della transazione.
- Per errori di runtime, Babelfish termina il batch ed esegue il rollback della transazione.
- Per errore di protocollo tra client e server, non viene eseguito il rollback della transazione.

Se un codice di errore non può essere mappato a un codice equivalente e il codice per un errore simile è disponibile, il codice di errore viene mappato al codice alternativo. Ad esempio, i comportamenti che causano i codici SQL Server `8143` e su cui `8144` sono entrambi mappati. `8143`

Gli errori che non possono essere mappati non rispettano un costrutto `TRY... CATCH`.

È possibile utilizzare `@@ERROR` per restituire un codice di errore SQL del server o la `@@PGERROR` funzione per restituire un codice di errore SQL Postgre. È possibile utilizzare anche la funzione `fn_mapped_system_error_list` per restituire un elenco di codici di errore mappati. Per informazioni sui codici di SQL errore Postgre, consulta il sito Web di Postgre. SQL

Modifica delle impostazioni dell'escape hatch Babelfish

Per gestire meglio le istruzioni che potrebbero non andare a buon fine, Babelfish definisce alcune opzioni chiamate escape hatch. Una via di fuga è un'opzione che specifica il comportamento di Babelfish quando incontra una caratteristica o una sintassi non supportata.

Puoi utilizzare la procedura memorizzata `sp_babelfish_configure` per controllare le impostazioni di una via di fuga. Usa lo script per impostare la via di fuga su `ignore` o `strict`. Se è impostata su `strict`, Babelfish restituisce un errore che è necessario correggere prima di continuare.

Includere la parola chiave `server` per applicare le modifiche alla sessione corrente e a livello di cluster.

L'utilizzo è il seguente:

- Per elencare tutte le vie di fuga e il relativo stato, oltre alle informazioni sull'utilizzo, eseguire `sp_babelfish_configure`.
- Per elencare le vie di fuga denominate e i relativi valori, per la sessione corrente o a livello di cluster, esegui il comando `sp_babelfish_configure 'hatch_name'` where *hatch_name* è l'identificatore di una o più vie di fuga. *hatch_name*può usare caratteri SQL jolly, come '%'.
- Per impostare una o più vie di fuga sul valore specificato, eseguire `sp_babelfish_configure ['hatch_name' [, 'strict'|'ignore' [, 'server']]]`. Per rendere le impostazioni permanenti a livello di cluster, includi la parola chiave `server`, come nell'esempio seguente:

  ```
  EXECUTE sp_babelfish_configure 'escape_hatch_unique_constraint', 'ignore', 'server'
  ```

  Per impostarle solo per la sessione corrente, non utilizzare `server`.
- Per ripristinare tutti gli escape hatch ai valori predefiniti, esegui `sp_babelfish_configure 'default'` (Babelfish 1.2.0 e versioni successive).

La stringa che identifica il tratteggio (o i tratteggi) può includere caratteri jolly. SQL Ad esempio, quanto segue imposta tutti gli escape hatch della sintassi `ignore` per il cluster Aurora Postgre. SQL

```
EXECUTE sp_babelfish_configure '%', 'ignore', 'server'
```

Nella tabella seguente è possibile trovare descrizioni e valori predefiniti per gli escape hatch predefiniti di Babelfish.

Via di fuga	Descrizione	Default
`escape_hatch_checkpoint`	Consente l'uso di un'CHECKPOINTistruzione nel codice procedurale, ma l'istruzione non è attualmente implement ataCHECKPOINT.	ignora
`escape_hatch_constraint_name_for_default`	Controlla il comportamento di Babelfish correlato ai nomi dei vincoli predefiniti.	ignora

Via di fuga	Descrizione	Default
`escape_hatch_database_misc_options`	Controlla il comportamento di Babelfish in relazione alle seguenti opzioni su CREATEDATABASE:CONTAINMENT, DB_CHAINING, TRUSTWORTHY_ _. PERSISTENT LOG BUFFER	ignora
`escape_hatch_for_replication`	Controlla il comportamento di Babelfish relativo alla FOR REPLICATION clausola [NOT] durante la creazione o la modifica di una tabella.	strict
`escape_hatch_fulltext`	Controlla il comportamento di Babelfish in relazione alle FULLTEXT funzionalità, come DEFAULT _ FULLTEXT _ LANGUAGE inCREATE/o sp_fulltext_database. ALTER DATABASE CREATE FULLTEXT INDEX	ignora
`escape_hatch_ignore_dup_key`	Controlla il comportamento di Babelfish relativo a/e. CREATE ALTER TABLE CREATE INDEX Quando IGNORE _ DUP _ KEY =ON, genera un errore se impostato su `strict` (impostazione predefinita) o ignora l'errore quando è impostato su `ignore` (Babelfish versione 1.2.0 e successive).	strict

Via di fuga	Descrizione	Default
`escape_hatch_index_clustering`	Controlla il comportamento di Babelfish relativo alle parole chiave o per gli indici e/o i vincoliCLUSTERED. NONCLUSTERED PRIMARY KEY UNIQUE Quando CLUSTERED viene ignorato, l'indice o il vincolo viene comunque creato come se fosse stato specificato. NONCLUSTERED	ignora
`escape_hatch_index_columnstore`	Controlla il comportamento di Babelfish in relazione alla clausola. COLUMNSTORE Se si specifica `ignore`, Babelfish crea un normale indice B-tree.	strict
`escape_hatch_join_hints`	Controlla il comportamento delle parole chiave in un JOIN operatore :LOOP,,,HASH,, MERGEREMOTE,REDUCE. REDISTRIBUTE REPLICATE	ignora

Via di fuga	Descrizione	Default
`escape_hatch_language_non_english`	Controlla il comportamento di Babelfish relativo a lingue diverse dall'inglese per i messaggi sullo schermo. Babelfish attualmente supporta solo i messaggi `us_english` sullo schermo. SETLANGUAGEpotrebbe utilizzare una variabile contenente il nome della lingua, quindi la lingua effettiva impostata può essere rilevata solo in fase di esecuzione.	strict
`escape_hatch_login_hashed_password`	Se ignorato, elimina l'errore per la parola chiave `HASHED` per `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_misc_options`	Se ignorato, elimina l'errore per altre parole chiave oltre `HASHED`, `MUST_CHANGE` , `OLD_PASSWORD` , e `UNLOCK` per `CREATE LOGIN` and `ALTER LOGIN`.	strict
`escape_hatch_login_old_password`	Se ignorato, elimina l'errore per la parola chiave `OLD_PASSWORD` per `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_login_password_must_change`	Se ignorato, elimina l'errore per la parola chiave `MUST_CHANGE` per `CREATE LOGIN` e `ALTER LOGIN`.	strict

Via di fuga	Descrizione	Default
`escape_hatch_login_password_unlock`	Se ignorato, elimina l'errore per la parola chiave `UNLOCK` per `CREATE LOGIN` e `ALTER LOGIN`.	strict
`escape_hatch_nocheck_add_constraint`	Controlla il comportamento di Babelfish in relazione alla NOCHECK clausola WITH CHECK or per i vincoli.	strict
`escape_hatch_nocheck_existing_constraint`	Controlla il comportamento di Babelfish in relazione ai vincoli o ai vincoli. FOREIGN KEY CHECK	strict
`escape_hatch_query_hints`	Controlla il comportamento di Babelfish correlato ai suggerimenti sulle query. Quando questa opzione è impostata su ignore, il server ignora i suggerimenti che utilizzano la clausola OPTION (...) per specificare gli aspetti di elaborazione delle query. Gli esempi includono... SELECT FROM OPTION(MERGEJOINHASH, MAXRECURSION 10).	ignora
`escape_hatch_rowversion`	Controlla il comportamento dei tipi di TIMESTAMP dati ROWVERSION e. Per informazioni sull'utilizzo, consulta Utilizzo di funzionalità Babelfish con implementazione limitata.	strict

Via di fuga	Descrizione	Default
`escape_hatch_schemabinding_function`	Controlla il comportamento di Babelfish in relazione alla clausola. WITH SCHEMABINDING Per impostazione predefinita, la WITH SCHEMABINDING clausola viene ignorata quando viene specifica ta con il comando o. CREATE ALTER FUNCTION	ignora
`escape_hatch_schemabinding_procedure`	Controlla il comportamento di Babelfish in relazione alla clausola. WITH SCHEMABINDING Per impostazione predefinita, la WITH SCHEMABINDING clausola viene ignorata quando viene specifica ta con il comando o. CREATE ALTER PROCEDURE	ignora
`escape_hatch_rowguidcol_column`	Controlla il comportamento di Babelfish relativo alla ROWGUIDCOL clausola durante la creazione o la modifica di una tabella.	strict
`escape_hatch_schemabinding_trigger`	Controlla il comportamento di Babelfish in relazione alla clausola. WITH SCHEMABINDING Per impostazione predefinita, la WITH SCHEMABINDING clausola viene ignorata quando viene specifica ta con il comando o. CREATE ALTER TRIGGER	ignora

Via di fuga	Descrizione	Default
`escape_hatch_schemabinding_view`	Controlla il comportamento di Babelfish in relazione alla clausola. WITH SCHEMABINDING Per impostazione predefinita, la WITH SCHEMABINDING clausola viene ignorata quando viene specifica ta con il comando o. CREATE ALTER VIEW	ignora
`escape_hatch_session_settings`	Controlla il comportamento di Babelfish nei confronti delle istruzioni a livello di sessione non supportate. SET	ignora
`escape_hatch_showplan_all`	Controlla il comportamento di Babelfish relativo a _ e. SET SHOWPLAN ALL SET STATISTIC S PROFILE Quando sono impostati su ignore, si comportan o come SET BABELFISH _ _ ALL e SHOWPLAN SET BABELFISH _ STATISTICSPROFILE; quando sono impostati su strict, vengono ignorati silenziosamente.	strict
`escape_hatch_storage_on_par tition`	Controlla il comportamento di Babelfish correlato alla clausola `ON partition_scheme column` durante la definizio ne del partizionamento. Babelfish attualmente non implementa il partizionamento.	strict

Via di fuga	Descrizione	Default
`escape_hatch_storage_options`	Escape hatch su qualsiasi opzione di archiviazione utilizzata inCREATE,,,. ALTER DATABASE TABLE INDEX Ciò include le clausole (LOG) ON, TEXTIMAGE _ON, FILESTREAM _ON che definiscono le posizioni di archiviazione (partizioni, gruppi di file) per tabelle, indici e vincoli e anche per un database. Questa impostazione di escape hatch si applica a tutte queste clausole (incluse ON [] e ON "«). PRIMARY DEFAULT L'eccezione è quando viene specificata una partizione per una tabella o un indice con ON partition _scheme (colonna).	ignora
`escape_hatch_table_hints`	Controlla il comportamento dei suggerimenti di tabella specificati utilizzando la clausola WITH (...).	ignora

Via di fuga	Descrizione	Default
`escape_hatch_unique_constraint`	Se impostato su strict, un'oscura differenza semantica tra SQL Server e Postgre SQL nella gestione dei NULL valori sulle colonne indicizzate può generare errori. La differenza semantica emerge solo in casi d'uso non realistici, quindi puoi impostare questo hatch di escape su "ignore" per evitare di vedere l'errore. Obsoleto dalle seguenti versioni: 3.6.0 e versioni successive, 4.2.0 e versioni successive	strict

Creazione di un cluster Babelfish per Aurora Postgre DB SQL

Babelfish for Aurora Postgre SQL è supportato nella versione 13.4 e successive di Aurora Postgre. SQL

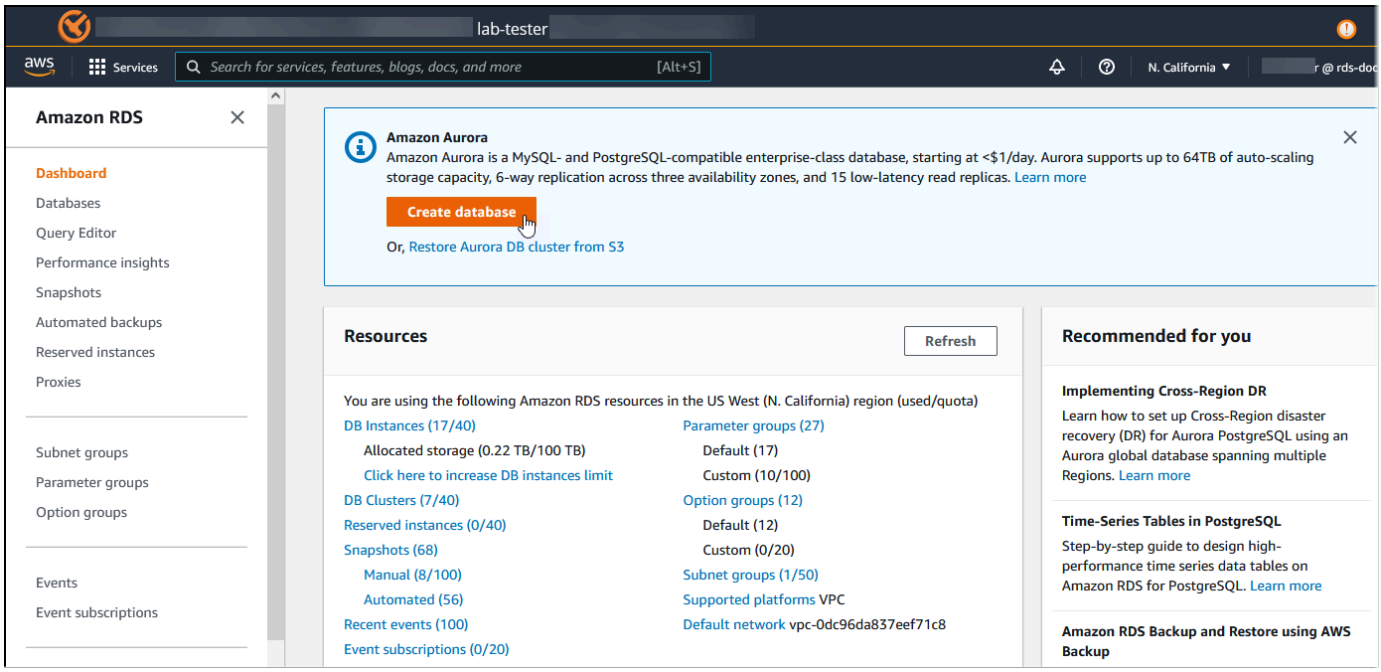
Puoi utilizzare il plugin AWS Management Console o AWS CLI per creare un cluster Aurora Postgre SQL con Babelfish.

> **Note**
>
> In un SQL cluster Aurora Postgre, il nome del `babelfish_db` database è riservato a Babelfish. La creazione del proprio database «babelfish_db» su un Babelfish per Aurora Postgre impedisce ad Aurora di effettuare correttamente il provisioning di Babelfish. SQL

Console

Per creare un cluster con Babelfish in esecuzione con AWS Management Console

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/e scegli Crea database.

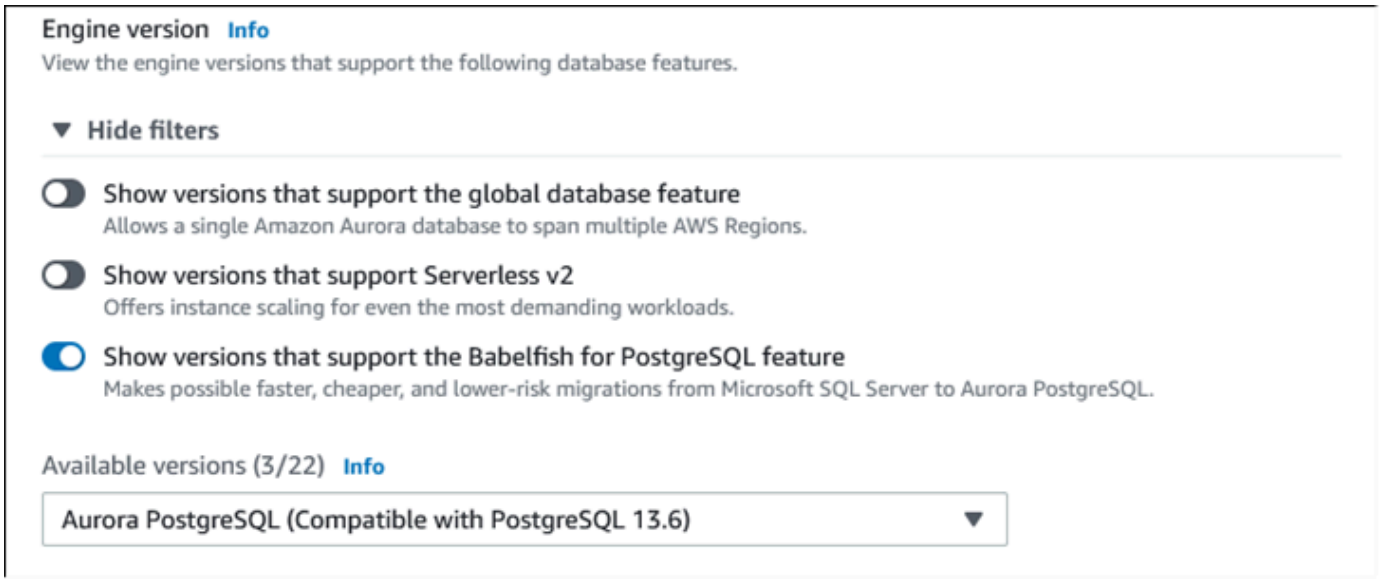


2. Per Scegli un metodo di creazione del database, procedi in uno dei seguenti modi:

   - Per specificare opzioni dettagliate del motore, scegli Standard create (Creazione standard).

- Per utilizzare opzioni preconfigurate che supportano le best practice per un cluster Aurora, scegli Creazione semplice.

3. Per Tipo di motore, scegli Aurora (compatibile con SQL Postgre).
4. Scegli Mostra filtri, quindi scegli Mostra versioni che supportano la SQL funzionalità Babelfish for Postgre per elencare i tipi di motore che supportano Babelfish. Babelfish è attualmente supportato su Aurora SQL Postgre 13.4 e versioni successive.
5. Per le versioni disponibili, scegli una versione Aurora SQL Postgre. Per ottenere le ultime funzionalità di Babelfish, scegli la versione principale di Aurora Postgre più alta. SQL

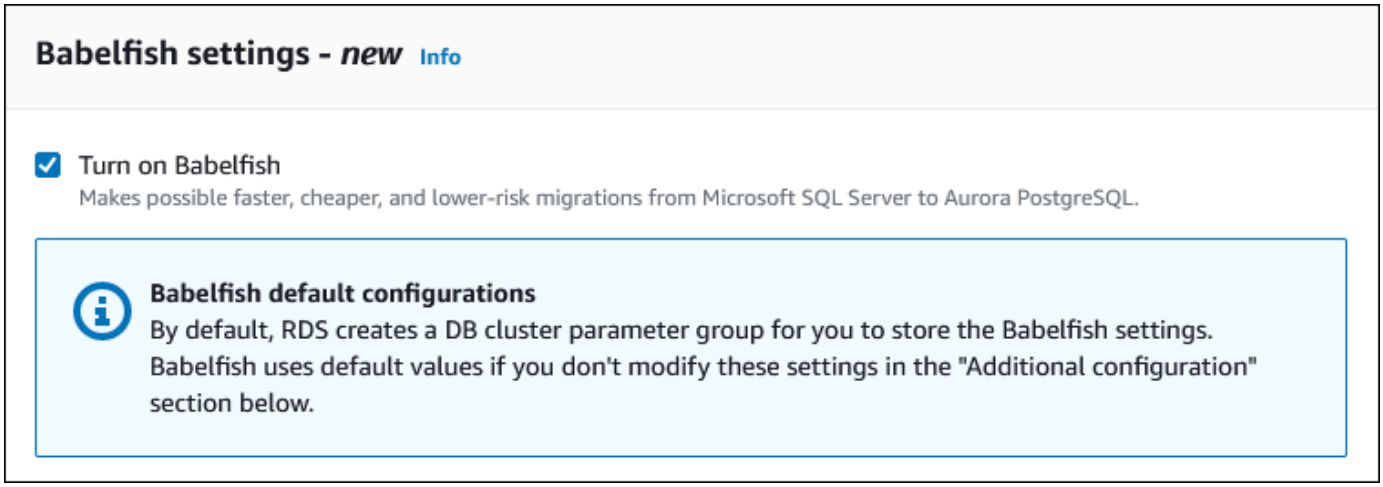


6. In Templates (Modelli), seleziona il modello che corrisponde al proprio caso d'uso.
7. Per Identificatore cluster di database, inserisci un nome che puoi facilmente trovare in seguito nell'elenco dei cluster di database.
8. Per Master username (Nome utente master), immetti un nome utente amministratore. Il valore predefinito per Aurora SQL Postgre è. `postgres` Puoi accettare il valore predefinito o scegliere un nome diverso. Ad esempio, per seguire la convenzione di denominazione utilizzata nei database SQL del Server, è possibile immettere `sa` (amministratore di sistema) come nome utente principale.

   Se non crei un utente chiamato `sa` in questo momento, puoi crearne uno in un secondo momento con la tua scelta del cliente. Dopo aver creato l'utente, utilizza il comando `ALTER SERVER ROLE` per aggiungerlo al gruppo (ruolo) `sysadmin` per il cluster.

> **Warning**
>
> Il nome utente principale deve sempre utilizzare caratteri minuscoli, altrimenti il cluster DB non può connettersi a Babelfish tramite la porta. TDS

9. Per Master password (Password master), crea una password complessa e confermala.
10. Per le opzioni che seguono, fino alla sezione Impostazioni di Babelfish,specifica le impostazioni del cluster di database. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.
11. Per rendere disponibile la funzionalità Babelfish, seleziona la casella Turn on Babelfish (Attiva Babelfish).



12. Per Gruppo di parametri del cluster database, procedere in uno dei seguenti modi:

    - Scegliere Crea nuovo per creare un nuovo gruppo di parametri con Babelfish attivato.
    - Scegliere Scegli esistente per utilizzare un gruppo di parametri esistente. Se si utilizza un gruppo esistente, assicurarsi di modificare il gruppo prima di creare il cluster e aggiungere valori per i parametri Babelfish. Per informazioni sull'estensione dei parametri consulta Impostazioni del gruppo di parametri del cluster database per Babelfish.

      Se si utilizza un gruppo esistente, specificare il nome del gruppo nella casella che segue.

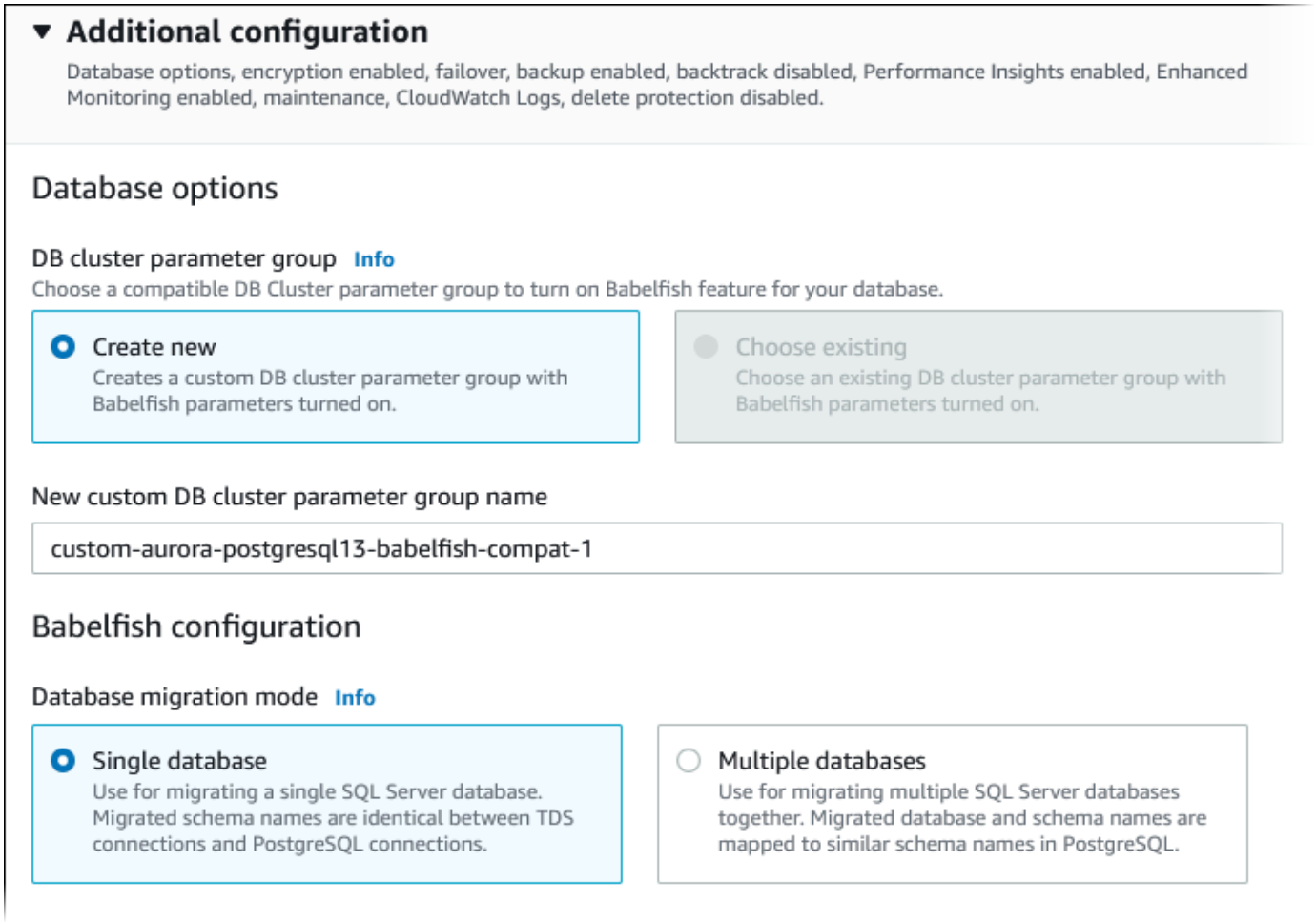
13. Per Database migration mode (Modalità di migrazione del database), scegliere una delle seguenti opzioni:

    - Database singolo per migrare un singolo database Server. SQL

In alcuni casi, potresti migrare più database di utenti insieme, con l'obiettivo finale di una migrazione completa alla versione nativa di Aurora SQL Postgre senza Babelfish. Se le applicazioni finali richiedono schemi consolidati (un unico dbo schema), assicuratevi di consolidare innanzitutto i database Server in un unico database server. SQL SQL Quindi effettuare la migrazione a Babelfish usando la modalità Database singolo.

- Database multipli per migrare più database SQL Server (provenienti da un'unica installazione del server). SQL La modalità con più database non consolida più database che non provengono da un'unica SQL installazione del server. Per ulteriori informazioni sulla migrazione di più database, consulta Utilizzo di Babelfish con un singolo database o più database.

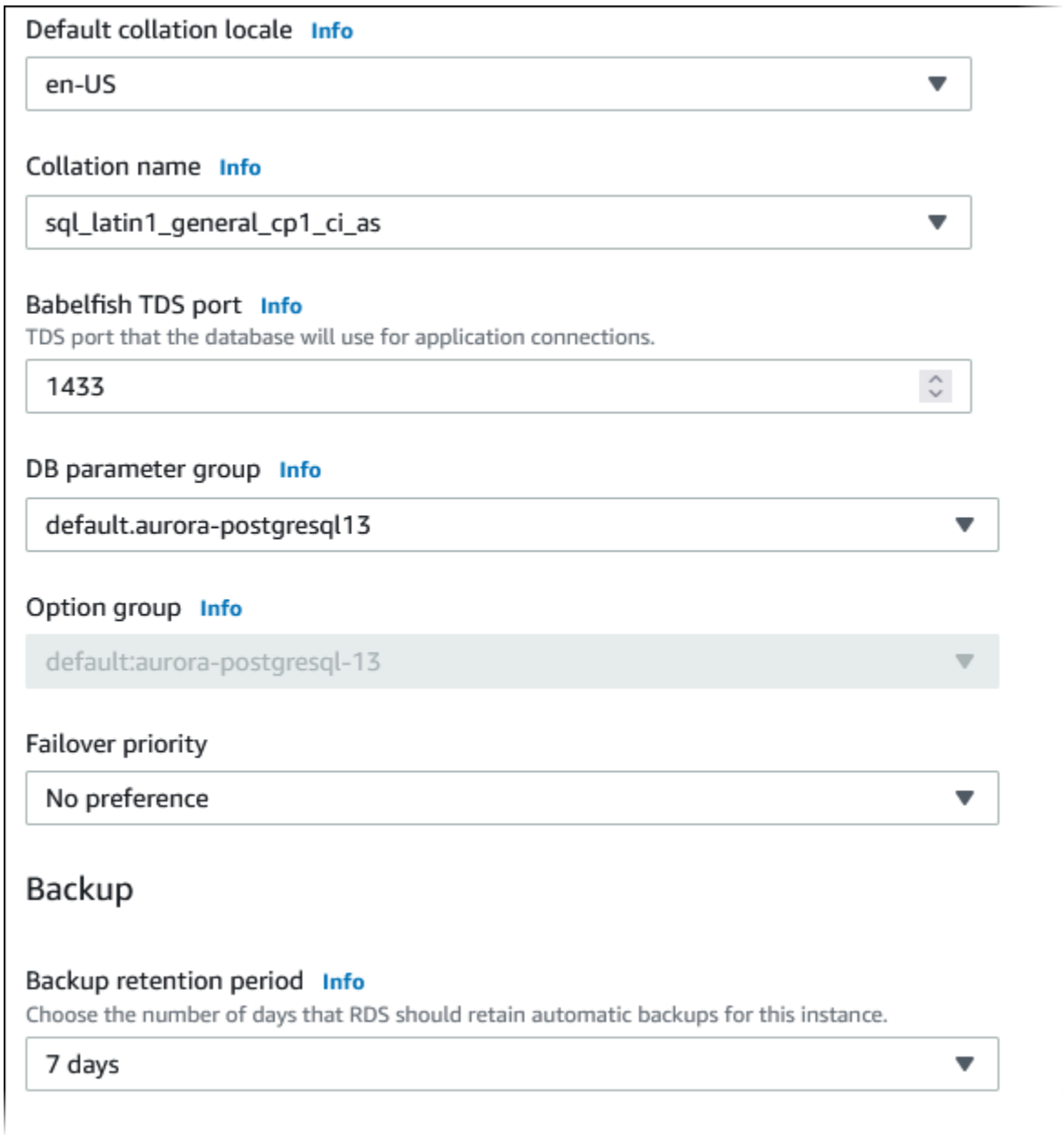
> **Note**
>
> A partire dalla versione Aurora Postgre SQL 16, per impostazione predefinita viene scelta la modalità di migrazione del database Multiple database.



14. Per Localizzazione raccolta predefinita, immetti le impostazioni internazionali del server. Il valore predefinito è `en-US`. Per informazioni dettagliate sulle raccolte consulta Comprendere le collazioni in Babelfish per Aurora Postgre SQL.
15. Per il campo Collation name (Nome raccolta), inserisci le regole di confronto predefinite. Il valore predefinito è `sql_latin1_general_cp1_ci_as`. Per informazioni dettagliate, consulta Comprendere le collazioni in Babelfish per Aurora Postgre SQL.
16. Per la porta Babelfish, inserisci la TDS porta predefinita. `1433` Attualmente, Babelfish supporta solo la porta `1433` per il cluster di database.
17. Per DB parameter group (Gruppo di parametri database), scegli un gruppo di parametri o fai in modo che Aurora crei un nuovo gruppo per te con impostazioni predefinite.
18. Per Failover priority (Priorità failover), seleziona una priorità di failover per l'istanza. Se non specifichi alcun valore, l'impostazione predefinita è `tier-1`. Questa priorità determina l'ordine

di promozione delle repliche durante il ripristino da un errore dell'istanza primaria. Per ulteriori informazioni, consulta Tolleranza ai guasti di un cluster DB Aurora.

19. Per Backup retention period (Tempo di conservazione del backup), scegli l'arco di tempo (da 1 a 35 giorni), nel quale Aurora conserverà le copie di backup del database. È possibile utilizzare copie di backup per point-in-time ripristinare (PITR) il database fino al secondo. Il periodo di conservazione predefinito è 60 giorni.



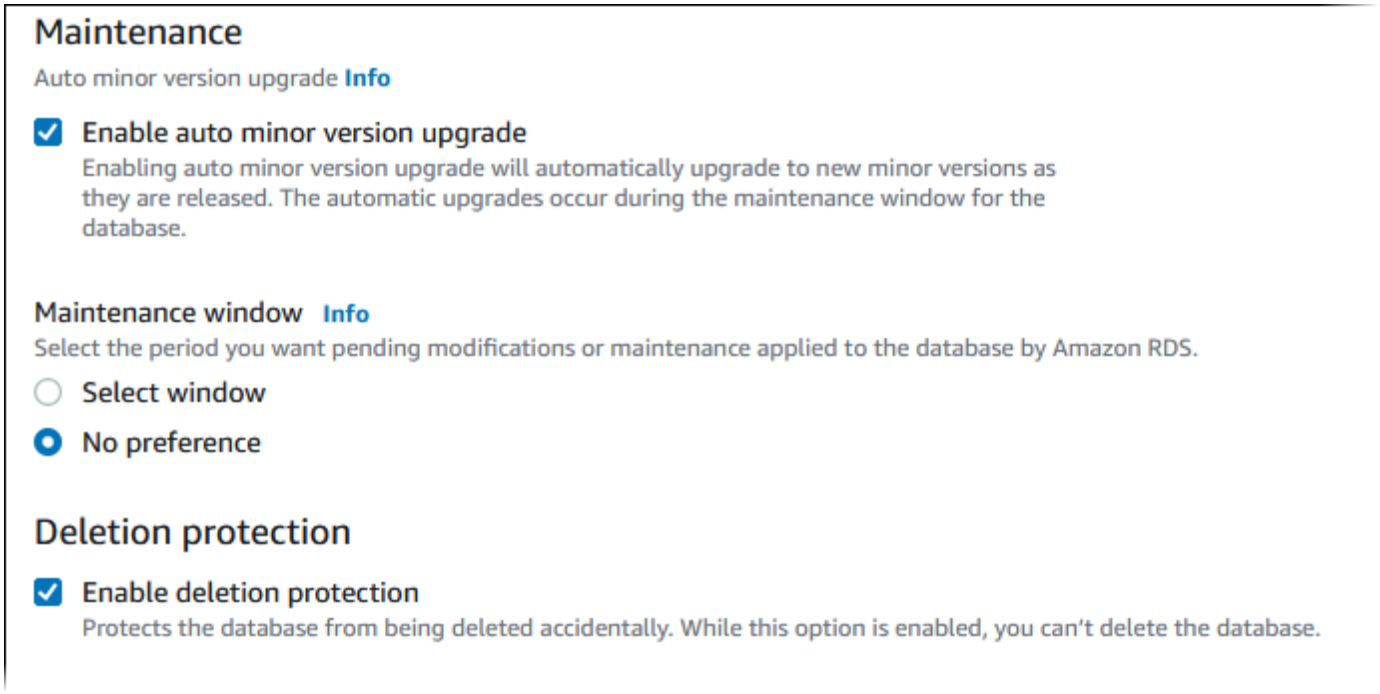
20. Selezionare Copy tags to snapshots (Copy tag a snapshot) per copiare i tag dell'istanza database in uno snapshot DB quando si crea uno snapshot.

> **Note**
>
> Quando si ripristina un cluster DB da un'istantanea, non viene ripristinato come cluster Babelfish for Aurora Postgre DB. SQL È necessario attivare i parametri che controllano le preferenze di Babelfish nel gruppo di parametri del cluster DB per abilitare nuovamente Babelfish. Per ulteriori informazioni sui parametri Babelfish, vedere. Impostazioni del gruppo di parametri del cluster database per Babelfish

21. Scegliere Enable encryption (Abilita crittografia) per attivare la crittografia a riposo (crittografia dello storage Aurora) per questo cluster database.
22. Scegli Enable Performance Insights per attivare Amazon RDS Performance Insights.
23. Scegliere Enable Enhanced monitoring (Abilita monitoraggio avanzato) per avviare la raccolta di parametri in tempo reale per il sistema operativo su cui viene eseguito il cluster di database.
24. Scegli Postgre SQL log per pubblicare i file di registro su Amazon CloudWatch Logs.
25. Scegliere Enable auto minor version upgrade (Abilita aggiornamento automatico versione secondaria) per aggiornare automaticamente il cluster Aurora DB quando è disponibile un aggiornamento di versione secondaria.
26. Per Maintenance window (Finestra di manutenzione), eseguire le seguenti operazioni:

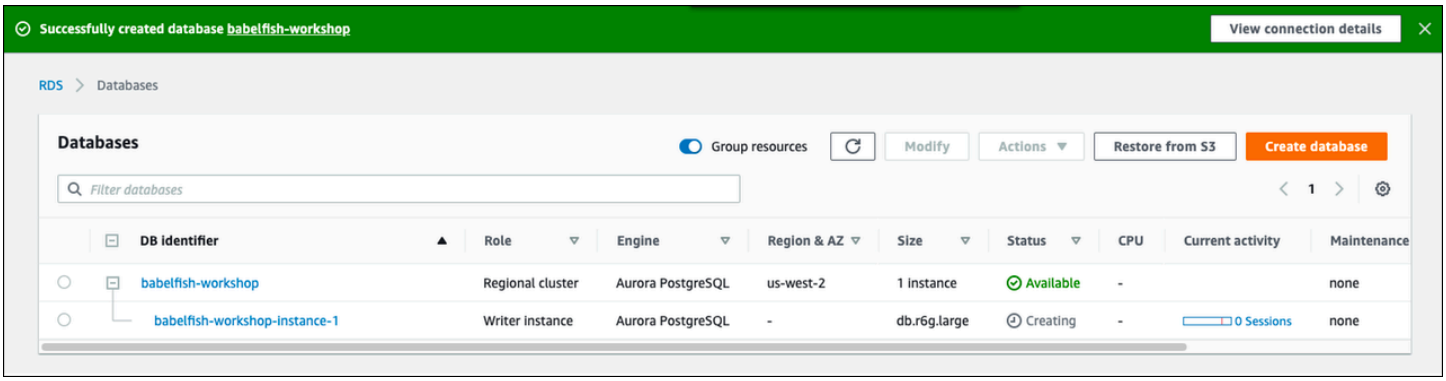
    - Per scegliere un orario in cui Amazon RDS deve apportare modifiche o eseguire la manutenzione, scegli Seleziona finestra.
    - Per eseguire la RDS manutenzione di Amazon in un momento non programmato, scegli Nessuna preferenza.
27. Selezionare la casella Enable deletion protection (Abilita protezione da eliminazione) per proteggere il database dall'eliminazione accidentale.

    Se si attiva questa funzione, non è possibile eliminare direttamente il database. Al contrario, è necessario modificare il cluster di database e disattivare questa funzione prima di eliminare il database.



28. Scegliere Crea database.

Puoi trovare il tuo nuovo database configurato per Babelfish nell'elenco Database. La colonna Status (Stato) visualizza Available (Disponibile) quando la distribuzione è completata.



AWS CLI

Quando crei un Babelfish per Aurora Postgre; usando il SQL AWS CLI, è necessario passare al comando il nome del gruppo di parametri del cluster DB da utilizzare per il cluster. Per ulteriori informazioni, consulta Prerequisiti per i cluster di database.

Prima di poter utilizzare il AWS CLI per creare un SQL cluster Aurora Postgre con Babelfish, procedi come segue:

- Scegli il tuo endpoint URL dall'elenco dei servizi degli endpoint e delle quote di Amazon Aurora.

- Creare gruppo di parametri del cluster di database Per ulteriori informazioni sui gruppi di parametri, consultare .
- Modificare il gruppo di parametri, aggiungendo il parametro che attiva Babelfish.

Per creare un cluster Aurora Postgre SQL DB con Babelfish usando AWS CLI

Gli esempi che seguono utilizzano il nome utente master predefinito, `postgres`. Sostituisci in base alle esigenze con il nome utente creato per il cluster database, ad esempio `sa` o qualsiasi nome utente scelto se non è stato accettato il valore predefinito.

1. Per creare un gruppo di parametri.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --db-parameter-group-family aurora-postgresql14 \
   --description "description"
   ```

   In Windows:

   ```
   aws rds create-db-cluster-parameter-group ^
   --endpoint-url endpoint-URL ^
   --db-cluster-parameter-group-name parameter-group ^
   --db-parameter-group-family aurora-postgresql14 ^
   --description "description"
   ```

2. Modifica il gruppo di parametri per attivare Babelfish.

   In Linux, macOS, oppure Unix:

   ```
   aws rds modify-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --parameters
    "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
   ```

   In Windows:

```
aws rds modify-db-cluster-parameter-group ^
--endpoint-url endpoint-url ^
--db-cluster-parameter-group-name paramater-group ^
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

3. Identifica il tuo sottorete DB e l'ID del gruppo di sicurezza del cloud privato virtuale (VPC) per il nuovo cluster DB, quindi richiamate il create-db-clustercomando.

   In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier  cluster-name\
--master-username postgres \
--manage-master-user-password \
--engine aurora-postgresql \
--engine-version 14.3            \
--vpc-security-group-ids security-group \
--db-subnet-group-name subnet-group-name \
--db-cluster-parameter-group-name parameter-group
```

   In Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier cluster-name ^
--master-username postgres ^
--manage-master-user-password ^
--engine aurora-postgresql ^
--engine-version 14.3 ^
--vpc-security-group-ids security-group ^
--db-subnet-group-name subnet-group ^
--db-cluster-parameter-group-name parameter-group
```

   In questo esempio è specificata l'opzione `--manage-master-user-password` per generare la password dell'utente master e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

4. Crea in maniera esplicita l'istanza primaria del cluster database. Utilizzate il nome del cluster creato nel passaggio 3 per l'`--db-cluster-identifier`argomento quando chiamate il create-db-instancecomando, come illustrato di seguito.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-instance \
   --db-instance-identifier instance-name \
   --db-instance-class db.r6g \
   --db-subnet-group-name subnet-group \
   --db-cluster-identifier cluster-name \
   --engine aurora-postgresql
   ```

   In Windows:

   ```
   aws rds create-db-instance ^
   --db-instance-identifier instance-name ^
   --db-instance-class db.r6g ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-identifier cluster-name ^
   --engine aurora-postgresql
   ```

Migrazione di un database SQL Server a Babelfish per Aurora Postgre SQL

Puoi usare Babelfish for Aurora Postgre SQL per migrare un database SQL Server in un cluster Amazon Aurora Postgre DB. SQL Prima di eseguire la migrazione, consulta Utilizzo di Babelfish con un singolo database o più database.

Argomenti

- Panoramica del processo di migrazione
- Valutazione e gestione delle differenze tra Server e Babelfish SQL
- Strumenti di importazione/esportazione per la migrazione da Server a Babelfish SQL

Panoramica del processo di migrazione

Il seguente riepilogo elenca i passaggi necessari per migrare correttamente l'applicazione Server e farla funzionare con Babelfish. SQL Per informazioni sugli strumenti che puoi utilizzare per i processi di esportazione e importazione e per ulteriori dettagli, consulta Strumenti di importazione/esportazione per la migrazione da Server a Babelfish SQL. Per caricare i dati, consigliamo di utilizzare AWS DMS con un cluster Aurora Postgre SQL DB come endpoint di destinazione.

1. Crea un nuovo cluster Aurora Postgre SQL DB con Babelfish attivato. Per scoprire come, consulta Creazione di un cluster Babelfish per Aurora Postgre DB SQL.

   Per importare i vari SQL artefatti esportati dal database SQL Server, connettiti al cluster Babelfish utilizzando uno strumento Server come sqlcmd. SQL Per ulteriori informazioni, consulta Utilizzo di un client SQL Server per la connessione al cluster database.
2. Sul database SQL Server che desideri migrare, esporta il linguaggio di definizione dei dati (). DDL DDLÈ un SQL codice che descrive gli oggetti del database che contengono dati utente (come tabelle, indici e viste) e il codice del database scritto dall'utente (come le stored procedure, le funzioni definite dall'utente e i trigger).

   Per ulteriori informazioni, consulta Utilizzo di SQL Server Management Studio () per migrare a Babelfish SSMS.
3. Esegui uno strumento di valutazione per valutare l'ambito di eventuali modifiche che potresti dover apportare in modo che Babelfish possa supportare efficacemente l'applicazione in esecuzione su Server. SQL Per ulteriori informazioni, consulta Valutazione e gestione delle differenze tra Server e Babelfish SQL.

4. Rivedi il AWS DMS individua le limitazioni degli endpoint e aggiorna lo DDL script se necessario. Per ulteriori informazioni, consulta Limitazioni all'utilizzo di un endpoint di SQL destinazione Postgre con le tabelle Babelfish in Uso di Babelfish per Aurora Postgre come target. SQL
5. Sul tuo nuovo cluster Babelfish DB, esegui il file DDL all'interno del SQL database T specificato per creare solo gli schemi, i tipi di dati definiti dall'utente e le tabelle con i relativi vincoli di chiave principale.
6. Utilizzo AWS DMS per migrare i dati dal Server alle tabelle Babelfish. SQL Per la replica continua con SQL Server Change Data Capture or SQL Replication, usa Aurora Postgre SQL anziché Babelfish come endpoint. Per farlo, vedi Usare Babelfish per Aurora SQL Postgre come obiettivo per AWS Database Migration Service.
7. Al termine del caricamento dei dati, crea tutti gli SQL oggetti T rimanenti che supportano l'applicazione sul tuo cluster Babelfish.
8. Riconfigura l'applicazione client per connetterti all'endpoint Babelfish anziché al database Server. SQL Per ulteriori informazioni, consulta Connessione a un cluster database Babelfish.
9. Modifica l'applicazione come richiesto e ripeti il test. Per ulteriori informazioni, consulta Differenze tra Babelfish per Aurora PostgreSQL e SQL Server.

Devi comunque valutare le tue domande sul lato client. SQL Gli schemi generati dall'istanza SQL Server convertono solo il codice lato server. SQL Si consiglia di effettuare la procedura seguente:

- Acquisisci le query lato client utilizzando SQL Server Profiler con il modello predefinito _Replay. TSQL Questo modello acquisisce informazioni sull'SQListruzione T che è quindi possibile riprodurre per l'ottimizzazione e il test iterativi. È possibile avviare il profiler in SQL Server Management Studio dal menu Strumenti. Scegliete SQLServer Profiler per aprire il profiler e scegliete il TSQL modello _Replay.

  Da utilizzare per la migrazione di Babelfish, avvia una traccia e quindi esegui l'applicazione utilizzando i test funzionali. Il profiler acquisisce le istruzioni T. SQL Al termine del test, interrompi la traccia. Salva il risultato in un XML file con le tue query sul lato client (File > Salva con nome > Trace XML File for Replay).

  Per ulteriori informazioni, vedere SQLServer Profiler nella documentazione Microsoft. Per ulteriori informazioni sul modello TSQL _Replay, consulta SQLServer Profiler Templates.
- Per le applicazioni con SQL query complesse sul lato client, si consiglia di utilizzare Babelfish Compass per analizzare queste query per verificarne la compatibilità con Babelfish. Se l'analisi

indica che le SQL istruzioni lato client contengono SQL funzionalità non supportate, esaminate gli aspetti dell'applicazione client e modificatele secondo necessità. SQL

- È inoltre possibile acquisire le SQL query come eventi estesi (formato.xel). A tale scopo, utilizzate il Profiler. SSMS XEvent Dopo aver generato il file.xel, estrai le SQL istruzioni in file.xml che Compass può quindi elaborare. Per ulteriori informazioni, consulta Use the SSMS XEvent Profler nella documentazione Microsoft.

Se sei soddisfatto di tutti i test, dell'analisi e delle eventuali modifiche necessarie all'applicazione migrata, puoi iniziare a utilizzare il database Babelfish in produzione. A tale scopo, interrompi il database originale e reindirizza le applicazioni client live per utilizzare la porta Babelfish. TDS

> **Note**
>
> AWS DMS ora supporta la replica dei dati da Babelfish. Per ulteriori informazioni, consulta AWS DMS ora supporta Babelfish for Aurora SQL Postgre come sorgente.

Valutazione e gestione delle differenze tra Server e Babelfish SQL

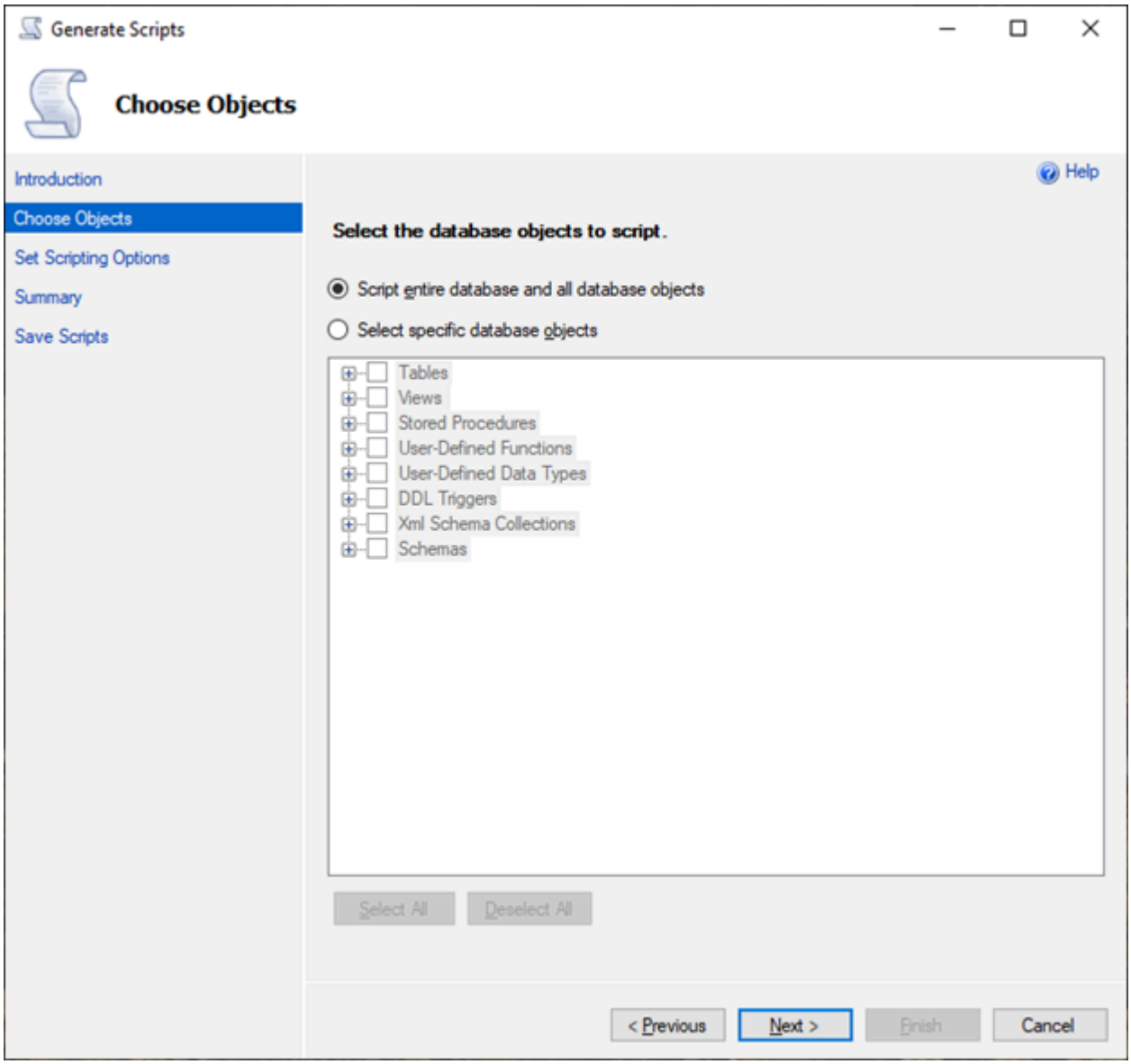
Per ottenere risultati ottimali, ti consigliamo di valutare il codice di query generatoDDL/DMLe il codice di query del client prima di migrare effettivamente l'applicazione di database SQL Server su Babelfish. A seconda della versione di Babelfish e delle funzionalità specifiche di SQL Server implementate dall'applicazione, potrebbe essere necessario rifattorizzare l'applicazione o utilizzare alternative per funzionalità che non sono ancora completamente supportate in Babelfish.

- Per valutare il codice dell'applicazione SQL Server, utilizzate Babelfish Compass sul codice generato DDL per determinare la quantità di codice T supportata da Babelfish. SQL Identifica il SQL codice T che potrebbe richiedere modifiche prima di essere eseguito su Babelfish. Per ulteriori informazioni su questo strumento, consulta lo strumento Babelfish Compass su. GitHub

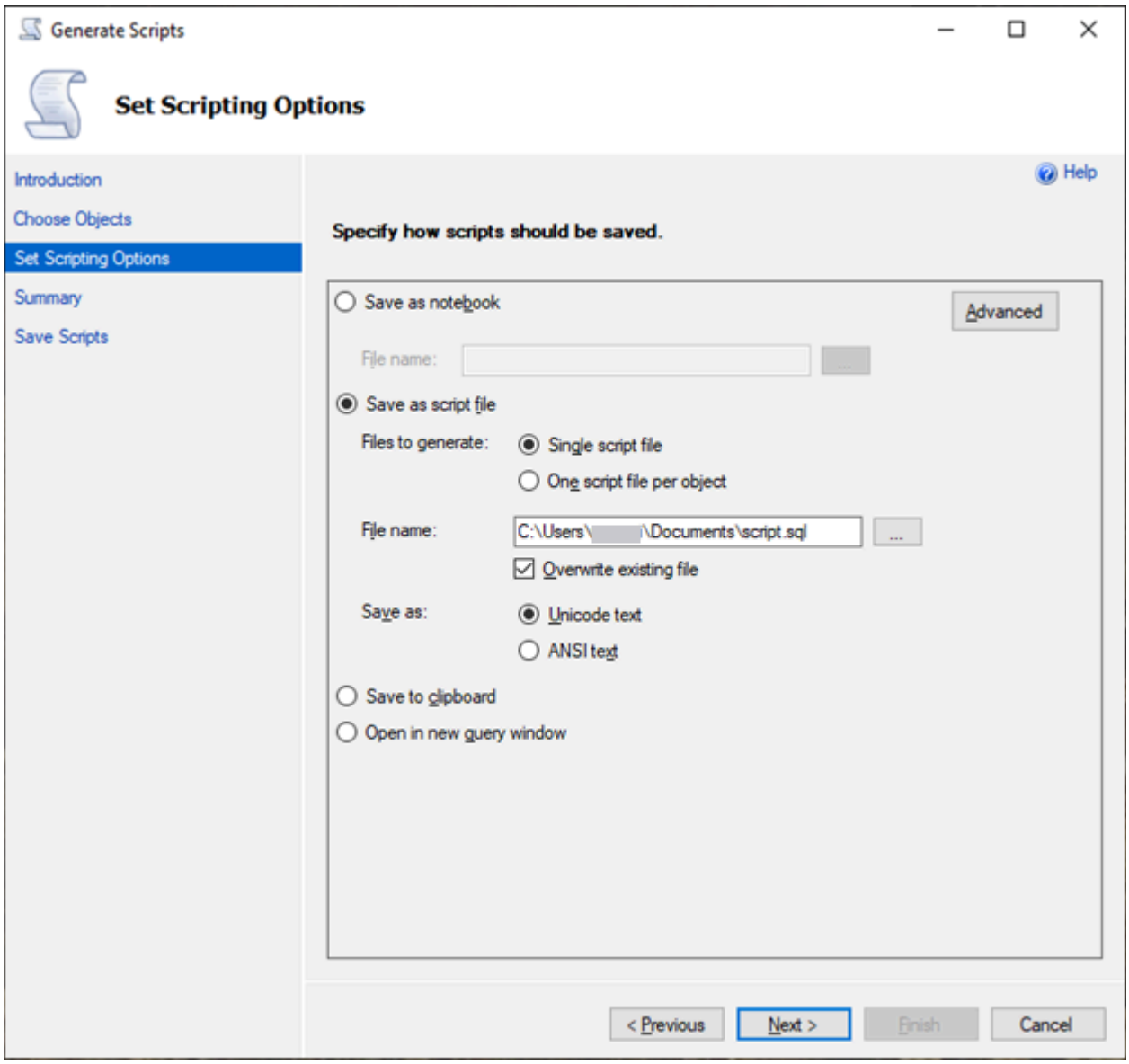
  > **Note**
  >
  > Babelfish Compass è uno strumento open source. Segnala eventuali problemi con Babelfish Compass tramite anziché tramite GitHub AWS Support.

È possibile utilizzare Generate Script Wizard con SQL Server Management Studio (SSMS) per generare il SQL file valutato da Babelfish Compass o AWS Schema Conversion Tool CLI. Consigliamo di effettuare i seguenti passaggi per semplificare la valutazione.

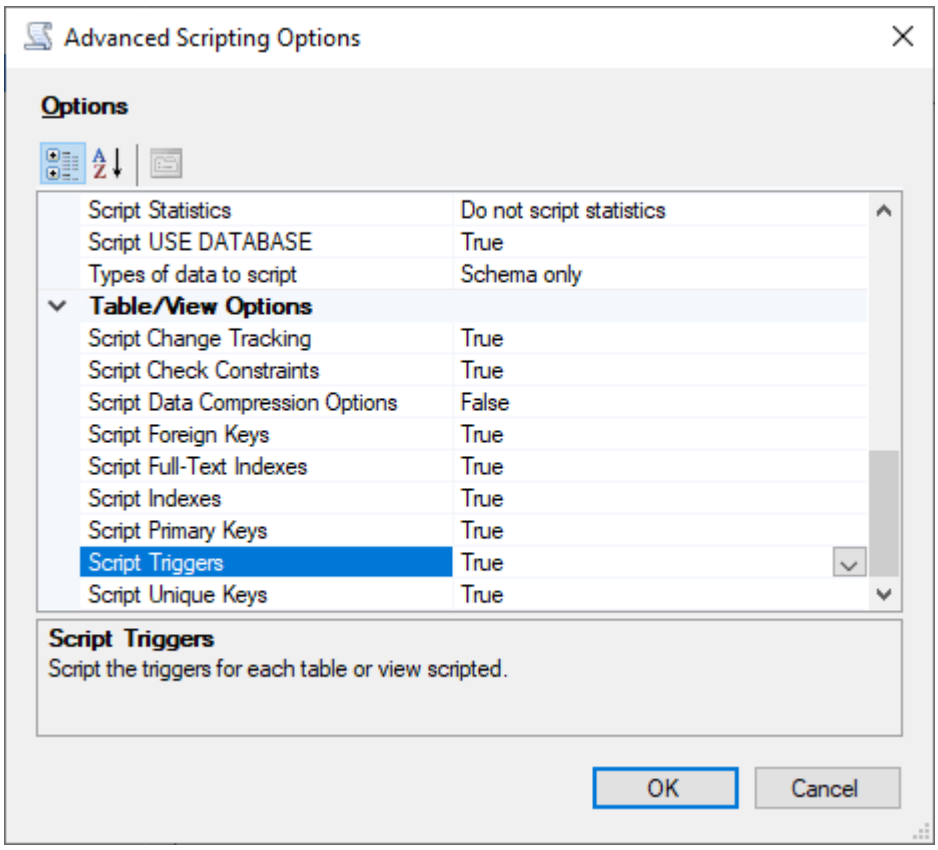
1. Nella pagina Choose Objects (Scegli oggetti), scegli Script entire database and all database objects (Script per l'intero database e tutti gli oggetti del database).



2. Per Set Scripting Options (Imposta le opzioni di script), scegli Save as script file (Salva come file script) come Single script file (File script singolo).



3. Scegli Advanced (Avanzato) per modificare le opzioni di script predefinite e identificare le funzionalità che normalmente sono impostate su false per una valutazione completa:
   - Script Change Tracking (Monitoraggio delle modifiche allo script) su True
   - Script Full-Text Indexes (Indici di testo completo dello script) su True
   - Script Triggers (Trigger script) su True
   - Script Logins (Accessi di script) su True
   - Script Owner (Proprietario script) su True
   - Script Object-Level Permissions (Autorizzazioni a livello di oggetto script) su True
   - Script Collations (Regole di confronto script) su True



4. Segui i restanti passaggi della procedura guidata per generare il file.

Strumenti di importazione/esportazione per la migrazione da Server a Babelfish SQL

Ti consigliamo di utilizzare AWS DMS come strumento principale per la migrazione da SQL Server a Babelfish. Tuttavia, Babelfish supporta diversi altri modi per migrare i dati utilizzando gli strumenti SQL Server che includono quanto segue.

- SQLServer Integration Services (SSIS) per tutte le versioni di Babelfish. Per ulteriori informazioni, consulta Migrare dal SQL server ad Aurora SQL SSIS Postgre utilizzando Babelfish.
- Usa l'SSMSImport/Export Wizard per le versioni 2.1.0 e successive di Babelfish. Questo strumento è disponibile tramiteSSMS, ma è disponibile anche come strumento autonomo. Per ulteriori informazioni, vedere Welcome to SQL Server Import and Export Wizard nella documentazione Microsoft.
- L'utilità Microsoft Bulk Data Copy Program (bcp) consente di copiare i dati da un'istanza di Microsoft SQL Server a un file di dati nel formato specificato. Per ulteriori informazioni, consulta bpc Utility nella documentazione di Microsoft. Babelfish ora supporta la migrazione dei dati tramite

il BCP client e l'utilità bcp ora supporta `-E` flag (per le colonne di identità) e -b flag (per gli inserti in batch). Alcune opzioni bcp non sono supportate, incluse `-C`, `-T`, `-G`, `-K`, `-R`, `-V` e `-h`.

Utilizzo di SQL Server Management Studio () per migrare a Babelfish SSMS

Si consiglia di generare file separati per ciascun tipo di oggetto specifico. È possibile utilizzare prima la procedura guidata Generate Scripts SSMS per ogni set di DDL istruzioni, quindi modificare gli oggetti come gruppo per risolvere eventuali problemi rilevati durante la valutazione.

Esegui questi passaggi per migrare i dati utilizzando AWS DMS o altri metodi di migrazione dei dati. Esegui prima questi tipi di script di creazione per un approccio migliore e più rapido al caricamento dei dati sulle tabelle Babelfish in Aurora Postgre. SQL

1. Esegui le istruzioni `CREATE SCHEMA`.
2. Esegui le istruzioni `CREATE TYPE` per creare i tipi di dati definiti dall'utente.
3. Esegui le istruzioni `CREATE TABLE` di base con le chiavi primarie o i vincoli univoci.

Esegui il caricamento dei dati utilizzando lo strumento di importazione/esportazione consigliato. Esegui gli script modificati per i passaggi seguenti in modo da aggiungere gli oggetti del database rimanenti. Sono necessarie le istruzioni create table per eseguire questi script per i vincoli, i trigger e gli indici. Dopo la generazione degli script, elimina le istruzioni create table.

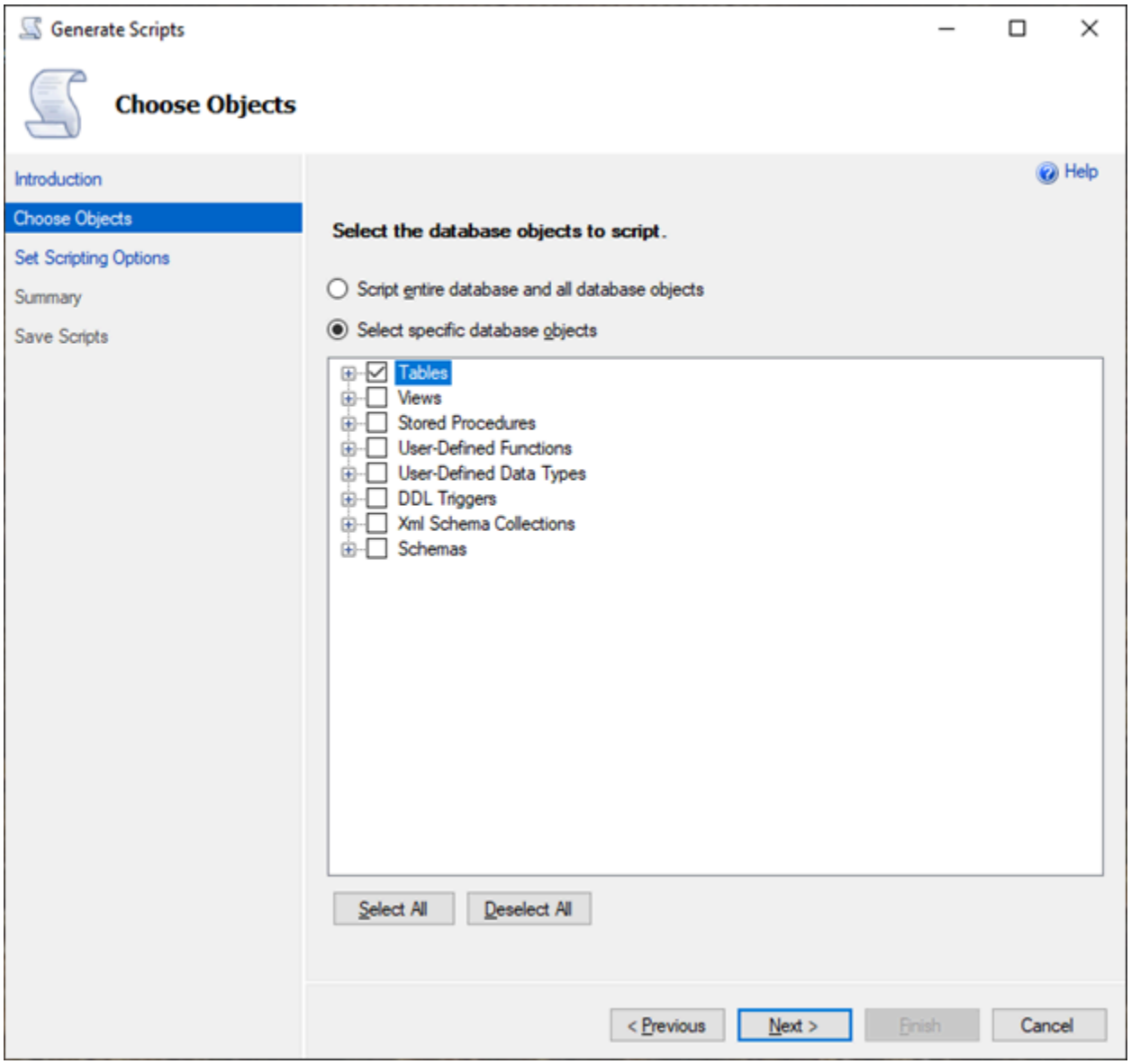
1. Esegui le istruzioni `ALTER TABLE` per i vincoli di controllo, i vincoli di chiave esterna, i vincoli predefiniti.
2. Esegui le istruzioni `CREATE TRIGGER`.
3. Esegui le istruzioni `CREATE INDEX`.
4. Esegui le istruzioni `CREATE VIEW`.
5. Esegui le istruzioni `CREATE STORED PROCEDURE`.

Per generare gli script per ogni tipo di oggetto

Utilizza i seguenti passaggi per creare le istruzioni di base per la creazione di tabelle utilizzando la procedura guidata Generate Scripts in. SSMS Segui la stessa procedura per generare gli script per i diversi tipi di oggetto.

1. Connect all'istanza SQL Server esistente.

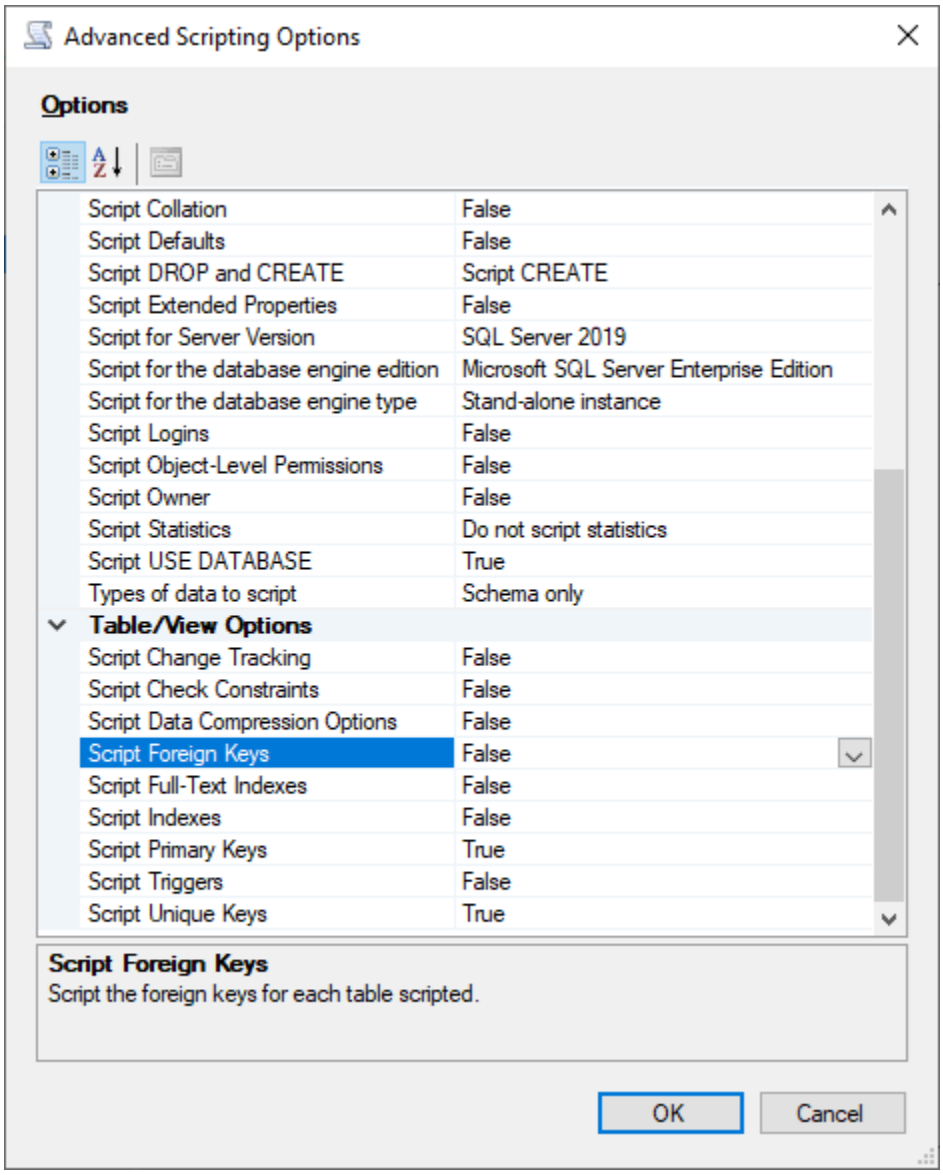
2. Apri il menu contestuale (tasto destro del mouse) per il nome di un database.
3. Scegli Tasks (Attività), quindi seleziona Generate Scripts... (Genera script...).
4. Nel riquadro Choose Objects (Scegli oggetti), seleziona Select specific database objects (Seleziona oggetti di database specifici). Scegli Tables (Tabelle), seleziona tutte le tabelle. Seleziona Successivo per continuare.



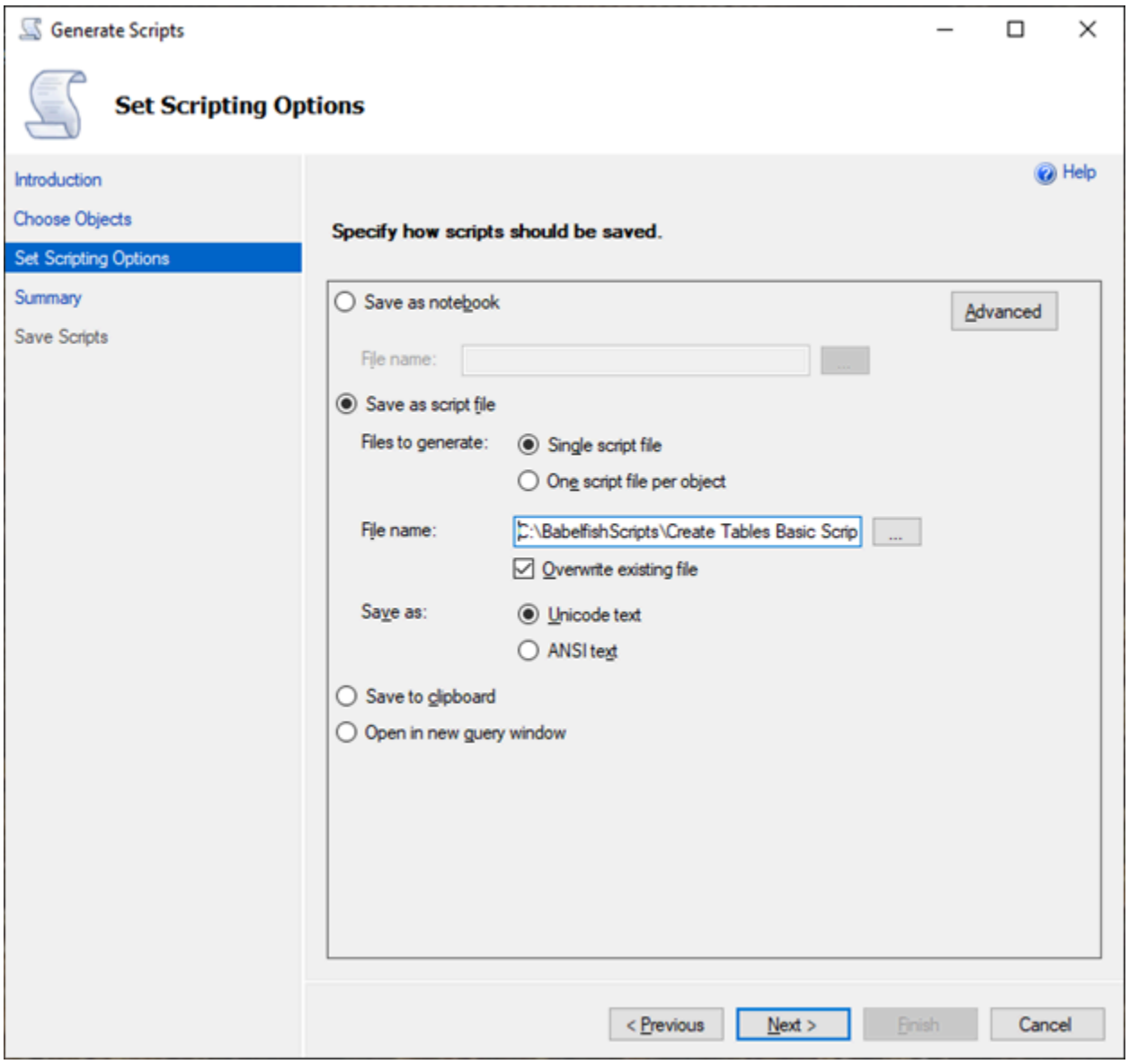
5. Nella pagina Set Scripting Options (Imposta opzioni di scripting), scegli Advanced (Avanzato) per aprire le impostazioni Options (Opzioni). Per generare le istruzioni di base create table, modifica i seguenti valori predefiniti:

   - Script Defaults (Valori predefiniti script) su False.

- Script Extended Properties (Proprietà estese script) su False. Babelfish non supporta le proprietà estese.
- Script Check Constraints (Vincoli di controllo script) su False. Script Foreign Keys (Chiavi esterne script) su False.



6. Scegli OK.
7. Nella pagina Set Scripting Options (Imposta le opzioni di script), scegli Save as script file (Salva come file script), quindi seleziona l'opzione Single script file (File script singolo). Immetti un valore nel campo File name (Nome file).



8. Scegli Next (Successivo) per visualizzare la pagina Summary wizard (Procedura guidata di riepilogo).
9. Scegli Next (Successivo) per avviare la generazione dello script.

   È possibile continuare a generare script per gli altri tipi di oggetti nella procedura guidata. Anziché scegliere Finish (Fine) dopo il salvataggio del file, seleziona il pulsante Previous (Precedente) per tre volte per tornare alla pagina Choose Objects (Scegli oggetti). Ripetete quindi i passaggi della procedura guidata per generare script per gli altri tipi di oggetti.

Autenticazione del database con Babelfish per Aurora Postgre SQL

Babelfish for Aurora Postgre SQL supporta due modi per autenticare gli utenti del database. L'autenticazione con password è disponibile per impostazione predefinita per tutti i cluster database Babelfish. È possibile aggiungere l'autenticazione Kerberos per lo stesso cluster database.

Argomenti

- Autenticazione password con Babelfish
- Autenticazione Kerberos con Babelfish
- Configurazione dell'autenticazione Kerberos utilizzando i gruppi di sicurezza Active Directory per Babelfish

Autenticazione password con Babelfish

Babelfish for Aurora SQL Postgre supporta l'autenticazione tramite password. Le password vengono memorizzate in forma crittografata su disco. Per ulteriori informazioni sull'autenticazione su un cluster Aurora PostgreSQL, vedere. Sicurezza con Amazon Aurora PostgreSQL

Potrebbero essere richieste credenziali ogni volta che ti connetti a Babelfish. Qualsiasi utente migrato o creato su Aurora SQL Postgre può utilizzare le stesse credenziali sia sulla porta Server che SQL sulla porta Postgre. SQL Babelfish non applica le policy sulle password, ma è consigliabile effettuare le seguenti operazioni:

- Richiedi una password complessa contenente almeno otto (8) caratteri.
- Applica un criterio di scadenza della password.

Per esaminare un elenco completo degli utenti del database, utilizza il comando `SELECT * FROM pg_user;`.

Autenticazione Kerberos con Babelfish

La versione SQL 15.2 di Babelfish per Aurora Postgre supporta l'autenticazione al cluster DB tramite Kerberos. Questo metodo ti consente di usare l'autenticazione di Microsoft Windows per gli utenti che si connettono al database Babelfish, ma A tale scopo, è necessario innanzitutto configurare il cluster DB da utilizzare AWS Directory Service for Microsoft Active Directory per l'autenticazione Kerberos. Per ulteriori informazioni, vedi Cos'è AWS Directory Service? nel AWS Directory Service Guida all'amministrazione.

Configurazione dell'autenticazione Kerberos

Il cluster Babelfish for Aurora Postgre SQL DB può connettersi utilizzando due porte diverse, ma la configurazione dell'autenticazione Kerberos è un processo unico. Pertanto, per prima cosa è necessario configurare l'autenticazione Kerberos per il cluster database. Per ulteriori informazioni, consulta Configurazione dell'autenticazione Kerberos. Dopo aver completato la configurazione, assicurati di poterti connettere con un client Postgre utilizzando Kerberos. SQL Per ulteriori informazioni, consulta Connessione con Autenticazione Kerberos.

Accesso e allocazione degli utenti in Babelfish

Gli accessi a Windows creati dalla porta Tabular Data Stream (TDS) possono essere utilizzati con la porta o con la TDS porta Postgre. SQL Innanzitutto, l'accesso che può utilizzare Kerberos per l'autenticazione deve essere fornito dalla TDS porta prima di essere utilizzato dagli SQL utenti e dalle applicazioni T- per connettersi a un database Babelfish. Durante la creazione di accessi a Windows, gli amministratori possono fornire l'accesso utilizzando il nome di dominio o il nome di DNS dominio Net. BIOS In genere, Net BIOS domain è il sottodominio del nome di dominio. DNS Ad esempio, se il nome di DNS dominio è`CORP.EXAMPLE.COM`, allora il BIOS dominio Net può essere`CORP`. Se viene fornito il formato del nome di BIOS dominio Net per un accesso, deve esistere una mappatura al nome di DNS dominio.

Gestione della mappatura tra nomi di BIOS dominio di rete e nomi di DNS dominio

Per gestire le mappature tra il nome di BIOS dominio Net e il nome di DNS dominio, Babelfish fornisce procedure memorizzate nel sistema per aggiungere, rimuovere e troncare le mappature. Solo un utente con il ruolo `sysadmin` può eseguire queste procedure.

Per creare una mappatura tra Net BIOS e il nome di DNS dominio, usa la procedura memorizzata dal sistema fornita da Babelfish. `babelfish_add_domain_mapping_entry` Entrambi gli argomenti devono avere un valore valido e non lo sono. NULL

Example

```
EXEC babelfish_add_domain_mapping_entry 'netbios_domain_name',
 'fully_qualified_domain_name'
```

L'esempio seguente mostra come creare la mappatura tra il nome di rete CORP e BIOS il nome di DNS dominio. CORP EXAMPLE. COM.

Example

```
EXEC babelfish_add_domain_mapping_entry 'corp', 'corp.example.com'
```

Per eliminare una voce di mappatura esistente, usaRE la stored procedure di sistema babelfish_remove_domain_mapping_entry:

Example

```
EXEC babelfish_remove_domain_mapping_entry 'netbios_domain_name'
```

L'esempio seguente mostra come rimuovere la mappatura per il BIOS nome CORP di rete.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'corp'
```

Per rimuovere tutte le voci di mappatura esistenti, usa la stored procedure di sistema babelfish_truncate_domain_mapping_table:

Example

```
EXEC babelfish_truncate_domain_mapping_table
```

Per visualizzare tutte le mappature tra Net BIOS e nome di DNS dominio, utilizzare la seguente query.

Example

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

Gestione degli accessi

Creazione degli accessi

Connect al DB tramite l'TDSendpoint utilizzando un login con le autorizzazioni corrette. Se non è stato creato un utente del database, l'accesso viene mappato all'utente guest. Se l'utente guest non è abilitato, il tentativo di accesso ha esito negativo.

Crea un accesso Windows utilizzando la seguente query. L'opzione `FROM WINDOWS` consente l'autenticazione tramite Active Directory.

```
CREATE LOGIN login_name FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Example

L'esempio seguente mostra come creare un accesso per l'utente di Active Directory [corp\test1] con il database predefinito db1.

```
CREATE LOGIN [corp\test1] FROM WINDOWS WITH DEFAULT_DATABASE=db1
```

Questo esempio presuppone che esista una mappatura tra il BIOS dominio Net CORP e il nome di dominio. DNS CORP EXAMPLE. COM. Se non è presente alcuna mappatura, è necessario fornire il nome di DNS dominio [CORP. EXAMPLE. COM\ test1].

> **Note**
>
> Gli accessi basati sugli utenti di Active Directory sono limitati a nomi contenenti meno di 21 caratteri.

Eliminazione di un accesso

Per eliminare un accesso, usa la stessa sintassi utilizzata per qualsiasi accesso, come mostrato nell'esempio seguente:

```
DROP LOGIN [DNS domain name\login]
```

Modifica di un accesso

Per modificare un accesso, usa la stessa sintassi utilizzata per qualsiasi accesso, come mostrato nell'esempio seguente:

```
ALTER LOGIN [DNS domain name\login] { ENABLE|DISABLE|WITH DEFAULT_DATABASE=[master] }
```

Il ALTER LOGIN comando supporta opzioni limitate per gli accessi a Windows, tra cui:

- DISABLE— Per disabilitare un accesso. Non puoi usare un accesso disabilitato per l'autenticazione.
- ENABLE— Per abilitare un accesso disabilitato.
- DEFAULT_ DATABASE — Per modificare il database predefinito di un login.

> **Note**
>
> Tutta la gestione delle password viene eseguita tramite AWS Directory Service, quindi il ALTER LOGIN comando non consente agli amministratori del database di modificare o impostare le password per gli accessi a Windows.

Connessione a Babelfish per Aurora Postgre con autenticazione Kerberos SQL

In genere, gli utenti del database eseguono l'autenticazione tramite Kerberos usando computer client che sono membri del dominio Active Directory. Utilizzano l'autenticazione di Windows dalle loro applicazioni client per accedere al server Babelfish for Aurora SQL Postgre sulla porta. TDS

Connessione a Babelfish per Aurora Postgre SQL sulla porta Postgre con autenticazione Kerberos SQL

È possibile utilizzare gli accessi creati dalla porta con la porta o con la TDS porta Postgre. TDS SQL Tuttavia, Postgre SQL utilizza per impostazione predefinita i confronti con distinzione tra maiuscole e minuscole per i nomi utente. Affinché Aurora Postgre SQL interpreti i nomi utente Kerberos senza distinzione tra maiuscole e minuscole, è necessario impostare il parametro come nel gruppo di parametri personalizzato del `krb_caseins_users` cluster Babelfish. `true` Questo parametro è impostato su `false` per impostazione predefinita. Per ulteriori informazioni, consulta Configurazione per nomi utente senza distinzione tra maiuscole e minuscole. Inoltre, è necessario specificare il nome utente di accesso nel formato <login@ nome di dominio> dalle applicazioni client Postgre. DNS SQL Non è possibile utilizzare il formato < domain name\ login>. DNS

Errori frequenti

È possibile configurare relazioni di trust forestale tra Microsoft Active Directory locale e AWS Managed Microsoft AD. Per ulteriori informazioni, consulta Creare una relazione di fiducia. Quindi, dovrai connetterti utilizzando un endpoint specifico del dominio specializzato anziché utilizzare il dominio Amazon `rds.amazonaws.com` nell'endpoint host. Se non utilizzi l'endpoint specifico del dominio corretto, potresti riscontrare il seguente errore:

```
Error: “Authentication method "NTLMSSP" not supported (Microsoft SQL Server, Error:
 514)"
```

Questo errore si verifica quando il TDS client non riesce a memorizzare nella cache il ticket di servizio per l'endpoint URL fornito. Per ulteriori informazioni, consulta Connessione con Kerberos.

Configurazione dell'autenticazione Kerberos utilizzando i gruppi di sicurezza Active Directory per Babelfish

A partire dalla versione 4.2.0 di Babelfish, puoi configurare l'autenticazione Kerberos per Babelfish con gruppi di sicurezza Active Directory. Di seguito sono riportati i prerequisiti da completare per la configurazione dell'autenticazione Kerberos tramite Active Directory:

- È necessario seguire tutti i passaggi indicati in. Autenticazione Kerberos con Babelfish
- Assicurati che l'istanza DB sia associata ad Active Directory. Per verificarlo, puoi visualizzare lo stato dell'appartenenza al dominio nella console o eseguendo il describe-db-instances AWS CLIcomando.

  Lo stato dell'istanza DB deve essere abilitato per Kerberos. Per ulteriori informazioni sulla comprensione dell'appartenenza al dominio, vedere. Appartenenza al dominio
- Verifica le mappature tra il nome di BIOS dominio di rete e il nome di DNS dominio utilizzando la seguente query:

  ```
  SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
  ```
- Prima di procedere ulteriormente, verifica che l'autenticazione Kerberos tramite login individuale funzioni come previsto. La connessione tramite l'autenticazione Kerberos come utente di Active Directory dovrebbe avere esito positivo. In caso di problemi, consulta. Errori frequenti

Configurazione dell'estensione pg_ad_mapping

È necessario seguire tutti i passaggi indicati in. Configurazione dell'estensione pg_ad_mapping Per verificare che l'estensione sia installata, esegui la seguente query dall'TDSendpoint:

```
1> SELECT extname, extversion FROM pg_extension where extname like 'pg_ad_mapping';
2> GO
extname       extversion
------------- ----------
pg_ad_mapping 0.1

(1 rows affected)
```

Gestione degli accessi di gruppo

Crea accessi di gruppo seguendo i passaggi indicati in. Gestione degli accessi Si consiglia che il nome di accesso sia lo stesso del nome del gruppo di sicurezza Active Directory (AD) per semplificare la manutenzione, sebbene non sia obbligatorio. Per esempio:

```
CREATE LOGIN [corp\accounts-group] FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Mappatura degli accessi ai gruppi T con il SQL gruppo di sicurezza AD

È necessario fornire esplicitamente T- SQL Windows Group Login per ogni gruppo di sicurezza AD che richiede l'accesso al server del database. Un utente AD, che fa parte di almeno un gruppo di sicurezza AD assegnato, avrà accesso al server del database.

> **Note**
>
> Questo SQL T-login non può più autenticarsi utilizzando un'autenticazione basata su password.

Ad esempio, accounts-group è un gruppo di sicurezza in AD e se desideri fornire questo gruppo di sicurezza in Babelfish, devi usare il formato [corp\ accounts-group].

- Gruppo di sicurezza AD: accounts-group
- TSQLAccesso: [corp\ accounts-group]
- Ruolo PG equivalente per un dato TSQL Login: accounts-group@. CORP EXAMPLE. COM

L'amministratore può ora procedere alla creazione della mappatura tra il gruppo di sicurezza AD e il SQL login T- dall'SQLendpoint Postgre tramite il seguente comando psql. Per ulteriori informazioni sull'utilizzo della funzione, vedere. Utilizzo delle funzioni dell'estensione `pg_ad_mapping`

> **Note**
>
> T- SQL login deve essere specificato nel FQDN formato login_name@ durante l'aggiunta della mappatura. I pesi vengono ignorati durante la connessione tramite endpoint. TDS Per ulteriori informazioni sull'utilizzo dei pesi, vedere. Connessione a Babelfish tramite l'SQLendpoint Postgre sulla porta Postgre SQL

```
postgres=>select pgadmap_set_mapping('accounts-group', 'accounts-
group@CORP.EXAMPLE.COM', <SID>, <Weight>);
```

Per informazioni sul recupero del gruppo SID di sicurezza AD, vedere. Recupero del gruppo Active Directory in SID PowerShell

La tabella seguente mostra un esempio di mappatura dai gruppi di sicurezza AD agli accessi T: SQL

Gruppi di sicurezza AD	TSQLAccessi	Ruolo PG equivalente per un determinato accesso TSQL	Weight
gruppo di conti	[gruppo\ accounts-group]	CORPgruppo-conti@. EXAMPLE. COM	7
gruppo di vendita	[corporazione\ gruppo di vendita]	gruppo CORP di vendita@. EXAMPLE. COM	10
gruppo di sviluppo	[gruppo\ dev-group]	CORPgruppo di sviluppo @. EXAMPLE. COM	7

```
postgres=> select admap.ad_sid, admap.ad_grp, lgn.orig_loginname, lgn.rolname,
 admap.weight from pgadmap_read_mapping() as admap, sys.babelfish_authid_login_ext as
 lgn where admap.pg_role = lgn.rolname;
    ad_sid    |     ad_grp     |    orig_loginname   |             rolname
 | weight
--------------+---------------+---------------------+--------------------------------
+--------
 S-1-5-67-890 | accounts-group | corp\accounts-group  | accounts-group@CORP.EXAMPLE.COM
 |     7
 S-1-2-34-560 | sales-group    | corp\sales-group     | sales-group@CORP.EXAMPLE.COM
 |     10
 S-1-8-43-612 | dev-group      | corp\dev-group       | dev-group@CORP.EXAMPLE.COM
 |     7
 (7 rows)
```

Connessione a Babelfish tramite endpoint TDS

Nell'esempio seguente, user1 è membro di accounts-group e sales-group, user2 è membro di accounts-group e dev-group.

Username	Iscrizione ai gruppi di sicurezza AD
user1	gruppo di account, gruppo di vendita
user2	gruppo-conti, gruppo-sviluppo

Connect al server di database Babelfish utilizzando l'utilità sqlcmd. Puoi verificare se un utente (user1 in questo esempio) è stato autenticato utilizzando Kerberos seguendo l'esempio seguente:

```
1> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();
2>  GO
principal               gss_authenticated
----------------------  -----------------
user1@CORP.EXAMPLE.COM  1

((1 rows affected))
1> select suser_name();
2>  GO
suser_name
```

```
----------
corp\user1

(1 rows affected)
```

In questo esempio, user1 erediterà i privilegi di accounts-group e sales-group. È possibile verificare l'appartenenza al gruppo utilizzando la visualizzazione di sistema. `sys.login_token`

```
1> SELECT name, type FROM sys.login_token;
2>  GO
name                type
------------------- ----
corp\accounts-group WINDOWS GROUP
corp\sales-group    WINDOWS GROUP

(2 rows affected)
```

Controllo e registrazione

Per determinare l'identità principale di sicurezza di AD, utilizza i seguenti comandi:

```
1> select suser_name();
2> GO
suser_name
----------
corp\user1

(1 rows affected)
```

Attualmente, l'identità dell'utente AD non è visibile nei registri. È possibile attivare il `log_connections` parametro per registrare l'istituzione della sessione DB. Per ulteriori informazioni, consulta log_connections. L'output include l'identità utente AD come principale, come illustrato nell'esempio seguente. Il backend PID associato a questo output può quindi aiutare ad attribuire le azioni all'utente AD effettivo.

```
bbf_group_ad_login@babelfish_db:[615]:LOG: connection authorized:
 user=bbf_group_ad_login database=babelfish_db application_name=sqlcmd GSS
 (authenticated=yes, encrypted=yes, principal=user1@CORP.EXAMPLE.COM)
```

Utilizzo dei privilegi di appartenenza al gruppo di sicurezza AD

Ereditazione dei privilegi a livello di server

Gli utenti AD che sono membri di un determinato gruppo di sicurezza AD erediteranno i privilegi a livello di server concessi all'accesso del gruppo Windows mappato. Ad esempio, si consideri il gruppo di sicurezza `accounts-group` AD, a cui viene concessa l'appartenenza al ruolo `sysadmin` server su Babelfish. È possibile ereditare i privilegi a livello di server utilizzando il seguente comando:

```
1> ALTER SERVER ROLE sysadmin ADD MEMBER [corp\accounts-group];
```

Di conseguenza, qualsiasi utente di Active Directory che è membro del gruppo di sicurezza `accounts-group` AD erediterà i privilegi a livello di server associati al ruolo. `sysadmin` Ciò significa che un utente di cui fa parte ora avrà la possibilità di `accounts-group` eseguire operazioni a livello di server all'interno di Babelfish. `corp\user1`

> **Note**
>
> Per eseguire l'accesso a livello di serverDDLs, è necessario che esista l'accesso a Windows per singoli utenti AD. Per ulteriori informazioni, consulta Limitazioni.

Ereditazione dei privilegi a livello di database

Per concedere i privilegi a livello di database, è necessario creare e mappare un utente del database con l'accesso di gruppo di Windows. Gli utenti AD che sono membri di un determinato gruppo di sicurezza AD erediteranno i privilegi a livello di database concessi a tale utente del database. Nell'esempio seguente, puoi vedere come vengono assegnati i privilegi a livello di database per il gruppo Windows [corp\ accounts-group].

```
1> CREATE DATABASE db1;
2> GO
1> USE db1;
2> GO
Changed database context to 'db1'.
1> CREATE TABLE dbo.t1(a int);
```

```
2> GO
```

Crea un utente del database [corp\ sales-group] per l'accesso al gruppo Windows [corp\ accounts-group]. Per eseguire questo passaggio, connettiti tramite TDS endpoint utilizzando login who is a member of sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group];
2> GO
```

Ora, connettiti come utente AD user1 per verificare l'accesso alla tabella t1. Poiché non abbiamo ancora concesso i privilegi a livello di database, si verificherà un errore di autorizzazione negata.

```
1> SELECT * FROM dbo.t1;
2> GO
Msg 33557097, Level 16, State 1, Server db-inst, Line 1
permission denied for table t1
```

SELECTConcedi la tabella t1 all'utente del database [corp\ accounts-group]. Per eseguire questo passaggio, connettiti tramite TDS endpoint utilizzando login who is member of sysadmin.

```
1> GRANT SELECT ON dbo.t1 TO [corp\accounts-group];
2> GO
```

Connect come utente AD1 per convalidare l'accesso.

```
1> SELECT * FROM dbo.t1;
2> GO
a
-----------

(0 rows affected)
```

Gestione del comportamento delle DDL dichiarazioni basato su uno schema predefinito o esplicito

Quando si utilizza una sessione autenticata da AD, lo schema predefinito per la sessione corrente è determinato dalle seguenti condizioni:

- Se esiste un singolo utente del database, lo schema predefinito dell'utente viene considerato lo schema predefinito della sessione corrente.
- Se esiste lo schema predefinito per un utente del database di gruppo, lo schema predefinito dell'utente del database di gruppo viene considerato lo schema predefinito della sessione corrente con l'ID principale più piccolo.

Comprensione del comportamento delle CREATE DDL dichiarazioni

Se non è specificato uno schema esplicito nell'CREATEDDListruzione, la creazione dell'oggetto avverrà nello schema predefinito della sessione corrente. Se non è possibile determinare se lo schema è predefinito o esplicito, l'DDListruzione genererà il seguente errore:

```
"Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command."
```

Example : lo schema predefinito non esiste per gli utenti del gruppo Windows

L'utente del gruppo Windows [corp\ accounts-group] ha uno schema NULL predefinito e l'utente AD user1 sta tentando di eseguirlo DDL senza specificare lo schema in modo esplicito. Poiché l'accesso e l'utente a Windows individuali non esistono per user1, otterrà solo i privilegi a livello di database dell'utente del gruppo Windows [corp\ accounts-group].

```
1> create TABLE t2(a int);
2> GO

Msg 33557097, Level 16, State 1, Server db-inst, Line 1
Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command.
```

> **Note**
>
> L'accesso e l'utente Windows individuali non esistono per l'utente AD user1

Example : esiste uno schema predefinito per gli utenti del gruppo Windows

Crea un utente di gruppo Windows per l'accesso a [corp\ accounts-group] con lo schema predefinito utilizzando sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group] WITH
 DEFAULT_SCHEMA = sch_acc;
2> GO
1> CREATE SCHEMA sch_acc AUTHORIZATION [gad\accounts-group];
2> GO
1> SELECT name, principal_id, default_schema_name FROM sys.database_principals WHERE
 name = 'corp\accounts-group';
2> GO

name               principal_id default_schema_name
------------------ ------------ ------------------
corp\accounts-group 24162        sch_acc

(1 rows affected)
```

Prova a creare un oggetto senza specificare lo schema in modo esplicito utilizzando l'utente AD user1. La tabella t2 verrà creata nello schema predefinito di [corp\ accounts-group] Utente del gruppo Windows Il proprietario di questo oggetto sarà lo stesso del proprietario dello schema sch_acc.

```
1> CREATE TABLE t_group(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't_group';
2> GO

name    schema_name
```

```
------- -----------
t_group sch_acc

(1 rows affected)
```

> **Note**
>
> L'accesso e l'utente Windows individuali non esistono per l'utente AD user1

Example : esiste anche un utente del database individuale per un utente AD

Se esiste anche un singolo utente del database per un utente AD, gli oggetti verranno sempre creati nello schema associato al singolo utente del database. Se lo schema non esiste per l'utente del database, verrà utilizzato lo schema dbo. Crea un login di Windows individuale e un utente del database per l'utente AD user1. Connect tramite TDS endpoint utilizzando un accesso sysadmin

```
1> CREATE LOGIN [corp\user1] FROM WINDOWS;
2> GO
1> CREATE USER [corp\user1] FOR LOGIN [corp\user1] WITH DEFAULT_SCHEMA = sch1;
2> GO
1> CREATE SCHEMA sch1 AUTHORIZATION [corp\user1];
2> GO
1> SELECT name, default_schema_name FROM sys.database_principals WHERE name = 'corp
\user1';
2> GO

name      default_schema_name
--------- -------------------
corp\user1 sch1

(1 rows affected)
```

Connettiti utilizzando l'utente AD user1 e prova a creare un oggetto senza specificare lo schema in modo esplicito. La tabella t2 verrà creata nello schema sch1. Si noti inoltre che il proprietario di questo oggetto sarà lo stesso del proprietario dello schema sch1.

```
1> CREATE TABLE t2(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't2';
2> GO

name schema_name
---- -----------
t2   sch1

(1 rows affected)
```

Limitazioni

- L'utilità Dump/Restore non supporta il dump delle mappature dell'estensione pg_ad_mapping. Dovrai ricreare quelle mappature dopo il ripristino.
- La distribuzione Blue-Green non è supportata per le istanze Babelfish e Aurora Postgre con. SQL `pg_ad_mapping`
- La creazione implicita di schemi non è supportata. DDLle istruzioni che richiedono la creazione implicita di schemi non sono supportate.
- L'attivazione a livello di server DDLs ALTER AUTHORIZATIONDATABASE, CREATEDATABASE, CREATELOGIN, ALTER LOGIN ALTER SERVERROLE, ALTER DATABASE non è supportata in una sessione autenticata di Group AD quando non esiste un accesso individuale a Windows, ma solo un accesso a Windows di gruppo. Per ovviare a questa limitazione, si consiglia di eseguire queste operazioni in una sessione autenticata con password o di creare un accesso individuale a Windows.
- La creazione implicita di utenti non è supportata. SQLComportamento T ideale [non ancora supportato in Babelfish]; in alcuni casi istruzioni di controllo dell'accesso comeGRANT/ REVOKEdove il nome dell'utente AD è specificato nel comando ma non esiste nel database, l'utente del database denominato utente AD viene creato implicitamente. DDL
- Per le SQL procedure o le funzioni DDLs in PL/pg create dall'PSQLendpoint ed eseguite dall'endpoint nella sessione autenticata di Group AD: TDS
  - ALTER/DROPle dichiarazioni saranno supportate.
  - CREATETABLE, CREATEVIEW, CREATEINDEX, CREATEFUNCTION/PROC,, CREATETYPE, CREATESEQUENCE, CREATE TRIGGER SELECT INTO CREATE FULLTEXTINDEX,

CREATE UNIQUE INDEX genererà un errore se lo schema non viene fornito esplicitamente e lo schema predefinito è nullo per la sessione corrente.

- CREATEDATABASECREATEEXTENSIONe tutte le altre CREATE istruzioni per la CREATE sottoscrizione, il CREATE tablespace, la CREATE politica e la CREATE conversione di oggetti specifici di PG (non in T-SQL) non saranno supportate.

- DDLsdall'SQLendpoint Postgre non è supportato nella sessione autenticata di Group AD. Come soluzione alternativa, puoi sempre connetterti utilizzando l'utente principale o qualsiasi altro utente utilizzando un meccanismo di autenticazione basato su password.
- Gli oggetti di sistema come SUSER _ SID (), IS_ (), IS_ SRVROLEMEMBER MEMBER (), sys.dm_exec_sessions presentano le seguenti limitazioni.
  - SUSER_ SID () non restituirà SID quando viene fornito l'utente AD o il gruppo di sicurezza AD.
  - IS_ SRVROLEMEMBER () non prenderà in considerazione l'appartenenza al ruolo se l'utente AD corrente eredita l'appartenenza al ruolo server dall'appartenenza al ruolo server di qualsiasi accesso al gruppo Windows.
  - IS_ MEMBER () restituirà false per qualsiasi query relativa al gruppo Windows.
  - sys.dm_exec_sessions non mostrerà i valori previsti nelle colonne login_name, nt_user_name.

Connessione a Babelfish tramite l'SQLendpoint Postgre sulla porta Postgre SQL

Puoi utilizzare gli accessi di gruppo creati dalla porta per connetterti anche tramite la TDS porta Postgre. SQL Per connetterti tramite la SQL porta Postgre, devi specificare il nome utente AD nel formato `<ad_username@FQDN>` delle applicazioni client Postgre. SQL Non puoi usare il formato. `<DNS domain name\ad_username>`

Per impostazione predefinita, Postgre SQL utilizza i confronti con distinzione tra maiuscole e minuscole per i nomi utente. Affinché Aurora Postgre SQL interpreti i nomi utente Kerberos senza distinzione tra maiuscole e minuscole, è necessario impostare il parametro krb_caseins_users su true nel gruppo di parametri del cluster Babelfish personalizzato. Questo parametro è impostato su false per impostazione predefinita. Per ulteriori informazioni, consulta Configurazione del cluster database Aurora PostgreSQL per nomi utente senza distinzione tra maiuscole e minuscole.

Differenze di comportamento tra gli SQL endpoint T SQL e Postgre quando un utente AD fa parte di più gruppi

Considerate che l'utente AD user1 fa parte di due gruppi di sicurezza AD [corp\ accounts-group] e [corp\ sales-group] e che DB admin ha impostato la mappatura degli utenti nel modo seguente.

```
postgres=> select * from pgadmap_read_mapping();

ad_sid       | pg_role                           | weight | ad_grp
-------------+-----------------------------------+--------+---------------
S-1-5-67-980 | accounts-group@CORP.EXAMPLE.COM | 7      | accounts-group
S-1-2-34-560 | sales-group@CORP.EXAMPLE.COM    | 10     | sales-group
(2 rows)
```

Se l'utente si connette dall'SQLendpoint T, durante l'autorizzazione erediterà i privilegi da tutti gli accessi T associati. SQL In questo esempio user1 erediterà l'unione dei privilegi sia dall'accesso al SQL gruppo T- e i pesi verranno ignorati. Questo è in linea con il comportamento T- standard. SQL

Tuttavia, se lo stesso utente si connette dall'SQLendpoint Postgre, può ereditare i privilegi da un solo accesso T associato con il peso maggiore. SQL Se ai due login del SQL gruppo T è stato assegnato lo stesso peso, l'utente AD erediterà i privilegi del SQL T-login corrispondente alla mappatura aggiunta più di recente. Per PostgreSQL, la raccomandazione è di specificare pesi che riflettano le autorizzazioni/privilegi relativi dei singoli ruoli del DB per evitare l'ambiguità. Nell'esempio seguente, l'utente 1 si è connesso tramite endpoint e ha ereditato solo i privilegi dei gruppi di vendita. PSQL

```
babelfish_db=> select session_user, current_user;

   session_user               |   current_user
------------------------------+---------------------------
 sales-group@CORP.EXAMPLE.COM | sales-group@CORP.EXAMPLE.COM
(1 row)


babelfish_db=> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();

     principal          | gss_authenticated
------------------------+-------------------
 user1@CORP.EXAMPLE.COM | t
(1 row)
```

Connessione a un cluster database Babelfish

Per connetterti a Babelfish, esegui la connessione all'endpoint del cluster Aurora PostgreSQL che esegue Babelfish. Il client può utilizzare uno dei seguenti driver client conformi a TDS versione 7.1 fino a 7.4:

- Open Database Connectivity (ODBC)
- Driver database OLE/msoledBSQL
- Java Database Connectivity (JDBC) versione 8.2.2 (mssql-jdbc-8.2.2) e versioni successive
- Microsoft SqlClient Data Provider per SQL Server
- Provider di dati .NET per SQL Server
- SQL Server Native Client 11.0 (obsoleto)
- Provider OLE DB/SQLOLEDB (obsoleto)

Con Babelfish esegui le seguenti operazioni:

- Strumenti, applicazioni e sintassi di SQL Server sulla porta TDS, per impostazione predefinita della porta 1433.
- Strumenti, applicazioni e sintassi PostgreSQL sulla porta PostgreSQL, per impostazione predefinita 5432.

Per ulteriori informazioni sulla connessione ad Aurora PostgreSQL, consulta Connessione a un cluster di database Amazon Aurora PostgreSQL.

> **Note**
>
> Gli strumenti di sviluppo di terze parti che utilizzano il provider OLEDB di SQL Server per accedere ai metadati non sono supportati. Ti consigliamo di utilizzare connessioni client SQL Server JDBC, ODBC o SQL Native per questi strumenti.

Argomenti

- Ricerca dell'endpoint del writer e del numero di porta
- Creazione di connessioni client C # o JDBC Babelfish
- Utilizzo di un client SQL Server per la connessione al cluster database

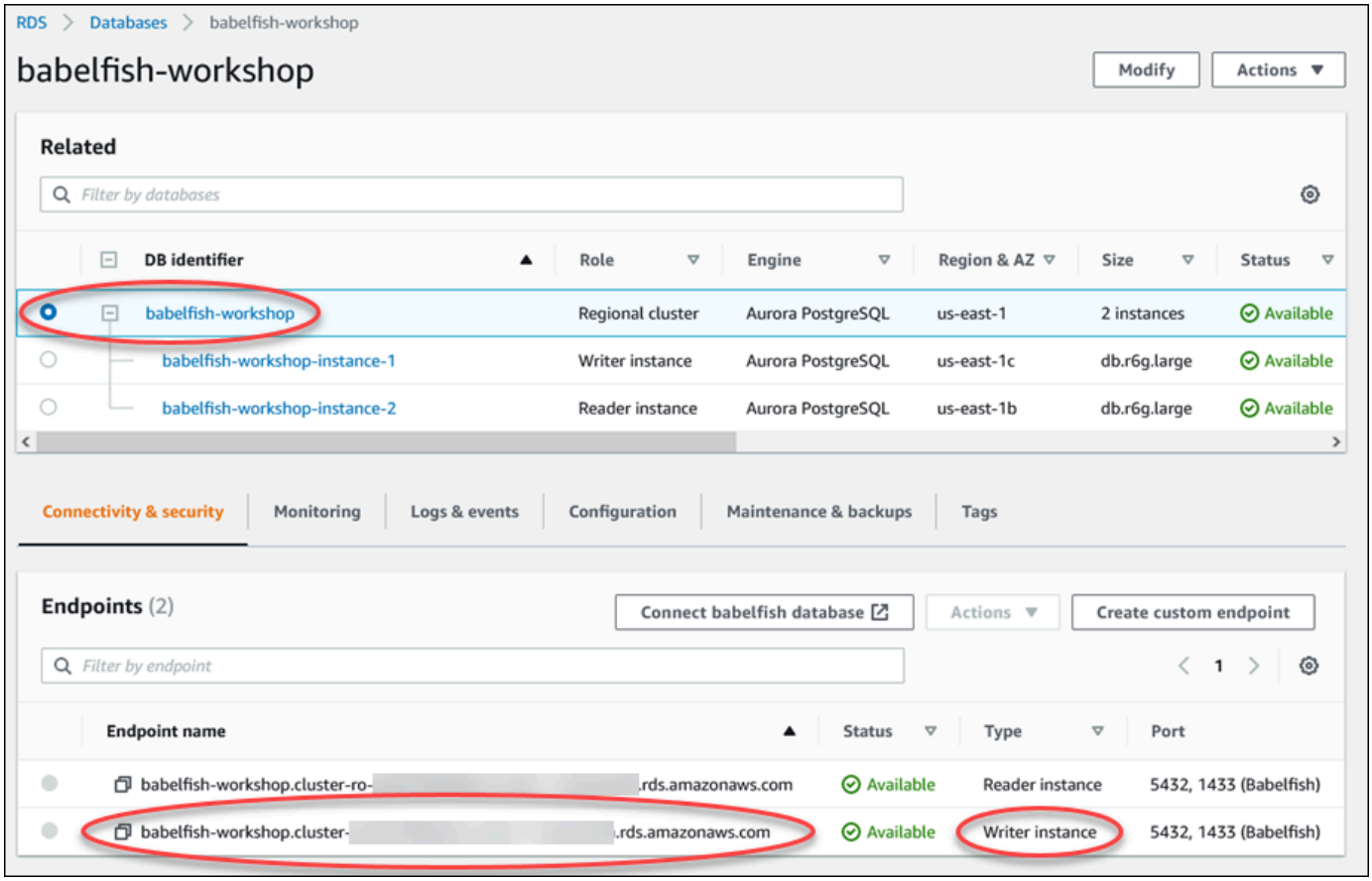
- Utilizzo di un client PostgreSQL per connetterti al cluster di database

Ricerca dell'endpoint del writer e del numero di porta

Per connetterti al cluster database Babelfish, utilizza l'endpoint associato all'istanza di scrittura (primaria) del cluster database. Lo stato dell'istanza deve essere Available (Disponibile). Dopo la creazione del cluster database Babelfish per Aurora PostgreSQL, possono essere necessari fino a 20 minuti affinché le istanze siano disponibili.

Per trovare l'endpoint del database

1. Apri la console per Babelfish.
2. Nel pannello di navigazione seleziona Database.
3. Scegli il cluster database Babelfish per Aurora PostgreSQL tra quelli elencati per visualizzarne i dettagli.
4. Sulla scheda Connettività e sicurezza, prendi nota dei valori Endpoint del cluster disponibile. Utilizza l'endpoint del cluster per l'istanza del writer nelle tue stringhe di connessione per tutte le applicazioni che eseguono operazioni di scrittura o lettura del database.



Per ulteriori informazioni sui dettagli del cluster database Aurora, consulta Creazione di un cluster database Amazon Aurora.

Creazione di connessioni client C # o JDBC Babelfish

Di seguito sono disponibili alcuni esempi di utilizzo delle classi C # e JDBC per connettersi a un Babelfish per Aurora PostgreSQL.

Example : Utilizzo del codice C # per connettersi a un cluster database

```
string dataSource = 'babelfishServer_11_24';

//Create connection
connectionString = @"Data Source=" + dataSource
    +";Initial Catalog=your-DB-name"
    +";User ID=user-id;Password=password";

SqlConnection cnn = new SqlConnection(connectionString);
cnn.Open();
```

Example : Utilizzo di classi e interfacce API JDBC generiche per connettersi a un cluster database

```
String dbServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";
String connectionUrl = "jdbc:sqlserver://" + dbServer + ":1433;" +
    "databaseName=your-DB-name;user=user-id;password=password";

// Load the SQL Server JDBC driver and establish the connection.
System.out.print("Connecting Babelfish Server ... ");
Connection cnn = DriverManager.getConnection(connectionUrl);
```

Example : Utilizzo di classi e interfacce JDBC specifiche di SQL Server per connettersi a un cluster database

```
// Create datasource.
SQLServerDataSource ds = new SQLServerDataSource();
ds.setUser("user-id");
ds.setPassword("password");
String babelfishServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";

ds.setServerName(babelfishServer);
ds.setPortNumber(1433);
ds.setDatabaseName("your-DB-name");
```

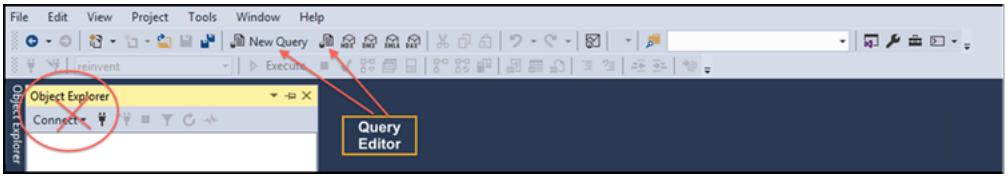
```
Connection con = ds.getConnection();
```

Utilizzo di un client SQL Server per la connessione al cluster database

È possibile utilizzare un client SQL Server per connettersi a Babelfish sulla porta TDS. A partire da Babelfish 2.1.0 e versioni successive, puoi utilizzare SSMS Object Explorer o l'editor di query SSMS per connetterti al cluster Babelfish.

Limitazioni

- In Babelfish 2.1.0 e versioni precedenti, l'utilizzo di `PARSE` per controllare la sintassi SQL non funziona come previsto. Invece di controllare la sintassi senza eseguire la query, il comando `PARSE` esegue la query ma non visualizza alcun risultato. L'utilizzo della combinazione di tasti <Ctrl><F5> SMSS per verificare la sintassi ha lo stesso comportamento anomalo, ovvero, Babelfish esegue inaspettatamente la query senza fornire alcun output.
- Babelfish non supporta MARS (Multiple Active Result Set). Assicurati che tutte le applicazioni client utilizzate per connetterti a Babelfish non siano impostate per utilizzare MARS.
- Per Babelfish 1.3.0 e versioni precedenti, solo l'editor di query è supportato per SSMS. Per utilizzare SSMS con Babelfish, assicurati di aprire la finestra di dialogo di connessione dell'editor di query in SSMS e non Object Explorer. Se si apre la finestra di dialogo Object Explorer, annulla la finestra di dialogo e apri nuovamente l'editor di query. Nell'immagine seguente, sono disponibili le opzioni di menu per scegliere quando connetterti a Babelfish 1.3.0 o versioni precedenti.



Per ulteriori informazioni sull'interoperabilità e le differenze di comportamento tra SQL Server e Babelfish, consulta Differenze tra Babelfish per Aurora PostgreSQL e SQL Server.

Utilizzo di sqlcmd per connettersi al cluster di database

È possibile connettersi e interagire con un cluster Aurora PostgreSQL DB che supporta Babelfish utilizzando solo la versione 19.1 e il client da riga di comando di SQL Server precedente. `sqlcmd` La versione 19.2 di SSMS non è supportata per la connessione a un cluster Babelfish. Utilizzare il seguente comando.

```
sqlcmd -S endpoint,port -U login-id -P password -d your-DB-name
```

Le opzioni sono le seguenti:

- `-S` è l'endpoint e la porta TDS (opzionale) del cluster database.
- `-U` è il nome di accesso dell'utente.
- `-P` è la password associata all'utente
- `-d` è il nome del database Babelfish.

Dopo la connessione, è possibile utilizzare molti degli stessi comandi utilizzati con SQL Server. Per alcuni esempi, consulta Recupero di informazioni dal catalogo di sistema Babelfish.

Utilizzo di SSMS per connettersi al cluster database

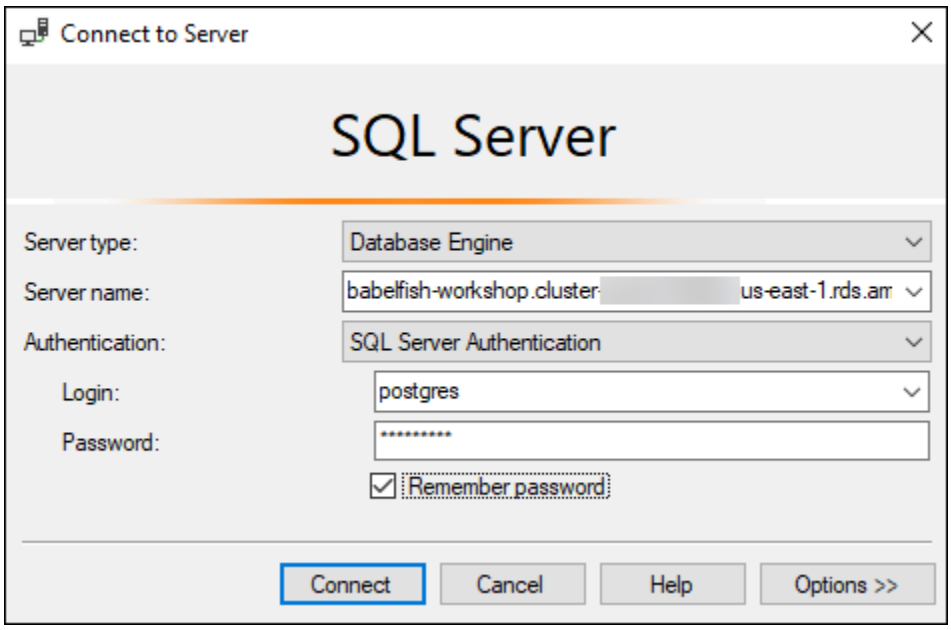
Puoi connetterti a un cluster database Aurora PostgreSQL che esegue Babelfish utilizzando Microsoft SQL Server Management Studio (SSMS). SSMS include una varietà di strumenti, incluso Importazione/Esportazione guidata di SQL Server discussa in Migrazione di un database SQL Server a Babelfish per Aurora Postgre SQL. Per ulteriori informazioni su SSMS, consulta Download SQL Server Management Studio (SSMS) nella documentazione di Microsoft.

Connessione al tuo database Babelfish con SSMS

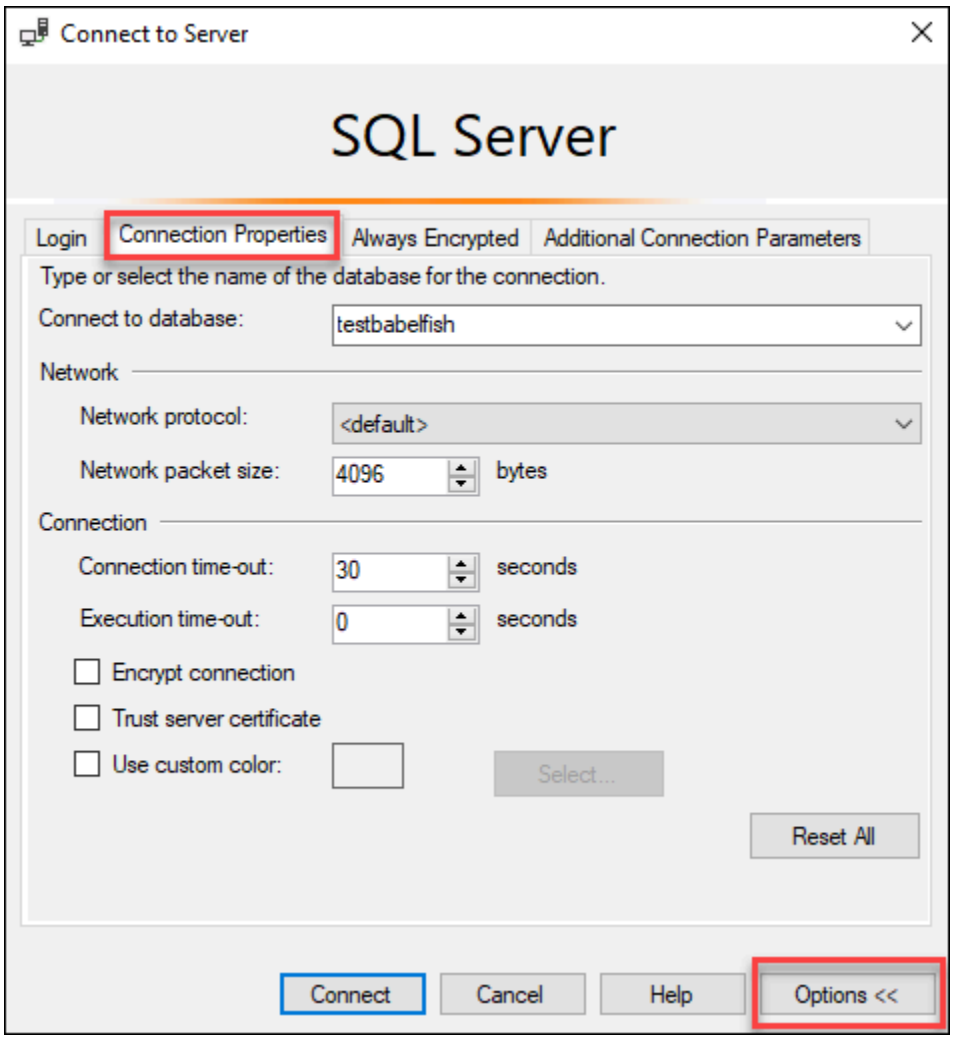
1. Avvia SSMS.
2. Apri la finestra di dialogo Connect to Server (Connettiti al server). Per continuare con la connessione, esegui una delle seguenti operazioni:
   - Scegliere New Query (Nuova query).
   - Se l'editor di query è aperto, scegliere Query,Connection (Connessione),Connect (Connetti).
3. Fornisci le seguenti informazioni per il tuo database:
   a. In Server type (Tipo di server) scegliere Database Engine (Motore di database).
   b. Per Server name (Server name), immettere il nome DNS. Ad esempio, il nome server dovrebbe essere simile a quanto segue.

      ```
      cluster-name.cluster-555555555555.aws-region.rds.amazonaws.com,1433
      ```

   c. In Authentication (Autenticazione) selezionare SQL Server Authentication (Autenticazione SQL Server).
   d. Per Login, immettere il nome utente scelto al momento della creazione del database.
   e. Per Password, immettere la password scelta al momento della creazione del database.



4. (Facoltativo) Scegli Opzioni e quindi scegli la scheda Connection Properties (Proprietà della connessione).



5. (Opzionale) Per Connect to database (Connetti al database), specificare il nome del database SQL Server migrato a cui connettersi e scegliere Connect (Collegarsi).

   Se viene visualizzato un messaggio che indica che SSMS non può applicare stringhe di connessione, scegliere OK.

   Se hai problemi a connetterti a Babelfish, consulta Errore della connessione.

   Per ulteriori informazioni sui problemi di connessione, consulta Risoluzione dei problemi di connessione all'istanza database di SQL Server nella Guida per l'utente di Amazon RDS.

Utilizzo di un client PostgreSQL per connetterti al cluster di database

È possibile utilizzare un client PostgreSQL per connettersi a Babelfish sulla porta PostgreSQL.

Usare psql per connettersi al cluster di database

Puoi scaricare il client PostgreSQL dal sito Web di PostgreSQL. Per installare psql, segui le istruzioni relative al tuo sistema operativo.

Puoi eseguire query su un cluster di database Aurora PostgreSQL che supporta Babelfish con il client a riga di comando `psql`. Durante la connessione, utilizza la porta PostgreSQL (per impostazione predefinita, porta 5432). In genere, non è necessario specificare il numero di porta, a meno che non sia stato modificato rispetto a quello predefinito. Utilizza il seguente comando per connetterti a Babelfish dal client `psql`:

```
psql -h bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com
-p 5432 -U postgres -d babelfish_db
```

I parametri sono i seguenti:

- `-h` – Nome host dell'istanza del cluster di database (endpoint del cluster) cui desideri accedere.
- `-p` – Numero di porta PostgreSQL usato per connettersi alla istanza database.
- `-d`: database a cui si desidera connettersi. Il valore predefinito è `babelfish_db`.
- `-U` – L'account utente del database cui desideri accedere. (L'esempio mostra il nome utente master predefinito.)

Quando si esegue un comando SQL sul client psql, si termina il comando con un punto e virgola. Ad esempio, il comando SQL seguente esegue la query della vista sistemi pg_tables per restituire informazioni su ogni tabella del database.

```
SELECT * FROM pg_tables;
```

Il client psql ha anche un set di metacomandi integrati. Un metacomando è un collegamento che regola la formattazione o fornisce una scorciatoia che restituisce i metadati in un formato facile da usare. Ad esempio, il seguente metacomando restituisce informazioni simili al precedente comando SQL:

```
\d
```

I metacomandi non devono essere terminati con un punto e virgola (;).

Per uscire dal client psql, immettere \q.

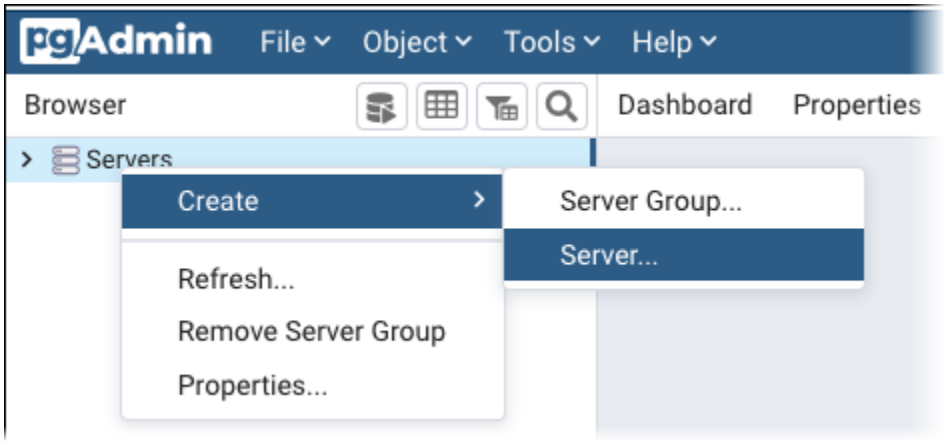
Per ulteriori informazioni sull'utilizzo del client psql per eseguire query su un cluster Aurora PostgreSQL, consulta la documentazione di PostgreSQL.

Uso di pgAdmin per connettersi al cluster di database

È possibile utilizzare il client pgAdmin per accedere ai dati nel dialetto PostgreSQL nativo.

Per connettersi al cluster con il client pgAdmin

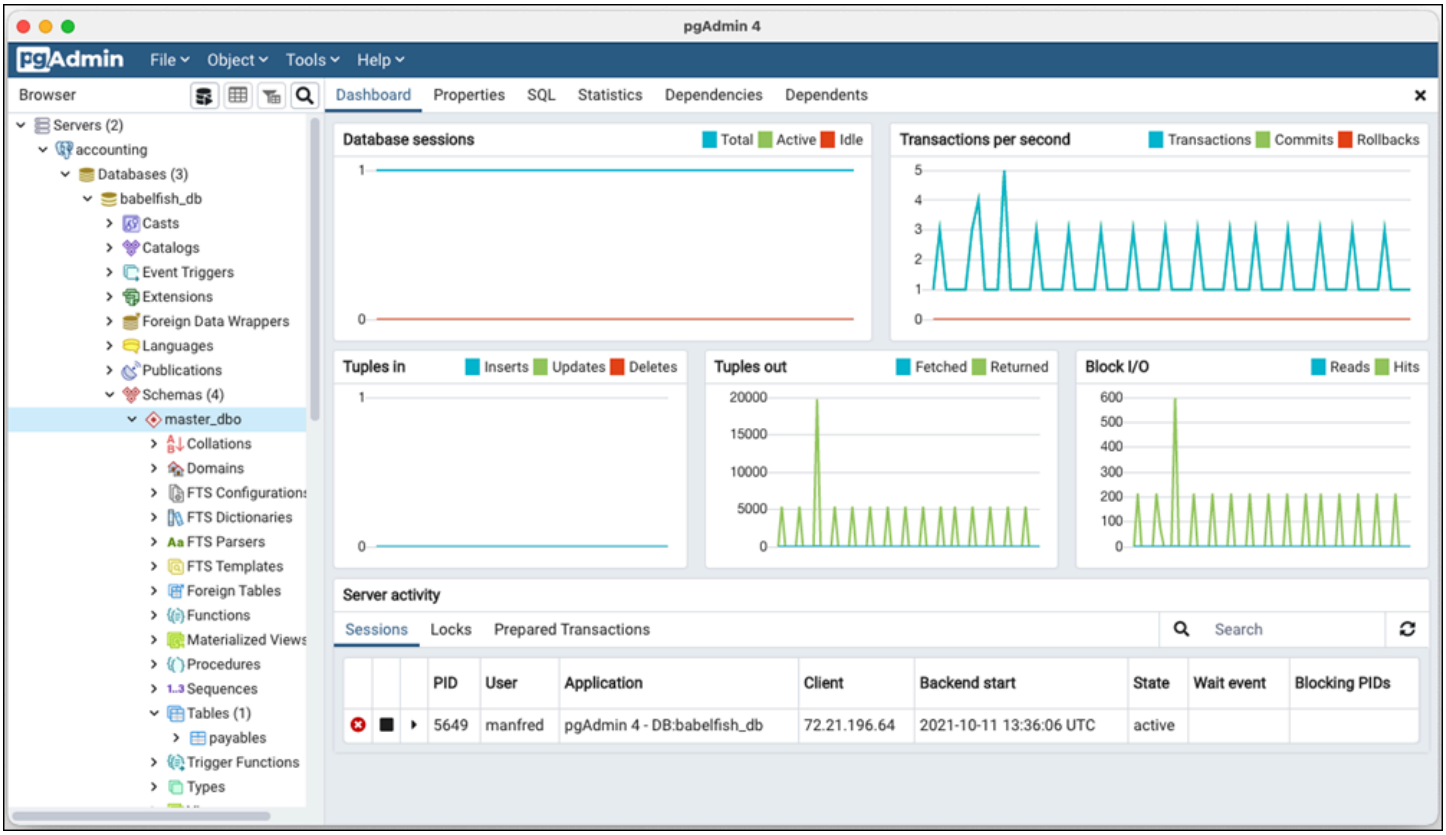
1. Scarica e installa il client pgAdmin dal sito web pgAdmin.
2. Apri il client e esegui l'autenticazione con pgAdmin.
3. Aprire il menu contestuale (pulsante destro del mouse) per Server e quindi scegliere Create (Crea), Server.



4. Inserimento delle informazioni nella finestra di dialogo Create Server (Crea Server).

   Sulla scheda Connection (Connessione), aggiungi l'indirizzo del cluster Aurora PostgreSQL per Host e il numero di porta PostgreSQL (per impostazione predefinita, 5432) per Port (Porta). Fornisci i dettagli di autenticazione e scegli Save (Salva).

Dopo la connessione, è possibile utilizzare la funzionalità PGAmin per monitorare e gestire il cluster Aurora PostgreSQL sulla porta PostgreSQL.



Per ulteriori informazioni, consulta la pagina Web pgAdmin.

Utilizzo di Babelfish

Di seguito sono disponibili informazioni di utilizzo per Babelfish, incluse alcune delle differenze tra l'utilizzo di Babelfish e SQL Server e tra i database Babelfish e PostgreSQL.

Argomenti

- Recupero di informazioni dal catalogo di sistema Babelfish
- Gestione delle autorizzazioni e del controllo degli accessi in Babelfish for Aurora PostgreSQL
- Differenze tra Babelfish per Aurora PostgreSQL e SQL Server
- Utilizzo di funzionalità Babelfish con implementazione limitata
- Miglioramento delle prestazioni della query Babelfish
- Utilizzo delle estensioni Aurora PostgreSQL con Babelfish
- Babelfish supporta i server collegati
- Utilizzo della ricerca nel testo completo in Babelfish
- Babelfish supporta i tipi di dati geospaziali
- Comprendere il partizionamento in Babelfish

Recupero di informazioni dal catalogo di sistema Babelfish

Puoi ottenere informazioni sugli oggetti di database archiviati nel cluster Babelfish eseguendo query su molte delle stesse viste di sistema utilizzate in SQL Server. Ogni nuova versione di Babelfish aggiunge supporto per ulteriori viste di sistema. Per un elenco delle viste attualmente disponibili, consulta la tabella SQL Server system catalog views.

Queste viste di sistema forniscono informazioni dal catalogo di sistema (`sys.schemas`). Nel caso di Babelfish, queste viste contengono entrambi gli schemi di sistema SQL Server e PostgreSQL. Per eseguire query su Babelfish per informazioni sul catalogo di sistema, puoi utilizzare la porta TDS o la porta PostgreSQL, come illustrato negli esempi seguenti.

- Esegui query sulla porta T-SQL utilizzando **`sqlcmd`** o un altro client SQL Server.

```
1> SELECT * FROM sys.schemas
2> GO
```

  Questa query restituisce gli schemi di sistema SQL Server e Aurora PostgreSQL, come mostrato di seguito.

```
name
------------------------------------------------------------
demographic_dbo
public
sys
master_dbo
tempdb_dbo
...
```

- Esegui query sulla porta PostgreSQL utilizzando **`psql`** o **`pgAdmin`**. In questo esempio viene utilizzato il meta-comando degli schemi di elenco `psql` (`\dn`):

  ```
  babelfish_db=> \dn
  ```

  La query restituisce lo stesso set di risultati restituito da `sqlcmd` sulla porta T-SQL.

  ```
         List of schemas
              Name
  ------------------------------

   demographic_dbo

   public
   sys
   master_dbo
   tempdb_dbo
  ...
  ```

Cataloghi di sistema SQL Server disponibili in Babelfish

Nella tabella seguente sono disponibili le viste SQL Server attualmente implementate in Babelfish. Per ulteriori informazioni sui cataloghi di sistema in SQL Server, consulta System Catalog Views (Transact-SQL) nella documentazione di Microsoft.

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.all_columns`	Tutte le colonne di tutte le tabelle e tutte le viste

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.all_objects`	Tutti gli oggetti in tutti gli schemi
`sys.all_sql_modules`	L'unione di `sys.sql_modules` e `sys.system_sql_modules`
`sys.all_views`	Tutte le visualizzazioni in tutti gli schemi
`sys.columns`	Tutte le colonne in tabelle e viste definite dall'utente
`sys.configurations`	Supporto Babelfish limitato a una singola configurazione di sola lettura.
`sys.data_spaces`	Contiene una riga per ogni spazio dati. Può essere un filegroup, uno schema di partizione o un filegroup di dati FILESTREAM.
`sys.database_files`	Una vista per database contenente una riga per ogni file di un database come archiviato nel database stesso.
`sys.database_mirroring`	Per informazioni, consulta sys.database_mirroring nella documentazione di Microsoft Transact-SQL.
`sys.database_principals`	Per informazioni, consulta sys.database_principals nella documentazione di Microsoft Transact-SQL.
`sys.database_role_members`	Per informazioni, consulta sys.database_role_members nella documentazione di Microsoft Transact-SQL.
`sys.databases`	Tutti i database in tutti gli schemi

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.dm_exec_connections`	Per informazioni, consulta sys.dm_exec_connections nella documentazione di Microsoft Transact-SQL.
`sys.dm_exec_sessions`	Per informazioni, consulta sys.dm_exec_sessions nella documentazione di Microsoft Transact-SQL.
`sys.dm_hadr_database_replica_states`	Per informazioni, consulta sys.dm_hadr_database_replica_states nella documentazione di Microsoft Transact-SQL.
`sys.dm_os_host_info`	Per informazioni, consulta sys.dm_os_host_info nella documentazione di Microsoft Transact-SQL.
`sys.endpoints`	Per informazioni, consulta sys.endpoints nella documentazione di Microsoft Transact-SQL.
`sys.indexes`	Per informazioni, consulta sys.indexes nella documentazione di Microsoft Transact-SQL.
`sys.languages`	Per informazioni, consulta sys.languages nella documentazione di Microsoft Transact-SQL.
`sys.schemas`	Tutti gli schemi
`sys.server_principals`	Tutti gli accessi e i ruoli
`sys.sql_modules`	Per informazioni, consulta sys.sql_modules nella documentazione di Microsoft Transact-SQL.
`sys.sysconfigures`	Supporto Babelfish limitato a una singola configurazione di sola lettura.

Nome della vista	Descrizione o limitazione Babelfish (se presente)
`sys.syscurconfigs`	Supporto Babelfish limitato a una singola configurazione di sola lettura.
`sys.sysprocesses`	Per informazioni, consulta sys.sysprocesses nella documentazione di Microsoft Transact-SQL.
`sys.system_sql_modules`	Per informazioni, consulta sys.system_sql_modules nella documentazione di Microsoft Transact-SQL.
`sys.table_types`	Per informazioni, consulta sys.table_types nella documentazione di Microsoft Transact-SQL.
`sys.tables`	Tutte le tabelle in uno schema
`sys.xml_schema_collections`	Per informazioni, consulta sys.xml_schema_collections nella documentazione di Microsoft Transact-SQL.

PostgreSQL implementa cataloghi di sistema simili alle viste del catalogo oggetti di SQL Server. Per l'elenco completo dei cataloghi di sistema, consulta System Catalogs (Cataloghi di sistema) nella documentazione di PostgreSQL.

Esportazioni DDL supportate da Babelfish

Dalle versioni Babelfish 2.4.0 e 3.1.0, Babelfish supporta le esportazioni DDL utilizzando vari strumenti. Ad esempio, puoi utilizzare questa funzionalità di SQL Server Management Studio (SSMS) per generare gli script di definizione dei dati per vari oggetti in un database Babelfish per Aurora PostgreSQL. Quindi puoi utilizzare i comandi DDL generati in questo script per creare gli stessi oggetti in un altro database Babelfish per Aurora PostgreSQL o SQL Server.

Babelfish supporta le esportazioni DDL per i seguenti oggetti nelle versioni specificate.

Elenco di oggetti	2.4.0	3.1.0
Tabelle dell'utente	Sì	Sì
Chiavi primarie	Sì	Sì
Chiavi esterne	Sì	Sì
Vincoli univoci	Sì	Sì
Indici	Sì	Sì
Vincoli check	Sì	Sì
Visualizzazioni	Sì	Sì
Stored procedure	Sì	Sì
Funzioni definite dall'utente	Sì	Sì
Funzioni con valori tabellari	Sì	Sì
Trigger	Sì	Sì
Tipi di dati definiti dall'utente	No	No
Tipi di tabella definiti dall'utente	No	No
Utenti	No	No
Accessi	No	No
Sequenze	No	No
Roles	No	No

Limitazioni relative alle esportazioni DDLs

- Utilizzate le vie di fuga prima di ricreare gli oggetti con quelli esportati DDLs: Babelfish non supporta tutti i comandi nello script DDL esportato. Usa gli escape hatch per evitare errori causati

durante la ricreazione degli oggetti dai comandi DDL in Babelfish. Per ulteriori informazioni sugli escape hatch, consulta Gestione degli errori di Babelfish con escape hatch.

- Oggetti contenenti vincoli CHECK con clausole COLLATE esplicite: gli script con questi oggetti generati da un database SQL Server hanno regole di confronto diverse ma equivalenti a quelle del database Babelfish. Ad esempio, alcune raccolte, come sql_latin1_general_cp1_cs_as, sql_latin1_general_cp1251_cs_as e latin1_general_cs_as vengono generate come latin1_general_cs_as, che è la regola di confronto di Windows più vicina.

Gestione delle autorizzazioni e del controllo degli accessi in Babelfish for Aurora PostgreSQL

In Babelfish for Aurora PostgreSQL, puoi gestire le autorizzazioni e il controllo degli accessi per database, schemi e oggetti. Le tabelle seguenti illustreranno i comandi SQL specifici per la concessione delle autorizzazioni in Babelfish per realizzare vari scenari di controllo degli accessi. Coprirà i casi d'uso supportati che possono essere implementati, nonché le soluzioni alternative per i casi attualmente non supportati. Ciò ti consentirà di configurare le autorizzazioni appropriate per soddisfare i tuoi requisiti di sicurezza e conformità quando lavori con i database Babelfish.

Casi di utilizzo supportati

La tabella seguente spiega i casi d'uso supportati in Babelfish. Per ogni caso d'uso, la tabella mostra l'azione necessaria per ottenerlo e alcuni esempi di comandi SQL.

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish
Consenti al login di eseguire SELECTs/DMLs/DDLs in qualsiasi database	Aggiungi il login al ruolo del server sysadmin	ALTER SERVER ROLE sysadmin AGGIUNGI MEMBRO `login`	Nessuno	Tutte le versioni
Consenti al login di eseguire SELECTs/D	Rendi login il proprietario del database	MODIFICA L'AUTORIZZAZIONE SUL	Un database può avere un solo proprietario.	Versione 3.4 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish
MLs/DDLs in un database		DATABASE:: `database` TO `login`		
Consenti all'utente del database di eseguire operazioni SELECTs/DMLs su uno schema	Concedi l'autorizzazione all'utente del database sullo schema	CONCESSIONE SELECT/EXECUTE/INSERT/UPDATE/DELETE SU SCHEMA:: `schema` A `user`	CASCADE non è supportato con Grant/Revoke on Schema. GRANT/REVOKE OPTION FOR.. on SCHEMA non è supportato in Babelfish. GRANT/REVOKE all'interno di CREATE SCHEMA non è supportato in Babelfish.	Versione 3.6 e successive, 4.2 e successive
Consenti all'utente del database di eseguire operazioni SELECTs/DMLs su uno schema	Rendi l'utente del database proprietario dello schema al momento della creazione dello schema	CREA L'`schema`AUTORIZZIONE DELLO SCHEMA`user`.	La modifica della proprietà dello schema dopo la creazione non è attualmente supportata.	Versione 1.2 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish
Consenti all'utente del database di eseguire operazioni SELECTs/DMLs su un oggetto	Concedi l'autorizzazione all'utente del database sull'oggetto	SELECT/EXECUTE/INSERT/UPDATE/DELETECONCEDI SULL'OGGETTO:: `object` A `user`	Nessuno	Tutte le versioni
Consenti all'utente del database di eseguire SELECTs/DMLs/DDLs in un database, inclusa l'eliminazione del database	Aggiungi utente al ruolo fisso del database db_owner	ALTER ROLE db_owner AGGIUNGI MEMBRO `user`	Solo agli utenti può essere aggiunto il ruolo fisso del database db_owner. L'aggiunta di ruoli al ruolo db_owner non è ancora supportata.	Versione 4.5 e successive, 5.1 e successive
Consenti all'utente o ai membri di un ruolo di database personalizzato di eseguire solo operazioni SELECTs in un database	Aggiungi utente o ruolo al ruolo fisso del database db_datareader	ALTERA RUOLO db_datareader AGGIUNGI MEMBRO/`userro`	Nessuno	Versione 4.5 e successive, 5.1 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish
Consenti all'utente o ai membri di un ruolo di database personalizzato di eseguire solo operazioni DMLs in un database	Aggiungi utente o ruolo al ruolo fisso del database db_datawriter	MODIFICA RUOLO db_datawriter AGGIUNGI MEMBRO/`userro`	Nessuno	Versione 4.5 e successive, 5.1 e successive
Consenti all'utente o ai membri di un ruolo di database personalizzato di eseguire solo operazioni DDLs in un database	Aggiungi utente o ruolo al ruolo fisso del database db_accessadmin	ALTER ROLE db_accessadmin AGGIUNGI MEMBRO/`userro`	Nessuno	Versione 4.5 e successive, 5.1 e successive
Consenti all'utente o ai membri di un ruolo di database personalizzato di CREATE/ALTER/DROP custom roles, GRANT/REVOKE permissions on objects in a database and/or CREARE SCHEMA solo in un database	Aggiungi utente o ruolo al ruolo fisso del database db_securityadmin	ALTER ROLE db_securityadmin AGGIUNGI MEMBRO/`userro`	Nessuno	Versione 4.5 e successive, 5.1 e successive

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish
Consenti all'utente o ai membri di un ruolo di database personalizzato di CREATE/ALTER/DROP any user, grant and revoke database access and map user accounts to logins and/or CREARE SCHEMA solo in un database	Aggiungi utente o ruolo al ruolo fisso del database db_accessadmin	ALTER ROLE db_accessadmin AGGIUNGI MEMBRO/`userro`	Nessuno	Versione 4.5 e successive, 5.1 e successive
Consenti l'accesso solo a CREATE/DROP/ALTER qualsiasi database	Aggiungi l'accesso al ruolo fisso del server dbcreator	ALTER SERVER ROLE dbcreator AGGIUNGI MEMBRO `login`	Possono essere modificati solo i database a cui dbcreator login ha accesso.	Versione 4.5 e successive, 5.1 e successive
Consenti l'accesso solo a CREATE/ALTER/DROP qualsiasi accesso	Aggiungi l'accesso al ruolo fisso del server securityadmin	ALTER SERVER ROLE securityadmin AGGIUNGI MEMBRO `login`	Nessuno	Versione 4.5 e successive, 5.1 e successive

Casi d'uso non supportati con le soluzioni alternative

La tabella seguente spiega i casi d'uso che non sono supportati in Babelfish, ma che possono essere raggiunti utilizzando una soluzione alternativa.

Caso d'uso	Azione	Comandi SQL	Commenti	Compatibilità della versione Babelfish per soluzioni alternative
Consenti all'utente di utilizzare SELECTs/DMLs un' object/schema opzione per CONCEDERE queste autorizzazioni ad altri utenti	CONCEDI le autorizzazioni direttamente a tutti gli altri utenti	CONCEDISELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT/SCHEMA::`object/schema` TO `user`	CONCEDERE ... WITH GRANT OPTION non è attualmente supportato.	Versione 3.6 e successive, 4.2 e successive
Consenti all'utente del database di eseguire SELECTs/DMLs/DDLs in un database, inclusa l'eliminazione del database	Aggiungi membri del ruolo al ruolo fisso del database db_owner	ALTER ROLE db_owner AGGIUNGI MEMBRO `user`	L'aggiunta di ruoli al ruolo db_owner non è attualmente supportata.	Versione 4.5 e successive, 5.1 e successive

Casi di utilizzo non supportati

La tabella seguente spiega i casi d'uso che non sono supportati in Babelfish.

Caso d'uso	Commenti
Impedisci a un utente o ai membri di un database personalizzato di svolgere SELECTs un ruolo all'interno di un database	Il ruolo fisso del database db_denydatareader non è ancora supportato

Caso d'uso	Commenti
Impedisci a un utente o ai membri di un ruolo di database personalizzato di svolgere in un database DMLs	Il ruolo fisso del database db_denydatawriter non è attualmente supportato.
Consenti l'accesso solo per INTERROMPERE qualsiasi connessione al database	Il ruolo fisso del server processadmin non è attualmente supportato.
Consenti l'accesso solo per aggiungere o rimuovere server collegati	Il ruolo fisso del server setupadmin non è attualmente supportato.

Differenze tra Babelfish per Aurora PostgreSQL e SQL Server

Babelfish è una funzionalità di Aurora PostgreSQL in evoluzione, con nuove funzionalità aggiunte in ogni versione a partire dall'offerta iniziale in Aurora PostgreSQL 13.4. È progettata per fornire semantica T-SQL nella parte superiore di PostgreSQL, attraverso il dialetto T-SQL utilizzando la porta TDS. Ogni nuova versione di Babelfish aggiunge funzionalità e funzioni che si allineano meglio alle funzionalità e al comportamento di T-SQL, come mostrato nella tabella Funzionalità supportate in Babelfish per versione. Per ottenere risultati ottimali durante l'utilizzo di Babelfish, si consiglia di comprendere le differenze attualmente esistenti tra il T-SQL supportato da SQL Server e Babelfish per la versione più recente. Per ulteriori informazioni, consulta Differenze T-SQL in Babelfish.

Oltre alle differenze tra T-SQL supportato da Babelfish e SQL Server, potrebbe anche essere necessario considerare i problemi di interoperabilità tra Babelfish e PostgreSQL nel contesto del cluster database Aurora PostgreSQL. Come citato in precedenza, Babelfish supporta la semantica T-SQL nella parte superiore di PostgreSQL, attraverso il dialetto T-SQL utilizzando la porta TDS. Allo stesso tempo, è anche possibile accedere al database Babelfish tramite la porta PostgreSQL standard con istruzioni SQL PostgreSQL. Se si intende utilizzare entrambe le funzionalità di PostgreSQL e Babelfish in un'implementazione della produzione, occorre tenere presente i potenziali problemi di interoperabilità tra i nomi degli schemi, gli identificatori, le autorizzazioni, la semantica transazionale, i set di risultati multipli, le regole di fascicolazione predefinite e così via. In poche parole, quando le istruzioni PostgreSQL o l'accesso PostgreSQL si verificano nel contesto di Babelfish, si può verificare un'interferenza tra PostgreSQL e Babelfish che può potenzialmente influire su sintassi, semantica e compatibilità quando vengono rilasciate nuove versioni di Babelfish. Per informazioni complete e le linee guida su tutte le considerazioni, consulta Guidance on Babelfish Interoperability nella documentazione di Babelfish per PostgreSQL.

> **Note**
>
> Prima di utilizzare la funzionalità nativa di PostgreSQL e la funzionalità di Babelfish nello stesso contesto di applicazione, è opportuno considerare i problemi discussi in Guidance on Babelfish Interoperability nella documentazione di Babelfish per PostgreSQL. Questi problemi di interoperabilità (Aurora PostgreSQL e Babelfish) sono pertinenti solo se si prevede di utilizzare l'istanza database PostgreSQL nello stesso contesto di applicazione di Babelfish.

Argomenti

- Discarica e ripristina Babelfish
- Differenze T-SQL in Babelfish
- Livelli di isolamento delle transazioni in Babelfish

Discarica e ripristina Babelfish

A partire dalle versioni 4.0.0 e 3.4.0, gli utenti Babelfish possono ora utilizzare le utilità di dump e restore per il backup e il ripristino dei propri database. Per ulteriori informazioni, consulta Babelfish dump and restore. Questa funzionalità si basa sulle utilità di dump e ripristino di PostgreSQL. Per ulteriori informazioni, vedi pg_dump e vedi pg_restore. Per utilizzare efficacemente questa funzionalità in Babelfish, è necessario utilizzare strumenti basati su PostgreSQL specificamente adattati per Babelfish. La funzionalità di backup e ripristino per Babelfish differisce notevolmente da quella di SQL Server. Per ulteriori informazioni su queste differenze, consulta Differenze di funzionalità di dump e ripristino: Babelfish e SQL Server. Babelfish for Aurora PostgreSQL offre funzionalità aggiuntive per il backup e il ripristino dei cluster Amazon Aurora PostgreSQL DB. Per ulteriori informazioni, consulta Backup e ripristino di un cluster DB Amazon Aurora.

Differenze T-SQL in Babelfish

Di seguito, è possibile trovare una tabella delle funzionalità T-SQL supportate nella versione corrente di Babelfish con alcune note sulle differenze nel comportamento rispetto a quello di SQL Server.

Per ulteriori informazioni sul supporto delle diverse versioni, consulta Funzionalità supportate in Babelfish per versione. Per informazioni sulle funzionalità attualmente non supportate, consulta Funzionalità non supportate in Babelfish.

Babelfish è disponibile con Aurora PostgreSQL-Compatible Edition. Per ulteriori informazioni sulle versioni di Babelfish, consulta Release notes for Amazon Aurora PostgreSQL-Compatible Edition.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
\ (carattere di continuazione della riga)	Il carattere di continuazione della riga (una barra rovesciata prima di una nuova riga) per stringhe di caratteri ed esadecimali non è attualmente supportato. Per le stringhe di caratteri, la barra rovesciata-nuova riga viene interpretata come caratteri nella stringa. Per le stringhe esadecimali, la barra rovesciata-nuova riga genera un errore di sintassi.
@version	Il formato del valore restituito da `@@version` è leggermente diverso dal valore restituito da SQL Server. Il codice potrebbe non funzionare correttamente se dipende dalla formattazione di `@@version` .
Funzioni di aggregazione	Le funzioni di aggregazione sono parzialmente supportate (sono supportate AVG, COUNT, COUNT_BIG, GROUPING, MAX, MIN, STRING_AGG e SUM). Per gli elenchi di funzionalità non supportate, vedere Funzioni non supportate.
ALTER TABLE	Supporta l'aggiunta o l'eliminazione di una sola colonna o di un solo vincolo.
ALTER TABLE..ALTER COLUMN	Al momento non è possibile specificare NULL e NOT NULL. Per modificare la nullabilità di una colonna, usa l'istruzione PostgreSQL ALTER TABLE..{SET|DROP} NOT NULL.

AT TIME ZONE	Durante il passaggio dall'ora legale (DST) all'ora solare, il periodo di sovrapposizione viene visualizzato utilizzando l'offset dell'ora solare. Per chiarire, consideriamo il seguente esempio:
	`SELECT CONVERT(DATETIME2(0), '2022-10-30T02:00:00', 126) AT TIME ZONE 'Central European Standard Time';`
	`GO;`
	`Result: 2022-10-30 02:00:00 +01:00`

Nomi di colonna vuoti senza alias di colonna	Le utility `sqlcmd` e `psql` gestiscono le colonne con nomi vuoti in modo diverso:

Funzionalità o sintassi	Descrizione del comportamento o della differenza
	• SQL Server `sqlcmd` restituisce un nome di colonna vuoto. • PostgreSQL `psql` restituisce un nome di colonna generato.
Funzione CHECKSUM	Babelfish e SQL Server utilizzano algoritmi di hashing diversi per la funzione CHECKSUM. Di conseguenza, i valori hash generati dalla funzione CHECKSUM in Babelfish potrebbero essere diversi da quelli generati dalla funzione CHECKSUM in SQL Server.
Impostazione predefinita colonna	Quando si crea una colonna predefinita, il nome del vincolo viene ignorato. Per eliminare una colonna predefinita, utilizza la sintassi seguente: `ALTER TABLE...ALTER COLUMN..DROP DEFAULT...`
Nome_vincolo	In SQL Server, i nomi dei vincoli devono essere univoci all'interno dello schema a cui appartiene la tabella. Tuttavia, in Babelfish, ciò si applica solo ai vincoli PRIMARY KEY e UNIQUE. Altri tipi di vincoli non sono soggetti a questa restrizione.
Vincoli	PostgreSQL non supporta l'attivazione e la disattivazione di singoli vincoli. L'istruzione DDL viene ignorata e viene emesso un avviso.
Vincoli con IGNORE_DUP_KEY	I vincoli vengono creati senza questa proprietà.
CREA, ALTERA, RILASCIA IL RUOLO DEL SERVER	RILASCIA IL RUOLO DEL SERVER è supportato solo per `sysadmin`. Tutta l'altra sintassi non è supportata. L'utente T-SQL in Babelfish ha un'esperienza simile a SQL Server per i concetti di accesso (principale del server), di un database e di un utente del database (principale del database).

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Le clausole CREATE, ALTER LOGIN sono supportate con una sintassi limitata	CREATE LOGIN... Sono supportate la clausola PASSWORD, DEFAULT_DATABASE e DEFAULT_LANGUAGE. L'ALTER LOGIN... La clausola PASSWORD è supportata, ma ALTER LOGIN... La clausola OLD_PASSWORD non è supportata. Solo un login che è un membro di sysadmin può modificare una password.
Raccolta distinzione tra lettere maiuscole e minuscole CREATE DATABASE	Le raccolte con distinzione tra maiuscole e minuscole non sono supportate con l'affermazione CREATE DATABASE
Parole chiave e clausole CREATE DATABASE	Le opzioni tranne COLLATE e CONTAINMENT=NONE non sono supportate. La clausola COLLATE è accettata ed è sempre impostata sul valore di `babelfishpg_tsql.server_collation_name` .
Clausole di supporto CREATE SCHEMA…	È possibile utilizzare il comando CREATE SCHEMA per creare uno schema vuoto. Utilizzare comandi aggiuntivi per creare oggetti dello schema.
I valori ID del database sono diversi su Babelfish	I database master e tempdb non saranno i database 1 e 2. IDs
La funzione FORMAT (tipo di data) è supportata con le seguenti limitazioni	Il meridiano a singolo carattere non è supportato. Il formato "yyy" in SQL Server restituisce 4 cifre per l'anno superiore a 1000, ma solo 3 cifre per gli altri. I formati "g" e "R" non sono supportati La traduzione locale "vi-VN" è leggermente diversa.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Identificatori superiori a 63 caratteri	PostgreSQL supporta un massimo di 63 caratteri per gli identific atori. Babelfish converte gli identificatori più lunghi di 63 caratteri in un nome che include un hash del nome originale. Ad esempio, una tabella creata come «AB (ABC12345678901234567890123 456789012345678901234567890123456789012345678 901234567890123456789012345678901234567890123 456789012345678901234567890123456789012345678901234567890". ABC123456789
Supporto delle colonne IDENTITY	Le colonne `IDENTITY` sono supportate per i tipi di dati `tinyint`, `smallint`, `int`, `bigint`. `numeric` e `decimal`. SQL Server supporta la precisione fino a 38 posizioni per i tipi di dati `numeric` e `decimal` nelle colonne IDENTITY. PostgreSQL supporta la precisione fino a 19 posizioni per i tipi di dati `numeric` e `decimal` nelle colonne IDENTITY.
Indici con IGNORE_DUP_KEY	La sintassi che crea un indice che include IGNORE_DUP_KEY crea un indice come se questa proprietà fosse omessa.
Indici con più di 32 colonne	Un indice non può includere più di 32 colonne. Le colonne di indice incluse contano fino al massimo in PostgreSQL ma non in SQL Server.
Indici (cluster)	Gli indici cluster vengono creati come se NON CLUSTERED fosse stato specificato.
Clausole Indice	Le seguenti clausole vengono ignorate: FILLFACTOR, ALLOW_PAGE_LOCKS, ALLOW_ROW_LOCKS, PAD_INDEX , STATISTICS_NORECOMPUTE, OPTIMIZE_FOR_SEQUE NTIAL_KEY, SORT_IN_TEMPDB, DROP_EXISTING, ONLINE, COMPRESSION_DELAY, MAXDOP e DATA_COMPRESSION
Supporto JSON	L'ordine delle coppie nome-valore non è garantito. Ma il tipo di array rimane inalterato.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Oggetti LOGIN	Tutte le opzioni per gli oggetti LOGIN non sono supportat e ad eccezione di PASSWORD, DEFAULT_DATABASE, DEFAULT_LANGUAGE, ENABLE, DISABLE.
La funzione NEWSEQUEN TIALID	Implementato come NEWID; il comportamento sequenziale non è garantito. Quando si richiama `NEWSEQUENTIALID` , PostgreSQL genera un nuovo valore GUID.
La clausola OUTPUT è supportata con le seguenti limitazioni	OUTPUT e OUTPUT INTO non sono supportati nella stessa query DML. I riferimenti alla tabella non di destinazione delle operazioni UPDATE o DELETE in una clausola OUTPUT non sono supportati. OUTPUT... DELETED *, INSERTED * non sono supportati nella stessa query.
Limite dei parametri di procedura o funzione	Babelfish supporta un massimo di 100 parametri per una procedura o una funzione.
ROWGUIDCOL	Al momento, questa clausola è ignorata. Le query che fanno riferimento a `$GUIDGOL` causano un errore di sintassi.
Supporto degli oggetti SEQUENCE	Gli oggetti SEQUENCE sono supportati per i tipi di dati tinyint, smallint, int, bigint, numerico e decimale. Aurora PostgreSQL supporta la precisione di 19 posizioni per i tipi di dati numerici e decimali in una SEQUENCE.
Ruoli a livello di server	Il ruolo a livello di server `sysadmin` è supportato. Altri ruoli a livello server (diversi da `sysadmin`) non sono supportati.
Ruoli a livello di database diversi da `db_owner`	Sono supportati i ruoli a livello di database `db_owner` e i ruoli a livello di database definiti dall'utente. Altri ruoli a livello server (diversi da db_owner) non sono supportati.
Parola chiave SQL `SPARSE`	La parola chiave SPARSE viene accettata e ignorata.
Clausola parola chiave SQL`ON filegroup`	Al momento, questa clausola è ignorata.

Funzionalità o sintassi	Descrizione del comportamento o della differenza
Parole chiave SQL `CLUSTERED` e `NONCLUSTERED` per indici e vincoli	Babelfish accetta e ignora le parole chiave `CLUSTERED` e `NONCLUSTERED`
`sysdatabases.cmptlevel`	`sysdatabases.cmptlevel` è sempre impostato su 120.
tempdb non viene reinizializzato al riavvio	Gli oggetti permanenti (come tabelle e procedure) creati in tempdb non vengono rimossi al riavvio del database.
Gruppo di file TEXTIMAGE_ON	Babelfish ignora la clausola `TEXTIMAGE_ON` *`filegroup`* .
Precisione temporale	Babelfish supporta la precisione a 6 cifre per secondi frazionari. Non sono previsti effetti avversi con questo comportamento.
Livelli di isolamento della transazione	READUNCOMMITTED è trattato allo stesso modo di READCOMMITTED.
Colonne computate virtuali (non persistenti)	Le colonne computate virtuali vengono create come persistenti.
Senza clausola SCHEMABINDING	Questa clausola non è supportata in funzioni, procedure, trigger o viste. L'oggetto viene creato, ma come se WITH SCHEMABINDING fosse stato specificato.

Livelli di isolamento delle transazioni in Babelfish

Babelfish supporta i livelli di isolamento delle transazioni, e. `READ UNCOMMITTED` `READ COMMITTED` `SNAPSHOT` A partire dalla versione Babelfish 3.4 sono supportati livelli `REPEATABLE READ` di isolamento aggiuntivi. `SERIALIZABLE` Tutti i livelli di isolamento di Babelfish sono supportati dal comportamento dei corrispondenti livelli di isolamento in Postgre. SQL SQL Server e Babelfish utilizzano diversi meccanismi sottostanti per implementare i livelli di isolamento delle transazioni (blocco per l'accesso simultaneo, blocchi bloccati dalle transazioni, gestione degli errori ecc.). Inoltre, ci sono alcune sottili differenze nel modo in cui l'accesso simultaneo può funzionare per diversi carichi di lavoro. Per ulteriori informazioni su questo SQL comportamento di Postgre, consulta Transaction Isolation.

Argomenti

- Panoramica dei livelli di isolamento delle transazioni
- Impostazione dei livelli di isolamento delle transazioni
- Abilitazione o disabilitazione dei livelli di isolamento delle transazioni
- Confronto tra i livelli di isolamento di Babelfish e SQL Server

Panoramica dei livelli di isolamento delle transazioni

I livelli di isolamento delle transazioni originali del SQL Server si basano su un blocco pessimistico in cui esiste solo una copia dei dati e le query devono bloccare risorse come le righe prima di accedervi. Successivamente, è stata introdotta una variazione del livello di isolamento. `READ COMMITTED` Ciò consente l'uso di versioni di riga per fornire una migliore concorrenza tra lettori e scrittori utilizzando un accesso non bloccante. Inoltre, è disponibile un nuovo livello di isolamento denominato`SNAPSHOT`. Utilizza inoltre le versioni di riga per fornire una concorrenza migliore rispetto al livello di `REPEATABLE READ` isolamento, evitando blocchi condivisi sui dati di lettura che vengono conservati fino alla fine della transazione.

A differenza di SQL Server, tutti i livelli di isolamento delle transazioni in Babelfish si basano su optimistic Locking (). MVCC Ogni transazione visualizza un'istantanea dei dati all'inizio dell'istruzione (`READ COMMITTED`) o all'inizio della transazione (`REPEATABLE READ`,`SERIALIZABLE`), indipendentemente dallo stato corrente dei dati sottostanti. Pertanto, il comportamento di esecuzione delle transazioni simultanee in Babelfish potrebbe differire da quello di Server. SQL

Ad esempio, si consideri una transazione con livello di isolamento inizialmente bloccata in SQL Server ma `SERIALIZABLE` che ha esito positivo in seguito. Potrebbe finire per fallire in Babelfish

a causa di un conflitto di serializzazione con una transazione simultanea che legge o aggiorna le stesse righe. Potrebbero esserci anche casi in cui l'esecuzione di più transazioni simultanee produca un risultato finale diverso in Babelfish rispetto a Server. SQL Le applicazioni che utilizzano livelli di isolamento devono essere accuratamente testate per verificare la presenza di scenari di concorrenza.

Livelli di isolamento nel server SQL	Livello di isolamento di Babelfish	Livello di isolamento SQL Postgree	Commenti
`READ UNCOMMITTED`	`READ UNCOMMITTED`	`READ UNCOMMITTED`	`READ UNCOMMITTED` è lo stesso di Babelfish o `READ COMMITTED` Postgre SQL
`READ COMMITTED`	`READ COMMITTED`	`READ COMMITTED`	SQL`READ COMMITTED` Il server è basato sul blocco pessimistico, Babelfish è basato su snapshot (). `READ COMMITTED` MVCC
`READ COMMITTED SNAPSHOT`	`READ COMMITTED`	`READ COMMITTED`	Entrambi sono basati su snapshot () MVCC ma non esattamente uguali.
`SNAPSHOT`	`SNAPSHOT`	`REPEATABLE READ`	Esattamente lo stesso.
`REPEATABLE READ`	`REPEATABLE READ`	`REPEATABLE READ`	SQL`REPEATABLE READ` Il server è basato sul blocco pessimistico, Babelfish `REPEATABLE READ` è basato su snapshot (). MVCC

Livelli di isolamento nel server SQL	Livello di isolamento di Babelfish	Livello di isolamento SQL Postgree	Commenti
`SERIALIZABLE`	`SERIALIZABLE`	`SERIALIZABLE`	SQL`SERIALIZABLE` Il server è basato su un isolamento pessimistico, Babelfish è basato su snapshot (). `SERIALIZABLE` MVCC

> **Note**
>
> I suggerimenti della tabella non sono attualmente supportati e il loro comportamento è controllato utilizzando l'escape hatch predefinito di Babelfish. `escape_hatch_table_hints`

Impostazione dei livelli di isolamento delle transazioni

Utilizzate il seguente comando per impostare il livello di isolamento delle transazioni:

Example

```
SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |
 SNAPSHOT | SERIALIZABLE }
```

Abilitazione o disabilitazione dei livelli di isolamento delle transazioni

I livelli di `REPEATABLE READ` isolamento delle transazioni `SERIALIZABLE` sono disabilitati di default in Babelfish e devi abilitarli esplicitamente impostando `babelfishpg_tsql.isolation_level_serializable` o `babelfishpg_tsql.isolation_level_repeatable_read` escape hatch su use. `pg_isolation` `sp_babelfish_configure` Per ulteriori informazioni, consulta Gestione degli errori di Babelfish con escape hatch.

Di seguito sono riportati alcuni esempi per abilitare o disabilitare l'uso di `REPEATABLE READ` e `SERIALIZABLE` nella sessione corrente impostando i rispettivi escape hatch. Facoltativamente, includi il `server` parametro per impostare l'escape hatch per la sessione corrente e per tutte le nuove sessioni successive.

Per abilitare l'uso di solo `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` nella sessione corrente.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation'
```

Per consentire l'uso di `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` nella sessione corrente e in tutte le nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation',
 'server'
```

Per disabilitare l'uso della `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` sessione corrente e delle nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'off', 'server'
```

Per abilitare l'uso di solo `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` nella sessione corrente.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation'
```

Per consentire l'uso di `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` nella sessione corrente e in tutte le nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation', 'server'
```

Per disabilitare l'uso della `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` sessione corrente e delle nuove sessioni successive.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'off', 'server'
```

Confronto tra i livelli di isolamento di Babelfish e SQL Server

Di seguito sono riportati alcuni esempi sulle sfumature del modo in cui SQL Server e Babelfish implementano i livelli di isolamento. ANSI

> **Note**
>
> - I livelli di isolamento `SNAPSHOT` sono `REPEATABLE READ` gli stessi in Babelfish.
> - Il livello di `READ UNCOMMITTED` isolamento è lo stesso in Babelfish. `READ COMMITTED`

L'esempio seguente mostra come creare la tabella di base per tutti gli esempi menzionati di seguito:

```
CREATE TABLE employee (
    id sys.INT NOT NULL PRIMARY KEY,
    name sys.VARCHAR(255)NOT NULL,
    age sys.INT NOT NULL
);
INSERT INTO employee (id, name, age) VALUES (1, 'A', 10);
INSERT INTO employee (id, name, age) VALUES (2, 'B', 20);
INSERT INTO employee (id, name, age) VALUES (3, 'C', 30);
```

Argomenti

- Babelfish READ UNCOMMITTED rispetto al livello di isolamento SQL del server READ UNCOMMITTED

- Babelfish READ COMMITTED rispetto al livello di isolamento SQL del server READ COMMITTED
- Babelfish READ COMMITTED rispetto al livello di isolamento SQL del server READ COMMITTED SNAPSHOT
- Babelfish REPEATABLE READ rispetto al livello di isolamento del server SQL REPEATABLE READ
- Babelfish SERIALIZABLE rispetto al livello di isolamento SQL del server SERIALIZABLE

Babelfish **READ UNCOMMITTED** rispetto al livello di isolamento SQL del server **READ UNCOMMITTED**

La tabella seguente fornisce dettagli sulle letture sporche quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `READ UNCOMMITTED` isolamento in SQL Server rispetto all'implementazione Babelfish.

Transazione 1	Transazione 2	SQLServer **READ UNCOMMITTED**	Babelfish **READ UNCOMMITTED**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	Nessuna	Nessuno
Inattivo in transazione	`UPDATE employee SET age=0;`	Aggiornamento riuscito.	Aggiornamento riuscito.
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 40);`	Inserimento riuscito.	Inserimento riuscito.
`SELECT * FROM employee;`	Inattivo in transazione	La transazione 1 può visualizzare modifiche non confermate rispetto alla transazione 2.	Come `READ COMMITTED` in Babelfish. Le modifiche non confermate dalla transazione 2 non

Transazione 1	Transazione 2	SQLServer **READ UNCOMMITTED**	Babelfish **READ UNCOMMITTED**
			sono visibili nella transazione 1.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Visualizza le modifiche apportate dalla transazione 2.	Visualizza le modifiche apportate dalla transazione 2.

Babelfish **`READ COMMITTED`** rispetto al livello di isolamento SQL del server **`READ COMMITTED`**

La tabella seguente fornisce dettagli sul comportamento di blocco di lettura-scrittura quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `READ COMMITTED` isolamento in SQL Server rispetto all'implementazione Babelfish.

Transazione 1	Transazione 2	SQLServer **READ COMMITTED**	Babelfish **READ COMMITTED**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`UPDATE employee SET age=100 WHERE id = 1;`	Aggiornamento riuscito.	Aggiornamento riuscito.
`UPDATE employee SET age = 0 WHERE age IN`	Inattivo in transazione	Fase bloccata fino al completamento della transazione 2.	Le modifiche alla transazione 2 non sono ancora visibili.

Transazione 1	Transazione 2	SQLServer **READ COMMITTED**	Babelfish **READ COMMITTED**
`(SELECT MAX(age) FROM employee);`			Aggiorna la riga con id=3.
Inattivo in transazione	`COMMIT`	La transazione 2 viene eseguita correttamente. La transazione 1 è ora sbloccata e vede l'aggiornamento della transazione 2.	La transazione 2 viene eseguita correttamente.
`SELECT * FROM employee;`	Inattivo in transazione	La transazione 1 aggiorna la riga con id = 1.	La transazione 1 aggiorna la riga con id = 3.

Babelfish **`READ COMMITTED`** rispetto al livello di isolamento SQL del server **`READ COMMITTED SNAPSHOT`**

La tabella seguente fornisce dettagli sul comportamento di blocco delle righe appena inserite quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `READ COMMITTED SNAPSHOT` isolamento in SQL Server rispetto all'implementazione `READ COMMITTED` Babelfish.

Transazione 1	Transazione 2	SQLServer **READ COMMITTED SNAPSHOT**	Babelfish **READ COMMITTED**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Nessuna	Nessuno

Transazione 1	Transazione 2	SQLServer **READ COMMITTED SNAPSHOT**	Babelfish **READ COMMITTED**
`INSERT INTO employee VALUES (4, 'D', 40);`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`UPDATE employee SET age = 99;`	Il passaggio è bloccato fino al completamento della transazione 1. La riga inserita è bloccata dalla transazione 1.	Tre righe aggiornate. La riga appena inserita non è ancora visibile.
`COMMIT`	Inattivo in transazione	Impegno riuscito. La transazione 2 è ora sbloccata.	Impegno riuscito.
Inattivo in transazione	`SELECT * FROM employee;`	Tutte e 4 le righe hanno età = 99.	La riga con id = 4 ha il valore di età 40 poiché non era visibile alla transazione 2 durante la query di aggiornamento. Le altre righe vengono aggiornate a age=99.

Babelfish **REPEATABLE READ** rispetto al livello di isolamento del server SQL **REPEATABLE READ**

La tabella seguente fornisce dettagli sul comportamento di blocco di lettura-scrittura quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `REPEATABLE READ` isolamento in SQL Server rispetto all'implementazione Babelfish. `REPEATABLE READ`

Transazione 1	Transazione 2	SQLServer **REPEATABLE READ**	Babelfish **REPEATABLE READ**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee WHERE id != 1;`	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	La transazione 2 è bloccata fino al completamento della transazione 1.	La transazione 2 procede normalmente.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	L'aggiornamento dalla transazione 1 è visibile.	L'aggiornamento dalla transazione 1 non è visibile.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	vede l'aggiornamento della transazione 1.	vede l'aggiornamento della transazione 1.

La tabella seguente fornisce dettagli sul comportamento di blocco scrittura-scrittura quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `REPEATABLE READ` isolamento in SQL Server rispetto all'implementazione Babelfish. `REPEATABLE READ`

Transazione 1	Transazione 2	SQLServer **`REPEATABLE READ`**	Babelfish **`REPEATABLE READ`**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`UPDATE employee SET name='A_TXN2' WHERE id=1;`	Transazione 2 bloccata.	Transazione 2 bloccata.
`COMMIT`	Inattivo in transazione	Il commit è riuscito e la transazione 2 è stata sbloccata.	Commit riuscito e la transazione 2 fallisce con errore, impossibile serializzare l'accesso a causa di un aggiornamento simultaneo.
Inattivo in transazione	`COMMIT`	Esecuzione riuscita.	La transazione 2 è già stata interrotta.
Inattivo in transazione	`SELECT * FROM employee;`	La riga con id=1 ha name='A_ '. TX2	La riga con id=1 ha name='A_ '. TX1

La tabella seguente fornisce dettagli sul comportamento delle letture nascoste quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `REPEATABLE READ` isolamento in SQL Server rispetto all'implementazione Babelfish. `REPEATABLE READ`

Transazione 1	Transazione 2	SQLServer **`REPEATABLE READ`**	Babelfish **`REPEATABLE READ`**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'NewRowName', 20);`	La transazione 2 procede senza alcun blocco.	La transazione 2 procede senza alcun blocco.
Inattivo in transazione	`SELECT * FROM employee;`	La riga appena inserita è visibile.	La riga appena inserita è visibile.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La nuova riga inserita dalla transazione 2 è visibile.	La nuova riga inserita dalla transazione 2 non è visibile.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La riga appena inserita è visibile.	La riga appena inserita è visibile.

La tabella seguente fornisce dettagli su quando vengono eseguite transazioni simultanee e i diversi risultati finali quando si utilizza il livello di `REPEATABLE READ` isolamento in SQL Server rispetto all'implementazione `REPEATABLE READ` Babelfish.

Transazione 1	Transazione 2	SQLServer **`REPEATABLE READ`**	Babelfish **`REPEATABLE READ`**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno
`UPDATE employee SET age = 100 WHERE age IN (SELECT MIN(age) FROM employee);`	Inattivo in transazione	La transazione 1 aggiorna la riga con id 1.	La transazione 1 aggiorna la riga con id 1.
Inattivo in transazione	`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	La transazione 2 è bloccata poiché l'`SELECT`istruzione tenta di leggere le righe bloccate dalla `UPDATE` query nella transazione 1.	La transazione 2 procede senza alcun blocco poiché la lettura non viene mai bloccata, l'`SELECT`istruzione viene eseguita e infine la riga con id = 3 viene aggiornata poiché le modifiche alla transazione 1 non sono ancora visibili.
Inattivo in transazione	`SELECT * FROM employee;`	Questo passaggio viene eseguito dopo il completamento della transazione 1.	La riga con id = 3 viene aggiornata dalla transazione 2.

Transazione 1	Transazione 2	SQLServer **REPEATABLE READ**	Babelfish **REPEATABLE READ**
		La riga con id = 1 viene aggiornata dalla transazione 2 nel passaggio precedente ed è visibile qui.	
`COMMIT`	Inattivo in transazione	La transazione 2 è ora sbloccata.	Impegno riuscito.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Entrambe le transazioni eseguono l'aggiornamento sulla riga con id = 1.	Le diverse righe vengono aggiornate dalle transazioni 1 e 2.

Babelfish **SERIALIZABLE** rispetto al livello di isolamento SQL del server **SERIALIZABLE**

La tabella seguente fornisce dettagli sugli intervalli di blocco quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `SERIALIZABLE` isolamento in SQL Server rispetto all'implementazione `SERIALIZABLE` Babelfish.

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 35);`	La transazione 2 è bloccata fino al completamento della transazione 1.	La transazione 2 procede senza alcun blocco.
Inattivo in transazione	`SELECT * FROM employee;`	Nessuno	Nessuno
`COMMIT`	Inattivo in transazione	La transazione 1 viene eseguita correttamente. La transazione 2 è ora sbloccata.	La transazione 1 viene eseguita correttamente.
Inattivo in transazione	`COMMIT`	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La riga appena inserita è visibile.	La riga appena inserita è visibile.

La tabella seguente fornisce dettagli su quando vengono eseguite transazioni simultanee e i diversi risultati finali quando si utilizza il livello di `SERIALIZABLE` isolamento in SQL Server rispetto all'implementazione `SERIALIZABLE` Babelfish.

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 40);`	Nessuno	Nessuno
`UPDATE employee SET age =99 WHERE id = 4;`	Inattivo in transazione	La transazione 1 è bloccata fino al completamento della transazione 2.	La transazione 1 procede senza alcun blocco.
Inattivo in transazione	`COMMIT`	La transazione 2 viene eseguita correttamente. La transazione 1 è ora sbloccata.	La transazione 2 viene eseguita correttamente.
`COMMIT`	Inattivo in transazione	Nessuno	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	La riga appena inserita è visibile con valore di età = 99.	La riga appena inserita è visibile con valore di età = 40.

La tabella seguente fornisce i dettagli relativi `INSERT` all'accesso a una tabella con vincolo univoco. Mostra i risultati osservati quando si utilizza il livello di `SERIALIZABLE` isolamento in SQL Server rispetto all'implementazione `SERIALIZABLE` Babelfish.

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
Inattivo in transazione	`INSERT INTO employee VALUES (4, 'D', 40);`	Nessuno	Nessuno
`INSERT INTO employee VALUES ((SELECT MAX(id)+1 FROM employee), 'E', 50);`	Inattivo in transazione	La transazione 1 è bloccata fino al completamento della transazione 2.	La transazione 1 è bloccata fino al completamento della transazione 2.
Inattivo in transazione	`COMMIT`	La transazione 2 viene eseguita correttamente. La transazione 1 è ora sbloccata.	La transazione 2 viene eseguita correttamente. La transazione 1 interrotta con errore «valore chiave duplicato » viola un vincolo univoco.
`COMMIT`	Inattivo in transazione	La transazione 1 viene eseguita correttamente.	Il commit della transazione 1 ha esito negativo e non è stato possibile serializzare l'accesso a causa di dipendenze di lettura o scrittura tra le transazioni.
`SELECT * FROM employee;`	Inattivo in transazione	viene inserita la riga (5, 'E', 50).	Esistono solo 4 righe.

In Babelfish, le transazioni simultanee eseguite con Isolation Level serializable falliranno con un errore di anomalia di serializzazione se l'esecuzione di queste transazioni non è coerente con tutte le possibili esecuzioni seriali (una alla volta) di tali transazioni.

Le tabelle seguenti forniscono dettagli sull'anomalia di serializzazione quando vengono eseguite transazioni simultanee. Mostra i risultati osservati quando si utilizza il livello di `SERIALIZABLE` isolamento in SQL Server rispetto all'implementazione Babelfish. `SERIALIZABLE`

Transazione 1	Transazione 2	SQLServer **`SERIALIZABLE`**	Babelfish **`SERIALIZABLE`**
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Nessuna	Nessuno
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
`UPDATE employee SET age=5 WHERE age=10;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	La transazione 2 è bloccata fino al completamento della transazione 1.	La transazione 2 procede senza alcun blocco.
Inattivo in transazione	`UPDATE employee SET age=35 WHERE age=30;`	Nessuno	Nessuno
`COMMIT`	Inattivo in transazione	La transazione 1 viene eseguita correttamente.	La transazione 1 viene confermata per prima ed è in

Transazione 1	Transazione 2	SQLServer **SERIALIZABLE**	Babelfish **SERIALIZABLE**
			grado di eseguirla con successo.
Inattivo in transazione	`COMMIT`	La transazione 2 viene eseguita correttamente.	Il commit della transazione 2 non riesce a causa di un errore di serializzazione, l'intera transazione è stata annullata. Riprova la transazione 2.
`SELECT * FROM employee;`	Inattivo in transazione	Le modifiche di entrambe le transazioni sono visibili.	La transazione 2 è stata annullata. Vengono visualizzate solo le modifiche alla transazione 1.

In Babelfish, l'anomalia di serializzazione è possibile solo se tutte le transazioni simultanee vengono eseguite a livello di isolamento. `SERIALIZABLE` Nella tabella seguente, prendiamo l'esempio precedente ma impostiamo invece la transazione 2 sul livello di isolamento. `REPEATABLE READ`

Transazione 1	Transazione 2	SQLLivelli di isolamento del server	Livelli di isolamento di Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Nessuno	Nessuna
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Nessuna	Nessuno

Transazione 1	Transazione 2	SQLLivelli di isolamento del server	Livelli di isolamento di Babelfish
`SELECT * FROM employee;`	Inattivo in transazione	Nessuno	Nessuno
`UPDATE employee SET age=5 WHERE age=10;`	Inattivo in transazione	Nessuno	Nessuno
Inattivo in transazione	`SELECT * FROM employee;`	La transazione 2 è bloccata fino al completamento della transazione 1.	La transazione 2 procede senza alcun blocco.
Inattivo in transazione	`UPDATE employee SET age=35 WHERE age=30;`	Nessuno	Nessuno
`COMMIT`	Inattivo in transazione	La transazione 1 viene eseguita correttamente.	La transazione 1 viene eseguita correttamente.
Inattivo in transazione	`COMMIT`	La transazione 2 viene eseguita correttamente.	La transazione 2 viene eseguita correttamente.
`SELECT * FROM employee;`	Inattivo in transazione	Le modifiche di entrambe le transazio ni sono visibili.	Le modifiche di entrambe le transazio ni sono visibili.

Utilizzo di funzionalità Babelfish con implementazione limitata

Ogni nuova versione di Babelfish aggiunge il supporto per ulteriori funzionalità che si allineano meglio alle funzionalità e al comportamento di T-SQL. Tuttavia, ci sono alcune caratteristiche e differenze non supportate nell'implementazione corrente. Di seguito, puoi trovare informazioni sulle differenze funzionali tra Babelfish e T-SQL, con alcune soluzioni alternative o note di utilizzo.

A partire dalla versione 1.2.0 di Babelfish, le seguenti funzionalità attualmente presentano implementazioni limitate:

- Cataloghi SQL Server (visualizzazioni di sistema) - I cataloghi `sys.sysconfigures`,`sys.syscurconfigs` e `sys.configurations`supportano solo una singola configurazione di sola lettura. `sp_configure` non è attualmente supportato. Per ulteriori informazioni sulle altre viste di SQL Server implementate da Babelfish, consulta Recupero di informazioni dal catalogo di sistema Babelfish.
- Autorizzazioni GRANT - GRANT...TO PUBLIC è supportato, ma GRANT..TO PUBLIC WITH GRANT OPTION non è attualmente supportato.
- Limitazioni sulla catena di proprietà e sul meccanismo di autorizzazione di SQL Server - In Babelfish, la catena di proprietà di SQL Server funziona per le viste ma non per le stored procedure. Ciò significa che alle procedure deve essere concesso un accesso esplicito ad altri oggetti di proprietà dello stesso proprietario delle procedure chiamanti. In SQL Server, concedere al chiamante le autorizzazioni EXECUTE sulla procedura è sufficiente per invocare altri oggetti di proprietà dello stesso proprietario. In Babelfish, al chiamante devono inoltre essere concesse le autorizzazioni sugli oggetti a cui si accede dalla procedura.
- Risoluzione di riferimenti agli oggetti non qualificati (senza nome dello schema) - Quando un oggetto SQL (procedura, vista, funzione o trigger) fa riferimento a un oggetto senza qualificarlo con un nome di schema, SQL Server risolve il nome dello schema dell'oggetto utilizzando il nome dello schema dell'oggetto SQL in cui si verifica il riferimento. Attualmente, Babelfish risolve questo problema in modo diverso, utilizzando lo schema predefinito dell'utente del database che esegue la procedura.
- Modifiche di schema, sessioni e connessioni predefinite - Se gli utenti modificano lo schema predefinito con`ALTER USER...WITH DEFAULT SCHEMA` , la modifica diventa immediatamente effettiva in quella sessione. Tuttavia, per altre sessioni attualmente connesse appartenenti allo stesso utente, la tempistica è diversa, come segue:
  - Per SQL Server: la modifica ha effetto su tutte le altre connessioni per questo utente immediatamente.
  - Per Babelfish: la modifica ha effetto solo per questo utente per le nuove connessioni.
- Implementazione dei tipi di dati ROWVERSION e TIMESTAMP e impostazione di escape hatch - I tipi di dati ROWVERSION e TIMESTAMP sono ora supportati in Babelfish. Per utilizzare ROWVERSION o TIMESTAMP in Babelfish, è necessario modificare l'impostazione dell'escape hatch `babelfishpg_tsql.escape_hatch_rowversion` dal suo valore predefinito (strict) a

`ignore`. L'implementazione Babelfish dei tipi di dati ROWVERSION e TIMESTAMP è per lo più semanticamente identica a quella di SQL Server, con le seguenti eccezioni:

- La funzione @@DBTS integrata si comporta in modo simile a SQL Server, ma con piccole differenze. Invece di restituire l'ultimo valore utilizzato per `SELECT @@DBTS`, Babelfish genera un nuovo timestamp, a causa del motore di database PostgreSQL sottostante e della sua implementazione del controllo della concorrenza multiversione (MVCC).
- In SQL Server, ogni riga inserita o aggiornata ottiene un ROWVERSION/TIMESTAMP value. In Babelfish, every inserted row updated by the same statement is assigned the same ROWVERSION/TIMESTAMP valore univoco.

  Ad esempio, quando un'istruzione UPDATE o un'istruzione INSERT-SELECT influisce su più righe, in SQL Server, le righe interessate hanno valori diversi nella colonna ROWVERSION/TIMESTAMP. In Babelfish (PostgreSQL), le righe hanno lo stesso valore.
- In SQL Server, quando si crea una nuova tabella con SELECT-INTO, è possibile assegnare un valore esplicito (come NULL) a un to-be-created ROWVERSION/TIMESTAMP column. When you do the same thing in Babelfish, an actual ROWVERSION/TIMESTAMP valore assegnato da Babelfish a ogni riga della nuova tabella.

Queste differenze minori nei tipi di dati ROWVERSION/TIMESTAMP non dovrebbero avere un impatto negativo sulle applicazioni in esecuzione su Babelfish.

Creazione, proprietà e autorizzazioni dello schema - Le autorizzazioni per creare e accedere a oggetti in uno schema di proprietà di un utente non DBO (utilizzando `CREATE SCHEMA` *`schema name`* `AUTHORIZATION` *`user name`*) differiscono per gli utenti SQL Server e Babelfish non DBO, come illustrato nella seguente tabella:

L'utente del database (non DBO) proprietario dello schema può eseguire le seguenti operazioni:	SQL Server	Babelfish
Creare oggetti nello schema senza ulteriori autorizzazioni da parte del DBO?	No	Sì
Accedere a oggetti creati dal DBO nello schema senza ulteriori autorizzazioni?	Sì	No

Miglioramento delle prestazioni della query Babelfish

Puoi velocizzare l'elaborazione delle query in Babelfish utilizzando i suggerimenti per le query e il sistema di ottimizzazione PostgreSQL.

Argomenti

- Utilizzo del piano explain per migliorare le prestazioni delle query Babelfish
- Utilizzo di suggerimenti per le query T-SQL per migliorare le prestazioni delle query Babelfish

Puoi anche migliorare le prestazioni delle query utilizzando la procedura `sp_babelfish_volatility`. Per ulteriori informazioni, consulta sp_babelfish_volatility.

È inoltre possibile migliorare le prestazioni delle query utilizzando la trasformazione delle subquery e la cache delle subquery. Per ulteriori informazioni, consulta Ottimizzazione delle sottoquery correlate in Aurora PostgreSQL.

Utilizzo del piano explain per migliorare le prestazioni delle query Babelfish

A partire dalla versione 2.1.0, Babelfish include due funzioni che utilizzano in modo trasparente l'ottimizzatore PostgreSQL per generare piani di query effettivi per query T-SQL sulla porta TDS. Queste funzioni sono simili all'utilizzo di SET STATISTICS PROFILE o SET SHOWPLAN_ALL con i database SQL Server per identificare e migliorare le query a esecuzione lenta.

> **Note**
>
> Il recupero di piani di query da funzioni, flussi di controllo e cursori non è attualmente supportato.

Nella tabella è disponibile un confronto tra le funzioni explain del piano di query in SQL Server, Babelfish e PostgreSQL.

SQL Server	Babelfish	PostgreSQL
SHOWPLAN_ALL	BABELFISH_SHOWPLAN_ALL	EXPLAIN

SQL Server	Babelfish	PostgreSQL
STATISTICS PROFILE	BABELFISH_STATISTICS PROFILE	EXPLAIN ANALYZE
Utilizza l'ottimizzatore SQL Server	Utilizza l'ottimizzatore PostgreSQL	Utilizza l'ottimizzatore PostgreSQL
Formato di input e output di SQL Server	Formato di input e output di SQL Server	Formato di input e output di PostgreSQL
Impostato per la sessione	Impostato per la sessione	Si applica a un'istruzione specifica
Supporta quanto segue: • SELECT • INSERT • UPDATE • DELETE • CURSOR • CREATE • EXECUTE • EXEC e funzioni, incluso il flusso di controllo (CASE, WHILE-BREAK-CONTINUE, WAITFOR, BEGIN-END, IF-ELSE e così via)	Supporta quanto segue: • SELECT • INSERT • UPDATE • DELETE • CREATE • EXECUTE • EXEC • RAISEERROR • THROW • PRINT • USE	Supporta quanto segue: • SELECT • INSERT • UPDATE • DELETE • CURSOR • CREATE • EXECUTE

Utilizza le funzioni Babelfish come segue:

- SET BABELFISH_SHOWPLAN_ALL [ON|OFF]: imposta su ON per generare un piano di esecuzione delle query stimato. Questa funzione implementa il comportamento del comando `EXPLAIN` PostgreSQL. Utilizza questo comando per ottenere il piano explain per la query specificata.

- SET BABELFISH_STATISTICS PROFILE [ON|OFF]: imposta su ON per i piani di esecuzione delle query effettivi. Questa funzione implementa il comportamento del comando `EXPLAIN ANALYZE` PostgreSQL.

Per ulteriori informazioni su `EXPLAIN` e `EXPLAIN ANALYZE` PostgreSQL, consulta EXPLAIN nella documentazione di PostgreSQL.

> **Note**
>
> A partire dalla versione 2.2.0, è possibile impostare il parametro `escape_hatch_showplan_all` su ignore per evitare l'uso del prefisso BABELFISH_ nella sintassi di SQL Server per i comandi SET `SHOWPLAN_ALL` e `STATISTICS PROFILE`.

Ad esempio, la sequenza di comandi seguente attiva la pianificazione delle query e quindi restituisce un piano di esecuzione delle query stimato per l'istruzione SELECT senza eseguire la query. In questo esempio, viene utilizzato il database SQL Server `northwind` di esempio che utilizza lo strumento a riga di comando `sqlcmd` per eseguire query sulla porta TDS:

```
1> SET BABELFISH_SHOWPLAN_ALL ON
2> GO
1> SELECT t.territoryid, e.employeeid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE e.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO

QUERY PLAN


------------------------------------------------------------------------------


Query Text: SELECT t.territoryid, e.employeeid FROM
dbo.employeeterritories e, dbo.territories t
WHERE e.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=6231.74..6399.22 rows=66992 width=10)
  Sort Key: t.territoryid NULLS FIRST
  ->  Nested Loop  (cost=0.00..861.76 rows=66992 width=10)
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1264 width=4)
              Filter: ((territoryid)::"varchar" IS NOT NULL)
```

```
        ->  Materialize  (cost=0.00..1.79 rows=53 width=6)
              ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Al termine della revisione e della regolazione della query, disattiva la funzione come illustrato di seguito:

```
1> SET BABELFISH_SHOWPLAN_ALL OFF
```

Con BABELFISH_STATISTICS PROFILE impostato su ON, ogni query eseguita restituisce il set di risultati regolare seguito da un set di risultati aggiuntivo che mostra i piani di esecuzione effettivi delle query. Babelfish genera il piano di query che fornisce il set di risultati più rapido quando richiama l'istruzione SELECT.

```
1> SET BABELFISH_STATISTICS PROFILE ON
1>
2> GO
1> SELECT e.employeeid, t.territoryid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE t.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO
```

Vengono restituiti il set di risultati e il piano di query (questo esempio mostra solo il piano di query).

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT e.employeeid, t.territoryid FROM
dbo.employeeterritories e, dbo.territories t
WHERE t.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=42.44..43.28 rows=337 width=10)
  Sort Key: t.territoryid NULLS FIRST


  ->  Hash Join  (cost=2.19..28.29 rows=337 width=10)
       Hash Cond: ((e.territoryid)::"varchar" = (t.territoryid)::"varchar")
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1270 width=36)
        ->  Hash  (cost=1.53..1.53 rows=53 width=6)
             ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Per ulteriori informazioni su come analizzare le query e i risultati restituiti dall'ottimizzatore PostgreSQL, consulta explain.depesz.com. Per ulteriori informazioni su PostgreSQL EXPLAIN e EXPLAIN ANALYZE, consulta EXPLAIN nella documentazione di PostgreSQL.

Parametri che controllano le opzioni explain di Babelfish

Puoi utilizzare i parametri mostrati nella tabella seguente per controllare il tipo di informazioni visualizzate dal piano di query.

Parametro	Descrizione
babelfishpg_tsql.explain_buffers	Un valore booleano che attiva (e disattiva) le informazioni sull'utilizzo del buffer per l'ottimizzatore. (Predefinito: off) (Consentito: off, on)
babelfishpg_tsql.explain_costs	Un valore booleano che attiva (e disattiva) le informazioni di avvio e del totale dei costi stimate per l'ottimizzatore. (Predefinito: on) (Consentito: off, on)
babelfishpg_tsql.explain_format	Specifica il formato di output per il piano `EXPLAIN`. (Predefinito: text) (Consentito: text, xml, json, yaml)
babelfishpg_tsql.explain_settings	Un valore booleano che attiva (o disattiva) l'inclusione di informazioni sui parametri di configurazione nell'output del piano EXPLAIN. (Predefinito: off) (Consentito: off, on)
babelfishpg_tsql.explain_summary	Un valore booleano che attiva (o disattiva) informazioni di riepilogo come il tempo totale dopo il piano di query. (Predefinito: on) (Consentito: off, on)
babelfishpg_tsql.explain_timing	Un valore booleano che attiva (o disattiva) il tempo di avvio effettivo e il tempo trascorso in ciascun nodo dell'output. (Predefinito: on) (Consentito: off, on)

Parametro	Descrizione
babelfishpg_tsql.explain_verbose	Un valore booleano che attiva (o disattiva) la versione più dettagliata di un piano explain. (Predefinito: off) (Consentito: off, on)
babelfishpg_tsql.explain_wal	Un valore booleano che attiva (o disattiva) la generazione di informazioni sui record WAL come parte di un piano explain. (Predefinito: off) (Consentito: off, on)

Puoi verificare i valori di qualsiasi parametro relativo a Babelfish sul sistema utilizzando il client PostgreSQL o il client SQL Server. Esegui il comando seguente per ottenere i valori dei parametri correnti:

```
1> execute sp_babelfish_configure '%explain%';
2> GO
```

Nell'output seguente, puoi vedere che tutte le impostazioni di questo particolare cluster database Babelfish sono sui loro valori predefiniti. Non tutto l'output è mostrato in questo esempio.

```
            name                   setting                    short_desc
---------------------------------- --------
 ---------------------------------------------------------
babelfishpg_tsql.explain_buffers   off      Include information on buffer usage
babelfishpg_tsql.explain_costs     on       Include information on estimated startup
 and total cost
babelfishpg_tsql.explain_format    text     Specify the output format, which can be
 TEXT, XML, JSON, or YAML
babelfishpg_tsql.explain_settings  off      Include information on configuration
 parameters
babelfishpg_tsql.explain_summary   on       Include summary information (e.g.,totaled
 timing information) after the query plan
babelfishpg_tsql.explain_timing    on       Include actual startup time and time spent
 in each node in the output
babelfishpg_tsql.explain_verbose   off      Display additional information regarding
 the plan
babelfishpg_tsql.explain_wal       off      Include information on WAL record
 generation
```

```
(8 rows affected)
```

Puoi modificare l'impostazione di questi parametri utilizzando `sp_babelfish_configure`, come mostrato nell'esempio seguente.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on';
2> GO
```

Per rendere le impostazioni permanenti a livello di cluster, includi la parola chiave server, come nell'esempio seguente.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on', 'server';
2> GO
```

Utilizzo di suggerimenti per le query T-SQL per migliorare le prestazioni delle query Babelfish

A partire dalla versione 2.3.0, Babelfish supporta l'uso dei suggerimenti per le query utilizzando `pg_hint_plan`. In Aurora PostgreSQL, `pg_hint_plan` è installato per impostazione predefinita. Per ulteriori informazioni sull'estensione PostgreSQL `pg_hint_plan`, consulta https://github.com/ossc-db/pg_hint_plan. Per informazioni dettagliate sulla versione di questa estensione supportata da Aurora PostgreSQL, consulta Extension versions for Amazon Aurora PostgreSQL (Versioni delle estensioni di Amazon Aurora PostgreSQL) nelle Note di rilascio per Aurora PostgreSQL.

L'ottimizzatore di query è progettato per trovare il piano di esecuzione ottimale per un'istruzione SQL. Quando si seleziona un piano, l'ottimizzatore di query considera sia il modello di costo del motore sia le statistiche di colonne e tabelle. Tuttavia, il piano suggerito potrebbe non soddisfare le esigenze dei tuoi set di dati. Pertanto, i suggerimenti di query risolvono i problemi di prestazioni per migliorare i piani di esecuzione. Un `query hint` è una sintassi aggiunta allo standard SQL che indica al motore di database come eseguire la query. Ad esempio, un suggerimento può indicare al motore di seguire una scansione sequenziale e sostituire qualsiasi piano selezionato dall'ottimizzatore di query.

Attivazione dei suggerimenti di query T-SQL in Babelfish

Attualmente, Babelfish ignora tutti i suggerimenti T-SQL per impostazione predefinita. Per applicare i suggerimenti T-SQL, esegui il comando `sp_babelfish_configure` con il valore enable_pg_hint impostato su ON (Attivato).

```
EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on' [, 'server']
```

Puoi rendere le impostazioni permanenti a livello di cluster includendo la parola chiave server. Per configurare l'impostazione solo per la sessione corrente, non utilizzare la parola chiave server.

Dopo aver impostato `enable_pg_hint` su ON (Attivato), Babelfish applica i seguenti suggerimenti T-SQL.

- Suggerimenti INDEX
- Suggerimenti JOIN
- Suggerimento FORCE ORDER
- Suggerimento MAXDOP

Ad esempio, la seguente sequenza di comandi attiva `pg_hint_plan`

```
1> CREATE TABLE t1 (a1 INT PRIMARY KEY, b1 INT);
2> CREATE TABLE t2 (a2 INT PRIMARY KEY, b2 INT);
3> GO
1> EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on';
2> GO
1> SET BABELFISH_SHOWPLAN_ALL ON;
2> GO
1> SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2; --NO HINTS (HASH JOIN)
2> GO
```

Nessun suggerimento viene applicato all'istruzione SELECT. Viene restituito il piano di query senza alcun suggerimento.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2
Hash Join (cost=60.85..99.39 rows=2260 width=16)
 Hash Cond: (t1.a1 = t2.a2)
 -> Seq Scan on t1 (cost=0.00..32.60 rows=2260 width=8)
 -> Hash (cost=32.60..32.60 rows=2260 width=8)
 -> Seq Scan on t2 (cost=0.00..32.60 rows=2260 width=8)
```

```
1> SELECT * FROM t1 INNER MERGE JOIN t2 ON t1.a1 = t2.a2;
2> GO
```

Il suggerimento di query viene applicato all'istruzione SELECT. L'output seguente mostra che viene restituito il piano di query con Merge Join.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT/*+ MergeJoin(t1 t2) Leading(t1 t2)*/ * FROM t1 INNER JOIN t2 ON
 t1.a1 = t2.a2
Merge Join (cost=0.31..190.01 rows=2260 width=16)
 Merge Cond: (t1.a1 = t2.a2)
 -> Index Scan using t1_pkey on t1 (cost=0.15..78.06 rows=2260 width=8)
 -> Index Scan using t2_pkey on t2 (cost=0.15..78.06 rows=2260 width=8)
```

```
1> SET BABELFISH_SHOWPLAN_ALL OFF;
2> GO
```

Restrizioni

Per utilizzare i suggerimenti di query, considera le seguenti restrizioni:

- Se un piano di query viene memorizzato nella cache prima dell'attivazione di `enable_pg_hint`, i suggerimenti non vengono applicati nella stessa sessione, ma nella nuova sessione.
- Se i nomi di schema vengono forniti in modo esplicito, i suggerimenti non possono essere applicati. È possibile utilizzare gli alias delle tabelle come soluzione alternativa.
- Un suggerimento di query non può essere applicato a viste e query secondarie.
- I suggerimenti non funzionano per le istruzioni UPDATE/DELETE con JOIN.
- Un suggerimento di indice per una tabella o un indice inesistente viene ignorato.
- Il suggerimento FORCE ORDER non funziona per gli HASH JOIN e i JOIN non ANSI.

Utilizzo delle estensioni Aurora PostgreSQL con Babelfish

Aurora PostgreSQL fornisce estensioni per lavorare con altri servizi. AWS Si tratta di estensioni opzionali che supportano diversi casi d'uso, ad esempio l'utilizzo di Amazon S3 con il cluster database per l'importazione o l'esportazione di dati.

- Per importare i dati da un bucket Amazon S3 nel cluster database Babelfish, è necessario configurare l'estensione Aurora PostgreSQL `aws_s3`. Questa estensione consente anche di esportare i dati dal cluster database di Aurora PostgreSQL in un bucket Simple Storage Service (Amazon S3).
- AWS Lambda è un servizio di elaborazione che consente di eseguire codice senza fornire o gestire server. È possibile utilizzare le funzioni Lambda per elaborare le notifiche di eventi da un'istanza database. Per ulteriori informazioni su Lambda, consulta Che cos'è AWS Lambda? nella Guida per gli sviluppatori di AWS Lambda . Per richiamare le funzioni Lambda dal cluster database Babelfish, è necessario configurare l'estensione Aurora PostgreSQL `aws_lambda`.

Per configurare queste estensioni per il cluster Babelfish, devi prima concedere l'autorizzazione all'utente Babelfish interno per caricare le estensioni. Dopo aver concesso l'autorizzazione, è possibile caricare le estensioni Aurora PostgreSQL.

Abilitazione delle estensioni Aurora PostgreSQL nel cluster database Babelfish

Per poter caricare le estensioni `aws_s3` o `aws_lambda`, concedere i privilegi necessari al cluster database Babelfish.

Nella procedura che segue viene utilizzato lo strumento a riga di comando `psql` di PostgreSQL per connettersi al cluster database. Per ulteriori informazioni, consulta Usare psql per connettersi al cluster di database. È possibile utilizzare anche pgAdmin. Per informazioni dettagliate, consultare Uso di pgAdmin per connettersi al cluster di database.

Questa procedura carica `aws_s3` e `aws_lambda`, uno dopo l'altro. Se si desidera utilizzare solo una di queste estensioni, non è necessario caricarle entrambe. L'estensione `aws_commons` è richiesta da ciascuno e viene caricata per impostazione predefinita come mostrato nell'output.

Per configurare il cluster database Babelfish con privilegi per le estensioni Aurora PostgreSQL

1. Connettiti al cluster database Babelfish. Utilizza il nome per l'utente "master" (-U) specificato al momento della creazione del cluster database Babelfish. Il valore predefinito (`postgres`) viene visualizzato negli esempi.

PerLinux, omacOS: Unix

```
psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com \
-U postgres \
-d babelfish_db \
-p 5432
```

Per Windows:

```
psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com ^
-U postgres ^
-d babelfish_db ^
-p 5432
```

Il comando risponde con un prompt per inserire la password per il nome utente (-U).

```
Password:
```

Inserisci la password del nome utente (-U) per il cluster database. Se il comando viene eseguito correttamente, verrà visualizzato un output simile al seguente.

```
psql (13.4)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

2. Concedi i privilegi all'utente Babelfish interno per creare e caricare le estensioni.

```
babelfish_db=> GRANT rds_superuser TO master_dbo;
GRANT ROLE
```

3. Crea e carica l'estensione `aws_s3`. L'estensione `aws_commons` è obbligatoria e viene installata automaticamente quando viene installato `aws_s3`.

```
babelfish_db=> create extension aws_s3 cascade;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

4. Crea e carica l'estensione `aws_lambda`.

```
babelfish_db=> create extension aws_lambda cascade;
CREATE EXTENSION
babelfish_db=>
```

Utilizzo di Babelfish con Simple Storage Service (Amazon S3)

Se non hai già un bucket Simple Storage Service (Amazon S3) da utilizzare con il cluster database Babelfish, puoi creane uno. Fornisci l'accesso per qualsiasi bucket Simple Storage Service (Amazon S3) che desideri utilizzare.

Prima di provare a importare o esportare dati utilizzando un bucket Simple Storage Service (Amazon S3), completa i seguenti passaggi una tantum.

Per configurare l'accesso per la tua istanza database Babelfish al bucket Simple Storage Service (Amazon S3)

1. Se necessario, crea un bucket Simple Storage Service (Amazon S3) per la tua istanza Babelfish. A tale scopo, segui le istruzioni in Crea un bucket nella Guida per l'utente di Amazon Simple Storage Service-S3-user;.
2. Carica i file nel tuo bucket Simple Storage Service (Amazon S3). A tale scopo, segui la procedura riportata in Aggiungere un oggetto a un bucket nella Guida per l'utente di Amazon Simple Storage Service UserGuide-S3-user;.
3. Imposta le autorizzazioni come necessario:

   - Per importare dati da Amazon S3, il cluster database Babelfish richiede l'autorizzazione per accedere al bucket. Ti consigliamo di utilizzare un ruolo AWS Identity and Access Management (IAM) e di associare una policy IAM a quel ruolo per il tuo cluster. A tale scopo, segui la procedura in Utilizzo di un ruolo IAM per accedere a un bucket Amazon S3..
   - Per esportare i dati dal cluster database Babelfish, è necessario concedere al cluster l'accesso al bucket Amazon S3. Come per l'importazione, si consiglia di utilizzare un ruolo e una policy IAM. A tale scopo, segui la procedura in Configurazione dell'accesso a un bucket Amazon S3.

Ora puoi utilizzare Amazon S3 con l'estensione `aws_s3` con il cluster database Babelfish.

Per importare dati da Simple Storage Service (Amazon S3) in Babelfish ed esportare dati Babelfish in Simple Storage Service (Amazon S3)

1. Utilizzo dell'estensione `aws_s3` con il cluster database Babelfish.

   In questa situazione, assicurati di fare riferimento alle tabelle così come sono presenti nel contesto di PostgreSQL. Vale a dire, se vuoi importare in una tabella Babelfish denominata `[database].[schema].[tableA]`, fai riferimento a quella tabella come `database_schema_tableA` nella funzione `aws_s3`:

   - Per un esempio di utilizzo di una funzione `aws_s3` per importare dati, vedi Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL.
   - Per esempi di utilizzo di funzioni `aws_s3` per esportare dati, vedi Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.
2. Assicurati di fare riferimento alle tabelle Babelfish usando la denominazione PostgreSQL quando utilizzi l'estensione `aws_s3` e Simple Storage Service (Amazon S3), come illustrato nella tabella seguente.

Tabella Babelfish	Tabella Aurora PostgreSQL
`database``.`*`schema`*`.`*`table`*	*`database`*`_`*`schema`*`_`*`table`*

Per ulteriori informazioni sull'uso di Amazon S3 con Aurora PostgreSQL, vedi Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL e Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.

Usare Babelfish con AWS Lambda

Dopo aver caricato l'estensione `aws_lambda` nel cluster database Babelfish ma prima di richiamare le funzioni Lambda, puoi fornire l'accesso a Lambda al cluster database seguendo questa procedura.

Per configurare l'accesso per il cluster database Babelfish per il funzionamento con Lambda

Questa procedura utilizza AWS CLI per creare la politica e il ruolo IAM e associarli al cluster Babelfish DB.

1. Crea una policy IAM che consente l'accesso a Lambda dal cluster database Babelfish.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

2. Crea un ruolo IAM che la policy può assumere in fase di esecuzione.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Collegare la policy al ruolo.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Associa il ruolo al tuo cluster database Babelfish

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Dopo aver completato queste attività, è possibile richiamare le funzioni Lambda. Per ulteriori informazioni ed esempi di configurazione AWS Lambda per il cluster Aurora PostgreSQL DB con, consulta. AWS LambdaAWS Lambda

Per richiamare una funzione Lambda dal cluster database Babelfish

AWS Lambda supporta funzioni scritte in Java, Node.js, Python, Ruby e altri linguaggi. Se la funzione restituisce un testo quando viene richiamata, è possibile richiamarla dal cluster database Babelfish. L'esempio seguente è una funzione python segnaposto che restituisce un saluto.

```
lambda_function.py
import json
def lambda_handler(event, context):
    #TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
```

Attualmente, Babelfish non supporta JSON. Se la funzione restituisce JSON, si utilizza un wrapper per gestire il JSON. Ad esempio, diciamo che la funzione `lambda_function.py` mostrata in precedenza è memorizzata in Lambda come `my-function`.

1. Collegati al cluster database Babelfish utilizzando il client `psql` (o il client pgAdmin). Per ulteriori informazioni, consulta Usare psql per connettersi al cluster di database.
2. Crea il wrapper. Questo esempio utilizza il linguaggio procedurale di PostgreSQL per SQL, PL/pgSQL. Per ulteriori informazioni, consulta PL/pgSQL–SQL Procedural Language (PL/pgSQL - Linguaggio procedurale SQL).

   ```
   create or replace function master_dbo.lambda_wrapper()
   returns text
   language plpgsql
   as
   $$
   declare
      r_status_code integer;
      r_payload text;
   begin
      SELECT payload INTO r_payload
        FROM aws_lambda.invoke(  aws_commons.create_lambda_function_arn('my-function',
    'us-east-1')
                               ,'{"body": "Hello from Postgres!"}'::json );
   ```

```
   return r_payload ;
end;
$$;
```

La funzione ora può essere eseguita dalla porta TDS Babelfish (1433) o dalla porta PostgreSQL (5433).

a. Per richiamare (chiamare) questa funzione dalla porta PostgreSQL:

```
SELECT * from aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function', 'us-east-1'), '{"body": "Hello from Postgres!"}'::json );
```

L'output è simile a quello riportato di seguito:

```
status_code |                       payload                        |
 executed_version | log_result
-------------+------------------------------------------------------
+------------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
         |
(1 row)
```

b. Per richiamare (chiamare) questa funzione dalla porta TDS, connettiti alla porta utilizzando il client della riga di comando `sqlcmd` di SQL Server. Per informazioni dettagliate, consultare Utilizzo di un client SQL Server per la connessione al cluster database. Una volta connesso, esegui quanto segue:

```
1> select lambda_wrapper();
2> go
```

Il comando restituisce un output simile al seguente:

```
{"statusCode": 200, "body": "\"Hello from Lambda!\""}
```

Per ulteriori informazioni sull'uso di Lambda con Aurora PostgreSQL, consulta . Per ulteriori informazioni sull'utilizzo delle funzioni Lambda, consulta Nozioni di base su Lambda nella Guida per gli sviluppatori di AWS Lambda .

Utilizzo di pg_stat_statements in Babelfish

Babelfish per Aurora PostgreSQL supporta l'estensione `pg_stat_statements` dalla versione 3.3.0. Per maggiori informazioni, consulta pg_stat_statements.

Per informazioni sulla versione di questa estensione supportata da Aurora PostgreSQL, consulta Extension versions.

Creazione dell'estensione pg_stat_statements

Per attivare `pg_stat_statements`, è necessario attivare il calcolo dell'identificatore di query. Questa operazione viene eseguita automaticamente se `compute_query_id` è impostato su on o `auto` nel gruppo di parametri. Il valore predefinito del parametro `compute_query_id` è `auto`. È inoltre necessario creare questa estensione per attivare questa funzionalità. Utilizza il comando seguente per installare l'estensione dall'endpoint T-SQL:

```
1>EXEC sp_execute_postgresql 'CREATE EXTENSION pg_stat_statements WITH SCHEMA sys';
```

È possibile accedere alle statistiche delle query utilizzando la seguente query:

```
postgres=>select * from pg_stat_statements;
```

> **Note**
>
> Se durante l'installazione non fornisci il nome dello schema per l'estensione, per impostazione predefinita l'estensione verrà creata in uno schema pubblico. Per accedervi, è necessario scrivere il qualificatore dello schema tra parentesi quadre, come mostrato di seguito:

```
postgres=>select * from [public].pg_stat_statements;
```

Puoi anche creare l'estensione dall'endpoint PSQL.

Autorizzazione dell'estensione

Per impostazione predefinita, è possibile visualizzare le statistiche per le query eseguite all'interno del database T-SQL senza bisogno di autorizzazioni.

Per accedere alle statistiche delle query create da altri, è necessario disporre del ruolo PostgreSQL `pg_read_all_stats`. Segui i passaggi indicati di seguito per costruire il comando GRANT pg_read_all_stats.

1. In T-SQL, usa la seguente query che restituisce il nome del ruolo PG interno.

```
SELECT rolname FROM pg_roles WHERE oid = USER_ID();
```

2. Connettiti al database PostgreSQL di Babelfish per Aurora con il privilegio rds_superuser e usa il seguente comando:

```
GRANT pg_read_all_stats TO <rolname_from_above_query>
```

Esempio

Dall'endpoint T-SQL:

```
1>SELECT rolname FROM pg_roles WHERE oid = USER_ID();
2>go
```

```
rolname
-------
master_dbo
(1 rows affected)
```

Dall'endpoint PSQL:

```
babelfish_db=# grant pg_read_all_stats to master_dbo;
```

```
GRANT ROLE
```

Puoi accedere alle statistiche delle query utilizzando la vista pg_stat_statements:

```
1>create table t1(cola int);
2>go
1>insert into t1 values (1),(2),(3);
2>go
```

```
(3 rows affected)
```

```
1>select userid, dbid, queryid, query from pg_stat_statements;
2>go
```

```
userid dbid queryid             query
------ ---- -------             -----
37503 34582 6487973085327558478 select * from t1
37503 34582 6284378402749466286 SET QUOTED_IDENTIFIER OFF
37503 34582 2864302298511657420 insert into t1 values ($1),($2),($3)
10    34582 NULL                <insufficient privilege>
37503 34582 5615368793313871642 SET TEXTSIZE 4096
37503 34582 639400815330803392  create table t1(cola int)
(6 rows affected)
```

Reimpostazione delle statistiche delle query

Puoi usare `pg_stat_statements_reset()` per eseguire il ripristino delle statistiche raccolte finora da pg_stat_statements. Per maggiori informazioni, consulta pg_stat_statements. Attualmente è supportato solo tramite endpoint PSQL. Per connetterti a Babelfish per Aurora PostgreSQL con il privilegio `rds_superuser`, usa il comando seguente:

```
SELECT pg_stat_statements_reset();
```

Limitazioni

- Attualmente, `pg_stat_statements()` non è supportato tramite l'endpoint T-SQL. La vista `pg_stat_statements` è il metodo consigliato per raccogliere le statistiche.

- Alcune query potrebbero essere riscritte dal parser T-SQL implementato dal motore di Aurora PostgreSQL, in tal caso la vista `pg_stat_statements` mostrerà la query riscritta e non la query originale.

  Esempio

  ```
  select next value for [dbo].[newCounter];
  ```

  La query precedente viene riscritta come segue nella vista pg_stat_statements.

  ```
  select nextval($1);
  ```

- In base al flusso di esecuzione delle istruzioni, alcune query potrebbero non essere tracciate da pg_stat_statements e non essere visibili nella vista. Ciò include le seguenti istruzioni: `use dbname`, `goto`, `print`, `raise error`, `set`, `throw`, `declare cursor`.
- Per le istruzioni CREATE LOGIN e ALTER LOGIN, query e queryid non verranno visualizzate. Verrà visualizzato un avviso di privilegi insufficienti.
- La vista `pg_stat_statements` contiene sempre le due voci seguenti, poiché vengono eseguite internamente dal client `sqlcmd`.
  - SET QUOTED_IDENTIFIER OFF
  - SET TEXTSIZE 4096

Usare pgvector in Babelfish

pgvector, un'estensione open source, consente di cercare dati simili direttamente all'interno del database Postgres. Babelfish ora supporta questa estensione a partire dalle versioni 15.6 e 16.2. Per ulteriori informazioni, pgvector Open source Documentation.

Prerequisiti

Per abilitare la funzionalità pgvector, installa l'estensione nello schema sys utilizzando uno dei seguenti metodi:

- Esegui il seguente comando nel client sqlcmd:

  ```
  exec sys.sp_execute_postgresql 'CREATE EXTENSION vector WITH SCHEMA sys';
  ```

- Connettiti `babelfish_db` ed esegui il seguente comando nel client psql:

```
CREATE EXTENSION vector WITH SCHEMA sys;
```

> **Note**
>
> Dopo aver installato l'estensione pgvector, il tipo di dati vettoriali sarà disponibile solo nelle nuove connessioni al database stabilite. Le connessioni esistenti non riconosceranno il nuovo tipo di dati.

Funzionalità supportate

Babelfish estende la funzionalità T-SQL per supportare quanto segue:

- Archiviazione

  Babelfish ora supporta la sintassi compatibile con i tipi di dati vettoriali, migliorando la compatibilità T-SQL. Per ulteriori informazioni sull'archiviazione dei dati con pgvector, vedi Archiviazione.
- Interrogazione

  Babelfish espande il supporto delle espressioni T-SQL per includere operatori di somiglianza vettoriale. Tuttavia, per tutte le altre interrogazioni, è ancora richiesta la sintassi T-SQL standard.

  > **Note**
  >
  > T-SQL non supporta il tipo Array e i driver del database non dispongono di alcuna interfaccia per gestirli. Come soluzione alternativa, Babelfish utilizza stringhe di testo (varchar/nvarchar) per memorizzare dati vettoriali. Ad esempio, quando richiedi un valore vettoriale [1,2,3], Babelfish restituirà una stringa '[1,2,3]' come risposta. Puoi analizzare e dividere questa stringa a livello di applicazione secondo le tue esigenze.

  Per ulteriori informazioni sull'interrogazione dei dati con pgvector, vedi Interrogazione.
- Indicizzazione

  T-SQL ora supporta la sintassi. `Create Index USING INDEX_METHOD` È ora possibile definire l'operatore di ricerca per similarità da utilizzare su una colonna specifica durante la creazione di un indice.

La grammatica viene inoltre estesa per supportare le operazioni di similarità vettoriale sulla colonna richiesta (Check column_name_list_with_order_for_vector grammar).

```
CREATE [UNIQUE] [clustered] [COLUMNSTORE] INDEX <index_name> ON <table_name> [USING
 vector_index_method] (<column_name_list_with_order_for_vector>)
Where column_name_list_with_order_for_vector is:
    <column_name> [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS | VECTOR_L2_OPS]
 (COMMA simple_column_name [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS |
 VECTOR_L2_OPS])
```

Per ulteriori informazioni sull'indicizzazione dei dati con pgvector, vedi Indexing.

- Prestazioni
  - Utilizzare per eseguire il debug dei piani di interrogazione dall'`SET BABELFISH_STATISTICS PROFILE ON`endpoint T-SQL.
  - Aumenta `max_parallel_workers_get_gather` utilizzando la `set_config` funzione supportata in T-SQL.
  - Utilizzare `IVFFlat` per ricerche approssimative. Per ulteriori informazioni, consulta IVFFlat.

  Per migliorare le prestazioni con pgvector, vedi Prestazioni.

Limitazioni

- Babelfish non supporta la ricerca a testo completo per la ricerca ibrida. Per ulteriori informazioni, consulta Hybrid Search.
- Babelfish attualmente non supporta la funzionalità di reindicizzazione. Tuttavia, puoi comunque utilizzare l'endpoint PostgreSQL per reindicizzare. Per ulteriori informazioni, consulta Aspirazione.

Utilizzo dell'apprendimento automatico di Amazon Aurora con Babelfish

Puoi estendere le funzionalità del tuo cluster Babelfish for Aurora PostgreSQL DB integrandolo con l'apprendimento automatico di Amazon Aurora. Questa perfetta integrazione ti consente di accedere a una gamma di potenti servizi come Amazon Comprehend o Amazon AI o SageMaker Amazon Bedrock, ciascuno personalizzato per soddisfare esigenze di machine learning distinte.

Come utente Babelfish, puoi utilizzare le conoscenze esistenti della sintassi e della semantica T-SQL quando lavori con l'apprendimento automatico Aurora. Segui le istruzioni fornite nella

AWS documentazione di Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora Postgre SQL.

Prerequisiti

- Prima di provare a configurare il cluster Babelfish for Aurora PostgreSQL DB per utilizzare l'apprendimento automatico di Aurora, è necessario comprendere i relativi requisiti e prerequisiti. Per ulteriori informazioni, consulta Requisiti per l'utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL.
- Assicurati di installare l'`aws_ml`estensione utilizzando l'endpoint Postgres o la procedura store. `sp_execute_postgresql`

```
exec sys.sp_execute_postgresql 'Create Extension aws_ml'
```

> **Note**
>
> Attualmente Babelfish non supporta le operazioni a cascata in Babelfish. `sp_execute_postgresql` Poiché `aws_ml` si basa su`aws_commons`, dovrai installarlo separatamente utilizzando l'endpoint Postgres.

```
create extension aws_common;
```

Gestione della sintassi e della semantica T-SQL con le funzioni **`aws_ml`**

Gli esempi seguenti spiegano come la sintassi e la semantica T-SQL vengono applicate ai servizi Amazon ML:

Example : aws_bedrock.invoke_model — Una semplice query che utilizza le funzioni di Amazon Bedrock

```
aws_bedrock.invoke_model(
   model_id      varchar,
   content_type  text,
   accept_type   text,
   model_input   text)
Returns Varchar(MAX)
```

L'esempio seguente mostra come richiamare un modello Anthropic Claude 2 per Bedrock utilizzando invoke_model.

```
SELECT aws_bedrock.invoke_model (
    'anthropic.claude-v2', -- model_id
    'application/json', -- content_type
    'application/json', -- accept_type
    '{"prompt": "\n\nHuman:
    You are a helpful assistant that answers questions directly
    and only using the information provided in the context below.
    \nDescribe the answerin detail.\n\nContext: %s \n\nQuestion:
    %s \n\nAssistant:","max_tokens_to_sample":4096,"temperature"
    :0.5,"top_k":250,"top_p":0.5,"stop_sequences":[]}' -- model_input
);
```

Example : aws_comprehend.detect_sentiment — Una semplice query che utilizza le funzioni Amazon Comprehend

```
aws_comprehend.detect_sentiment(
   input_text varchar,
   language_code varchar,
   max_rows_per_batch int)
Returns table (sentiment varchar, confidence real)
```

L'esempio seguente mostra come richiamare il servizio Amazon Comprehend.

```
select sentiment from aws_comprehend.detect_sentiment('This is great', 'en');
```

Example : aws_sagemaker.invoke_endpoint — Una semplice query che utilizza le funzioni di Amazon SageMaker

```
aws_sagemaker.invoke_endpoint(
  endpoint_name varchar,
  max_rows_per_batch int,
```

```
  VARIADIC model_input "any") -- Babelfish inherits PG's variadic parameter type
Rerurns Varchar(MAX)
```

Poiché model_input è contrassegnato come VARIADIC e di tipo «any», gli utenti possono passare un elenco di qualsiasi lunghezza e qualsiasi tipo di dati alla funzione che fungerà da input (input) per il modello. L'esempio seguente mostra come richiamare il SageMaker servizio Amazon.

```
SELECT CAST (aws_sagemaker.invoke_endpoint(
    'sagemaker_model_endpoint_name',
    NULL,
    arg1, arg2 -- model inputs are separate arguments )
AS INT) -- cast the output to INT
```

Per informazioni più dettagliate sull'uso del machine learning di Aurora con Aurora PostgreSQL, consulta. Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora Postgre SQL

Limitazioni

- Sebbene Babelfish non consenta la creazione di array, può comunque gestire i dati che rappresentano gli array. Quando si utilizzano funzioni come quelle `aws_bedrock.invoke_model_get_embeddings` che restituiscono gli array, i risultati vengono forniti come una stringa contenente gli elementi dell'array.

Babelfish supporta i server collegati

Babelfish for Aurora Postgre SQL supporta i server collegati utilizzando l'estensione Postgre nella versione 3.1.0. SQL `tds_fdw` Per utilizzare i server collegati, devi installare l'estensione `tds_fdw`. Per ulteriori informazioni sull'estensione `tds_fdw`, consulta Utilizzo dei wrapper di dati esterni supportati per Amazon Aurora Postgre Amazon for SQL .

Installazione dell'estensione **`tds_fdw`**

È possibile installare un'estensione `tds_fdw` utilizzando i seguenti metodi.

Utilizzo dall'endpoint Postgre CREATE EXTENSION SQL

1. Connect alla tua istanza SQL DB Postgre sul database Babelfish nella porta Postgre. SQL Usa un account con il ruolo rds_superuser.

```
psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
username=test --dbname=babelfish_db --password
```

2. Installa l'estensione `tds_fdw`. Questo è un processo di installazione da compiere una sola volta. Non è necessario ripetere l'installazione al riavvio del cluster di database.

```
babelfish_db=> CREATE EXTENSION tds_fdw;
CREATE EXTENSION
```

Chiamata TDS della stored procedure dall'endpoint **`sp_execute_postgresql`**

Babelfish supporta l'installazione dell'estensione `tds_fdw` richiamando la procedure `sp_execute_postgresql` a partire dalla versione 3.3.0. È possibile eseguire SQL istruzioni Postgre dall'SQLendpoint T- senza uscire dalla porta T. SQL Per ulteriori informazioni, consulta Lavorare con Babelfish per le procedure di Aurora Postgre SQL

1. Connect alla propria istanza Postgre SQL DB sul database Babelfish nella porta T. SQL

```
sqlcmd -S your-DB-instance.aws-region.rds.amazonaws.com -U test -P password
```

2. Installa l'estensione `tds_fdw`.

```
1>EXEC sp_execute_postgresql N'CREATE EXTENSION tds_fdw';
2>go
```

Funzionalità supportate

Babelfish supporta l'aggiunta di endpoint RDS remoti for SQL Server o Babelfish for Aurora SQL Postgre come server collegato. Puoi anche aggiungere altre istanze remote del Server come server collegati. SQL Quindi, usa `OPENQUERY( )` per recuperare i dati da questi server collegati. A partire dalla versione 3.2.0 di Babelfish, sono supportati anche i nomi in quattro parti.

Le seguenti stored procedure e visualizzazioni del catalogo sono supportate per utilizzare i server collegati.

Stored procedure

- sp_addlinkedserver: Babelfish non supporta il parametro `@provstr`.
- sp_addlinkedsrvlogin
  - È necessario fornire un nome utente e una password remoti espliciti per connetterti all'origine dati remota. Non puoi connetterti mediante le credenziali automatiche dell'utente. Babelfish supporta solo `@useself = false`.
  - Babelfish non supporta il parametro `@locallogin` poiché non è supportata la configurazione dell'accesso del server remoto specifico per l'accesso locale.
- sp_linkedservers
- sp_helplinkedsrvlogin
- sp_dropserver
- sp_droplinkedsrvlogin: Babelfish non supporta il parametro `@locallogin` poiché non è supportata la configurazione dell'accesso del server remoto specifico per l'accesso locale.
- sp_serveroption: Babelfish supporta le seguenti opzioni del server:
  - timeout delle query (a partire dalla versione 3.2.0)
  - timeout di connessione (a partire dalla versione 3.3.0)
- sp_testlinkedserver (a partire dalla versione 3.3.0 di Babelfish)
- sp_enum_oledb_providers (a partire dalla versione 3.3.0 di Babelfish)

Visualizzazioni del catalogo

- sys.servers
- sys.linked_logins

Utilizzo della crittografia in transito per la connessione

La connessione dal server Babelfish di origine per Aurora Postgre al SQL server remoto di destinazione utilizza la crittografia in transito (TLS/SSL), a seconda della configurazione del database del server remoto. Se il server remoto non è configurato per la crittografia, il server Babelfish che effettua la richiesta al database remoto esegue il fallback su non crittografato.

Per applicare la crittografia della connessione

- Se il server collegato di destinazione è un'istanza for Server, RDS `rds.force_ssl = on` impostalo per l'istanza SQL Server di destinazione. SQL Per ulteriori informazioni suSSL/

TLSconfiguration RDS for SQL Server, vedere Utilizzo SSL con un'istanza DB di Microsoft SQL Server

- Se il server collegato di destinazione è un SQL cluster Babelfish per Aurora Postgre, imposta `babelfishpg_tds.tds_ssl_encrypt = on` e per il server di destinazione. `ssl = on` Per ulteriori informazioni su/, vedere. SSL TLS Impostazioni SSL Babelfish e connessioni client

Aggiungere Babelfish come server collegato da Server SQL

Babelfish for Aurora Postgre SQL può essere aggiunto come server collegato da un Server. SQL Su un database SQL Server, puoi aggiungere Babelfish come server collegato utilizzando il provider Microsoft OLE DB perODBC:. MSDASQL

Esistono due modi per configurare Babelfish come server collegato da SQL Server utilizzando il provider: MSDASQL

- Fornire ODBC la stringa di connessione come stringa del provider.
- Fornisci il sistema DSN di origine ODBC dati durante l'aggiunta del server collegato.

Limitazioni

- OPENQUERY() funziona solo per SELECT e non funziona perDML.
- I nomi degli oggetti in quattro parti funzionano solo per la lettura e non per la modifica della tabella remota. An UPDATE può fare riferimento a una tabella remota nella FROM clausola senza modificarla.
- L'esecuzione di stored procedure sui server collegati a Babelfish non è supportata.
- L'aggiornamento della versione principale di Babelfish potrebbe non funzionare se sono presenti oggetti dipendenti da `OPENQUERY()` o oggetti a cui si fa riferimento tramite nomi in quattro parti. Devi verificare che tutti gli oggetti che fanno riferimento a `OPENQUERY()` o a nomi in quattro parti vengano eliminati prima di un aggiornamento della versione principale.
- I seguenti tipi di dati non funzionano come previsto sul server Babelfish remoto: `nvarchar(max)`, `varchar(max)`, `varbinary(max)`, `binary(max)` e `time`. Si consiglia di utilizzare la CAST funzione per convertirli nei tipi di dati supportati.

Esempio

Nell'esempio seguente, un'istanza di Babelfish for Aurora Postgre si connette a un'SQListanza di RDS for Server nel cloud. SQL

```
EXEC master.dbo.sp_addlinkedserver @server=N'rds_sqlserver', @srvproduct=N'',
 @provider=N'SQLNCLI', @datasrc=N'myserver.CB2XKFSFFMY7.US-WEST-2.RDS.AMAZONAWS.COM';
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'rds_sqlserver',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassw
```

Quando il server collegato è attivo, puoi quindi utilizzare T- SQL OPENQUERY () o la denominazione standard in quattro parti per fare riferimento a una tabella, una vista o altri oggetti supportati sul server remoto:

```
SELECT * FROM OPENQUERY(rds_sqlserver, 'SELECT * FROM TestDB.dbo.t1');
SELECT * FROM rds_sqlserver.TestDB.dbo.t1;
```

Per eliminare il server collegato e tutti gli accessi associati:

```
EXEC master.dbo.sp_dropserver @server=N'rds_sqlserver', @droplogins=N'droplogins';
```

Risoluzione dei problemi

È possibile utilizzare lo stesso gruppo di sicurezza sia per i server di origine sia per quelli remoti per consentire la comunicare tra loro. Il gruppo di sicurezza dovrebbe consentire solo il traffico in entrata sulla TDS porta (1433 per impostazione predefinita) e l'IP di origine nel gruppo di sicurezza può essere impostato come ID del gruppo di sicurezza stesso. Per ulteriori informazioni su come impostare le regole per la connessione a un'istanza da un'altra istanza con lo stesso gruppo di sicurezza, consulta Regole per la connessione alle istanze da un'istanza con lo stesso gruppo di sicurezza.

Se l'accesso non è configurato correttamente, quando si tenta di eseguire una query sul server remoto viene visualizzato un messaggio di errore simile al seguente.

```
TDS client library error: DB #: 20009, DB Msg: Unable to connect: server is unavailable
 or does not exist (mssql2019.aws-region.rds.amazonaws.com), OS #: 110, OS Msg:
 Connection timed out, Level: 9
```

Utilizzo della ricerca nel testo completo in Babelfish

A partire dalla versione 4.0.0, Babelfish fornisce un supporto limitato per Full Text Search (). FTS FTSè una potente funzionalità dei database relazionali che consente la ricerca e l'indicizzazione efficienti di dati ad alto contenuto di testo. Consente di eseguire ricerche testuali complesse e recuperare rapidamente i risultati pertinenti. FTSè particolarmente utile per le applicazioni che gestiscono grandi volumi di dati testuali, come sistemi di gestione dei contenuti, piattaforme di e-commerce e archivi di documenti.

Comprensione delle funzionalità supportate da Babelfish Full Text Search

Babelfish supporta le seguenti funzionalità di ricerca a testo completo:

- CONTAINSClausola:
  - Supporto di base per la CONTAINS clausola.

    ```
    CONTAINS (
         {
            column_name
         }
         , '<contains_search_condition>'
    )
    ```

    > **Note**
    >
    > Attualmente è supportata solo la lingua inglese.

  - Gestione e traduzione complete delle stringhe di `simple_term` ricerca.
- `FULLTEXT INDEX`Clausola:
  - Supporta solo la `CREATE FULLTEXT INDEX ON table_name(column_name [...n]) KEY INDEX index_name` dichiarazione.
  - Supporta la `DROP FULLTEXT INDEX` dichiarazione completa.

> **Note**
>
> Per reindicizzare l'indice di testo completo, è necessario eliminare l'indice di testo completo e crearne uno nuovo nella stessa colonna.

- Caratteri speciali nella condizione di ricerca:
  - Babelfish assicura che le singole occorrenze di caratteri speciali nelle stringhe di ricerca vengano gestite in modo efficace.

    > **Note**
    >
    > Anche se Babelfish ora identifica i caratteri speciali nella stringa di ricerca, è essenziale riconoscere che i risultati ottenuti possono variare rispetto a quelli ottenuti con T-. SQL

- Alias di tabella in column_name:
  - Con il supporto degli alias di tabella, gli utenti possono creare query più concise e leggibili per la ricerca full-text. SQL

Limitazioni di Babelfish Full Text Search

- Attualmente, le seguenti opzioni non sono supportate in Babelfish for Clause. `CONTAINS`
  - I caratteri speciali e le lingue diverse dall'inglese non sono supportati. Riceverai il messaggio di errore generico per i caratteri e la lingua non supportati

    ```
    Full-text search conditions with special characters or languages other than English
     are not currently supported in Babelfish
    ```

  - Colonne multiple come `column_list`
  - PROPERTYattributo
  - `prefix_term`, `generation_term`, `generic_proximity_term`, `custom_proximity_term`, e `weighted_term`
  - Gli operatori booleani non sono supportati e riceverai il seguente messaggio di errore quando vengono utilizzati:

```
boolean operators not supported
```

  - I nomi di identificatori con punti non sono supportati.
- Attualmente, le seguenti opzioni non sono supportate in Babelfish for Clause. `CREATE FULLTEXT INDEX`
  - [nome_tipo_colonna] TYPE COLUMN
  - LANGUAGE[termine_lingua]
  - [STATISTICAL_SEMANTICS]
  - opzioni del filegroup di catalogo
  - con opzioni
- La creazione di un catalogo di testo completo non è supportata. La creazione di un indice di testo completo non richiede un catalogo di testo completo.
- `CREATE FULLTEXT INDEX`non supporta i nomi di identificatori con punti.
- Babelfish attualmente non supporta caratteri speciali consecutivi nelle stringhe di ricerca. Quando viene utilizzato, riceverai il seguente messaggio di errore:

```
Consecutive special characters in the full-text search condition are not currently
 supported in Babelfish
```

Babelfish supporta i tipi di dati geospaziali

A partire dalle versioni 3.5.0 e 4.1.0, Babelfish include il supporto per i seguenti due tipi di dati spaziali:

- Tipo di dati geometrici: questo tipo di dati è destinato alla memorizzazione di dati planari o euclidei (terra piatta).
- Tipo di dati geografici: questo tipo di dati è destinato alla memorizzazione di dati ellissoidali o terrestri circolari, come le coordinate GPS di latitudine e longitudine.

Questi tipi di dati consentono l'archiviazione e la manipolazione di dati spaziali, ma con limitazioni.

Comprensione dei tipi di dati geospaziali in Babelfish

- I tipi di dati geospaziali sono supportati in vari oggetti di database come viste, procedure e tabelle.

- Supporta il tipo di dati puntuale 2D per archiviare i dati sulla posizione come punti definiti da latitudine, longitudine e un SRID (Spatial Reference System Identifier) valido.
- Le applicazioni che si connettono a Babelfish tramite driver come JDBC, ODBC, DOTNET e PYTHON possono utilizzare questa funzionalità geospaziale.

Funzioni relative ai tipi di dati geometrici supportate in Babelfish

- ST GeomFromText (***geometry_tagged_text, SRID) – Crea un'istanza geometrica utilizzando la rappresentazione Wooden Text*** (WKT).
- ST PointFromText (***point_tagged_text***, SRID) — Crea un'istanza puntuale utilizzando la rappresentazione WKT.
- Point (X, Y, SRID) — Crea un'istanza puntuale utilizzando valori float delle coordinate x e y.
- .ST AsText () <geometry_instance>— Estrae la rappresentazione WKT dall'istanza geometrica.
- .stDistance (other_geometry) — Calcola la distanza tra due istanze geometriche<geometry_instance>.
- .STX — Estrae la coordinata X (longitudine<geometry_instance>) per l'istanza geometrica.
- .STY — Estrae la coordinata Y <geometry_instance>(latitudine) per l'istanza geometrica.

Funzioni relative ai tipi di dati geografici supportate in Babelfish

- ST GeomFromText (***geography_tagged_text***, SRID) — Crea un'istanza geografica utilizzando la rappresentazione WKT.
- ST PointFromText (***point_tagged_text, SRID) – Crea un'istanza puntuale utilizzando*** la rappresentazione WKT.
- Point (Lat, Long, SRID) — Crea un'istanza a punti utilizzando i valori float di Latitudine e Longitudine.
- .ST AsText () <geography_instance>— Estrae la rappresentazione WKT dall'istanza geografica.
- .stDistance (other_geography) — Calcola la distanza tra due istanze <geography_instance>geografiche.
- .Lat — Estrae il valore di latitudine <geography_instance>per l'istanza geografica.
- .Long<geography_instance>: estrae il valore di longitudine per l'istanza geografica.

Limitazioni di Babelfish per i tipi di dati geospaziali

- Attualmente, Babelfish non supporta funzionalità più avanzate come i flag Z-M per le istanze puntuali dei tipi di dati geospaziali.
- I tipi di geometria diversi dall'istanza puntuale non sono attualmente supportati:
  - LineString
  - CircularString
  - CompoundCurve
  - Poligono
  - CurvePolygon
  - MultiPoint
  - MultiLineString
  - MultiPolygon
  - GeometryCollection
- Attualmente, l'indicizzazione spaziale non è supportata per i tipi di dati geospaziali.
- Attualmente sono supportate solo le funzioni elencate per questi tipi di dati. Per ulteriori informazioni, consulta Funzioni relative ai tipi di dati geometrici supportate in Babelfish e Funzioni relative ai tipi di dati geografici supportate in Babelfish.
- L'output della funzione stDistance per i dati geografici potrebbe presentare variazioni di precisione minori rispetto a T-SQL. Ciò è dovuto all'implementazione PostGIS sottostante. Per ulteriori informazioni, vedere ST_Distance
- Per prestazioni ottimali, utilizzate i tipi di dati geospaziali incorporati, senza creare ulteriori livelli di astrazione in Babelfish.

> **Tip**
>
> Sebbene sia possibile creare tipi di dati personalizzati, non è consigliabile crearli in aggiunta ai dati geospaziali. Ciò potrebbe introdurre complessità, che potrebbero portare a comportamenti imprevisti a causa del supporto limitato.

- In Babelfish, i nomi delle funzioni geospaziali vengono utilizzati come parole chiave ed eseguiranno operazioni spaziali solo se utilizzate nel modo previsto.

> **Tip**
>
> Quando crei funzioni e procedure definite dall'utente in Babelfish, evita di usare gli stessi nomi delle funzioni geospaziali integrate. Se avete oggetti di database esistenti con gli stessi nomi, usateli per rinominarli. `sp_rename`

Comprendere il partizionamento in Babelfish

A partire dalla versione 4.3.0, Babelfish introduce il partizionamento di tabelle e indici con supporto limitato. Le sezioni seguenti forniscono dettagli sulla creazione di funzioni di partizione, sulla definizione di schemi di partizione e sull'implementazione di tabelle e indici partizionati in Babelfish.

Argomenti

- Introduzione al partizionamento in Babelfish
- Limitazioni e soluzioni alternative

Introduzione al partizionamento in Babelfish

- Funzioni di partizione:
  - `CREATE PARTITION FUNCTION:` Definisce il modo in cui una tabella o un indice vengono partizionati specificando il tipo di colonna di partizionamento e l'intervallo di valori per ciascuna partizione.
  - `DROP PARTITION FUNCTION:` Rimuove una funzione di partizione esistente.
- Schemi di partizione:
  - `CREATE PARTITION SCHEME:` Definisce la mappatura tra partizioni e filegroup.

    > **Note**
    >
    > In Babelfish, i filegroup vengono trattati come oggetti fittizi e non rappresentano posizioni di archiviazione fisiche.

  - `DROP PARTITION SCHEME:` Rimuove uno schema di partizione esistente.
- Funzione del sistema:

  - `$PARTITION`: Questa funzione di sistema restituisce il numero di partizione a cui apparterrebbe un valore specificato in una colonna di partizionamento in una tabella partizionata specificata.
- Tabelle e indici partizionati:
  - `CREATE TABLE ... ON partition_scheme_name (partition_column_name)`: Crea una tabella partizionata basata su uno schema di partizione e una colonna di partizionamento specificati.
  - `CREATE INDEX ... ON partition_scheme_name (partition_column_name)`: Crea un indice partizionato basato su uno schema di partizione e una colonna di partizionamento specificati.
- Visualizzazioni di sistema per il partizionamento dei metadati:

  Le seguenti viste di sistema vengono aggiunte per fornire metadati relativi al partizionamento:
  - `sys.destination_data_spaces`
  - `sys.partitions`
  - `sys.partition_functions`
  - `sys.partition_parameters`
  - `sys.partition_range_values`
  - `sys.partition_schemes`

Limitazioni e soluzioni alternative

Le seguenti funzionalità di partizionamento di SQL Server non sono ancora supportate da Babelfish:

- `ALTER PARTITION FUNCTION` e `ALTER PARTITION SCHEME`.

  > **Note**
  >
  > Babelfish non supporta le operazioni di divisione e unione. Definisci tutte le partizioni nelle funzioni di partizione durante la creazione perché non puoi aggiungere o rimuovere partizioni in un secondo momento.

- Colonne calcolate come colonne di partizionamento.
- `INSERT BULK` e `BCP` utilità per tabelle partizionate.
- `LEFT` opzione di confine per le funzioni di partizione.
- `SQL_VARIANT` tipo di dati per le funzioni di partizione.

- `TRUNCATE TABLE ... WITH PARTITION`.
- `ALTER TABLE ... SWITCH PARTITION`.
- Indici partizionati non allineati come lo schema di partizione e la colonna di partizione che differiscono dalla tabella partizionata.
- La migrazione DMS dal codice sorgente Babelfish è supportata solo per le attività Full Load su tabelle partizionate.
- Utilizzo della collazione nella funzione di partizione.
- Utilizzo di una colonna di partizionamento con regole di confronto diverse da quelle predefinite del database.
- Babelfish non supporta queste opzioni di sintassi ma fornisce soluzioni alternative:
  - Utilizzo dello schema di partizione con vincoli o indici nell'istruzione CREATE TABLE.
  - MODIFICA TABELLA... AGGIUNGI VINCOLO... ON partition_scheme_name (partition_column_name).

> **Note**
>
> È possibile impostare la porta di fuga in modo che venga ignorata. `babelfishpg_tsql.escape_hatch_storage_on_partition` Ciò consentirà al parser di ignorare l'opzione dello schema di partizione utilizzata con vincoli o indici e il backend creerà vincoli o indici individuali per ogni partizione.

Risoluzione dei problemi relativi a Babelfish

Di seguito, puoi trovare idee per la risoluzione dei problemi e soluzioni alternative del cluster database Babelfish.

Argomenti

- Errore della connessione

Errore della connessione

Di seguito, le cause più comuni dei problemi di connessione a un nuovo cluster di database Aurora:

- Il gruppo di sicurezza non consente l'accesso - Se hai problemi a connetterti a un Babelfish, assicurati di aver aggiunto il tuo indirizzo IP al gruppo di sicurezza Amazon EC2 predefinito. È possibile utilizzare https://checkip.amazonaws.com/ per stabilite il tuo indirizzo IP e quindi aggiungerlo alla regola in ingresso per la porta TDS e la porta PostgreSQL. Per ulteriori informazioni, consulta Aggiunta di regole a un gruppo di sicurezza nella Guida per l'utente di Amazon EC2.
- Configurazioni SSL non corrispondenti - Se il parametro `rds.force_ssl` è attivato (impostato su 1) su Aurora PostgreSQL, allora i client devono connettersi a Babelfish tramite SSL. Se il client non è configurato correttamente, visualizzerai un messaggio di errore del tipo seguente:

  ```
  Cannot connect to your-Babelfish-DB-cluster, 1433
  ---------------------
  ADDITIONAL INFORMATION:
  no pg_hba_conf entry for host "256.256.256.256", user "your-user-name",
  "database babelfish_db", SSL off (Microsoft SQL Server, Error: 33557097)
  ...
  ```

  Questo errore indica un possibile problema di configurazione SSL tra il client locale e il cluster di database Babelfish e che il cluster richiede che i client utilizzino SSL (il suo parametro `rds.force_ssl` è impostato su 1). Per ulteriori informazioni sulla configurazione di SSL, consulta Utilizzo di SSL con un'istanza database PostgreSQL nella Guida per l'utente di Amazon RDS.

  Se utilizzi SQL Server Management Studio (SSMS) per connetterti a Babelfish e vedi questo errore, puoi scegliere tra le opzioni di connessione Encrypt connection (Crittografa connessione) e Trust server certificate (Considera attendibile il certificato del server) nel riquadro Connection

Properties (Proprietà connessione) e riprova. Queste impostazioni gestiscono i requisiti di connessione SSL per SSMS.

Per ulteriori informazioni sulla risoluzione dei problemi di connessione al database Aurora, consulta Impossibile connettersi all'istanza database di Amazon RDS.

Disattivazione di Babelfish

Quando non hai più bisogno di Babelfish, puoi disattivare la funzionalità Babelfish.

Ricorda alcune considerazioni:

- In alcuni casi, potresti disattivare Babelfish prima di completare una migrazione ad Aurora PostgreSQL. Se lo fai e il tuo DDL dipende dai tipi di dati di SQL Server o utilizzi qualsiasi funzionalità T-SQL nel codice, il codice non va a buon fine.
- Se dopo aver eseguito il provisioning di un'istanza Babelfish si disattiva l'estensione Babelfish, non è possibile eseguire nuovamente il provisioning dello stesso database sullo stesso cluster.

Per disattivare Babelfish, modifica il gruppo di parametri, impostando `rds.babelfish_status` su `OFF`. È possibile continuare a utilizzare i tipi di dati di SQL Server con Babelfish disattivato, impostando `rds.babelfish_status` su `datatypesonly`.

Se disattivi Babelfish nel gruppo di parametri, tutti i cluster che utilizzano quel gruppo di parametri perdono la funzionalità Babelfish.

Per ulteriori informazioni sulla modifica dei parametri dei gruppi, consulta . Per informazioni sui parametri specifici di Babelfish, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish.

Gestione degli aggiornamenti di versione di Babelfish per Aurora Postgre SQL

Babelfish è un'opzione disponibile con Aurora Postgre versione 13.4 e successive. SQL Gli aggiornamenti a Babelfish diventano disponibili con alcune nuove versioni del motore di database Aurora Postgre. SQL Per ulteriori informazioni, consulta le note di rilascio per Aurora Postgre. SQL

> **Note**
>
> I cluster DB Babelfish in esecuzione su qualsiasi versione di Aurora Postgre 13 non SQL possono essere aggiornati ad Aurora Postgre 14.3, 14.4 e 14.5. SQL Inoltre, Babelfish non supporta l'aggiornamento diretto da 13.x a 15.x. È necessario prima aggiornare il cluster di database 13.x alla versione 14.6 e successiva e quindi alla versione 15.x.

Per un elenco di funzionalità supportate tra versioni di Babelfish differenti, consulta Funzionalità supportate in Babelfish per versione.

Per un elenco di funzionalità attualmente non supportate, consulta Funzionalità non supportate in Babelfish.

Puoi ottenere un elenco delle SQL versioni di Aurora Postgre che supportano Babelfish interrogando il Regione AWS describe-db-engine-versions AWS CLI comando, come segue.

In Linux, macOS, oppure Unix:

```
$ aws rds describe-db-engine-versions --region us-east-1 \
    --engine aurora-postgresql \
    --query '*[]|[?SupportsBabelfish==`true`].[EngineVersion]' \
    --output text
```

Per ulteriori informazioni, consulta describe-db-engine-versionsnel riferimento ai AWS CLI comandi.

Nei seguenti argomenti, puoi imparare come identificare la versione di Babelfish in esecuzione sul tuo cluster Aurora Postgre SQL DB e come eseguire l'aggiornamento a una nuova versione.

Indice

- Identificazione della versione di Babelfish
- Aggiornamento del cluster Babelfish a una nuova versione

- Aggiornamento di Babelfish a una nuova versione secondaria
- Aggiornamento di Babelfish a una nuova versione principale
  - Operazioni preliminari all'aggiornamento di Babelfish a una nuova versione principale
  - Esecuzione dell'aggiornamento alla versione principale
  - Operazioni successive all'aggiornamento a una nuova versione principale
  - Esempio: aggiornamento del cluster di database Babelfish a una versione principale
- Utilizzo del parametro di versione del prodotto Babelfish
  - Configurazione del parametro di versione del prodotto Babelfish
  - Query e parametri interessati
  - Interfaccia con il parametro babelfishpg_tsql.version

Identificazione della versione di Babelfish

Puoi interrogare Babelfish per trovare dettagli sulla versione Babelfish, sulla versione Aurora Postgre e sulla SQL versione compatibile di Microsoft Server. SQL Puoi usare la porta o la porta Postgre. TDS SQL

- To use the TDS port to query for version information
- To use the PostgreSQL port to query for version information

Per utilizzare la TDS porta per richiedere informazioni sulla versione

1. Utilizza `sqlcmd` o `ssms` per connetterti all'endpoint per il cluster database Babelfish.

   ```
   sqlcmd -S bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
       login-id -P password -d db_name
   ```

2. Per identificare la versione Babelfish, esegui la query riportata di seguito:

   ```
   1> SELECT CAST(serverproperty('babelfishversion') AS VARCHAR)
   2> GO
   ```

   La query restituisce risultati simili a quanto esposto di seguito:

   ```
   serverproperty
   ------------------------------
   ```

```
3.4.0

(1 rows affected)
```

3. Per identificare la versione del cluster Aurora Postgre SQL DB, esegui la seguente query:

```
1> SELECT aurora_version() AS aurora_version
2> GO
```

La query restituisce risultati simili a quanto esposto di seguito:

```
aurora_version


-----------------------------------------------
15.5.0

(1 rows affected)
```

4. Per identificare la versione compatibile di Microsoft SQL Server, esegui la seguente query:

```
1> SELECT @@VERSION AS version
2> GO
```

La query restituisce risultati simili a quanto esposto di seguito:

```
Babelfish for Aurora PostgreSQL with SQL Server Compatibility - 12.0.2000.8
Dec 7 2023 09:43:06
Copyright (c) Amazon Web Services
PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)

(1 rows affected)
```

Come esempio che mostra una piccola differenza tra Babelfish e Microsoft SQL Server, puoi eseguire la seguente query. Su Babelfish, la query viene restituita1, mentre su Microsoft SQL Server, la query viene restituita. NULL

```
SELECT CAST(serverproperty('babelfish') AS VARCHAR) AS runs_on_babelfish
```

È inoltre possibile utilizzare la SQL porta Postgre per ottenere informazioni sulla versione, come illustrato nella procedura seguente.

Per utilizzare la SQL porta Postgre per richiedere informazioni sulla versione

1. Utilizza `psql` o `pgAdmin` per connetterti all'endpoint per il cluster database Babelfish.

```
psql host=bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com
     port=5432 dbname=babelfish_db user=sa
```

2. Attiva la funzionalità estesa (`\x`) di `psql` per un output più leggibile.

```
babelfish_db=> \x
babelfish_db=> SELECT
babelfish_db=> aurora_version() AS aurora_version,
babelfish_db=> version() AS postgresql_version,
babelfish_db=> sys.version() AS Babelfish_compatibility,
babelfish_db=> sys.SERVERPROPERTY('BabelfishVersion') AS Babelfish_Version;
```

La query restituisce un output simile al seguente:

```
-[ RECORD 1 ]-----------
+-------------------------------------------------------------------------------------------
aurora_version          | 15.5.0
postgresql_version      | PostgreSQL 15.5 on x86_64-pc-linux-gnu, compiled by
 x86_64-pc-linux-gnu-gcc (GCC) 9.5.0, 64-bit
babelfish_compatibility | Babelfish for Aurora Postgres with SQL Server
 Compatibility - 12.0.2000.8                      +
                         | Dec 7 2023 09:43:06
                                    +
                         | Copyright (c) Amazon Web Services
                                     +
                         | PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)
babelfish_version       | 3.4.0
```

Aggiornamento del cluster Babelfish a una nuova versione

Le nuove versioni di Babelfish sono disponibili con alcune nuove versioni del motore di database Aurora Postgre SQL dopo la versione 13.4. Ogni nuovo rilascio di Babelfish ha un proprio numero

di versione. Come con Aurora PostgreSQL, Babelfish utilizza il. *major* *minor*. *patch*schema di denominazione per le versioni. Ad esempio, la prima versione di Babelfish, la versione 1.0.0 di Babelfish, è diventata disponibile come parte di Aurora Postgre 13.4.0. SQL

Babelfish non richiede un processo di installazione separato e, Come discusso in precedenzaCreazione di un cluster Babelfish per Aurora Postgre DB SQL, Turn on Babelfish è un'opzione che scegli quando crei un cluster Aurora Postgre DB. SQL

Analogamente, non è possibile aggiornare Babelfish in modo indipendente dal cluster di database Aurora che lo supporta. Per aggiornare un cluster Babelfish for Aurora Postgre SQL DB esistente a una nuova versione di Babelfish, aggiorni il cluster Aurora SQL Postgre DB a una nuova versione che supporti la versione di Babelfish che desideri utilizzare. La procedura da seguire per l'aggiornamento dipende dalla versione di Aurora Postgre SQL che supporta la distribuzione di Babelfish, come segue.

Aggiornamenti di una versione principale

È necessario aggiornare le seguenti versioni di Aurora Postgre SQL ad Aurora Postgre SQL 14.6 e versioni successive prima di eseguire l'aggiornamento alla versione Aurora Postgre 15.2. SQL

- Aurora Postgre SQL 13.8 e tutte le versioni successive
- Aurora Postgre SQL 13.7.1 e tutte le versioni secondarie successive
- Aurora Postgre SQL 13.6.4 e tutte le versioni secondarie successive

È possibile aggiornare Aurora Postgre SQL 14.6 e versioni successive ad Aurora Postgre 15.2 e versioni successive. SQL

L'aggiornamento di un cluster Aurora Postgre SQL DB a una nuova versione principale comporta diverse attività preliminari. Per ulteriori informazioni, consulta Esecuzione di un aggiornamento della versione principale. Per aggiornare correttamente il tuo cluster DB Babelfish for Aurora Postgre, devi creare un gruppo di parametri del cluster SQL DB personalizzato per la nuova versione di Aurora Postgre. SQL Questo nuovo gruppo di parametri deve contenere le impostazioni dei parametri Babelfish uguali a quelle del cluster da aggiornare. Per ulteriori informazioni e per esaminare una tabella con gli aggiornamenti delle versioni principali di origine e di destinazione, consulta Aggiornamento di Babelfish a una nuova versione principale.

Aggiornamenti della versione secondaria e patch

Per l'aggiornamento delle versioni secondarie e delle patch non è richiesta la creazione di un nuovo gruppo di parametri cluster di database e si può utilizzare il processo di aggiornamento della versione secondaria, che può essere eseguito automaticamente o manualmente. Per

ulteriori informazioni e per esaminare una tabella con le versioni di origine e di destinazione, consulta Aggiornamento di Babelfish a una nuova versione secondaria.

> **Note**
>
> Prima di eseguire un aggiornamento principale o minore, applica tutte le attività di manutenzione in sospeso al cluster Babelfish for Aurora Postgre. SQL

Argomenti

- Aggiornamento di Babelfish a una nuova versione secondaria
- Aggiornamento di Babelfish a una nuova versione principale

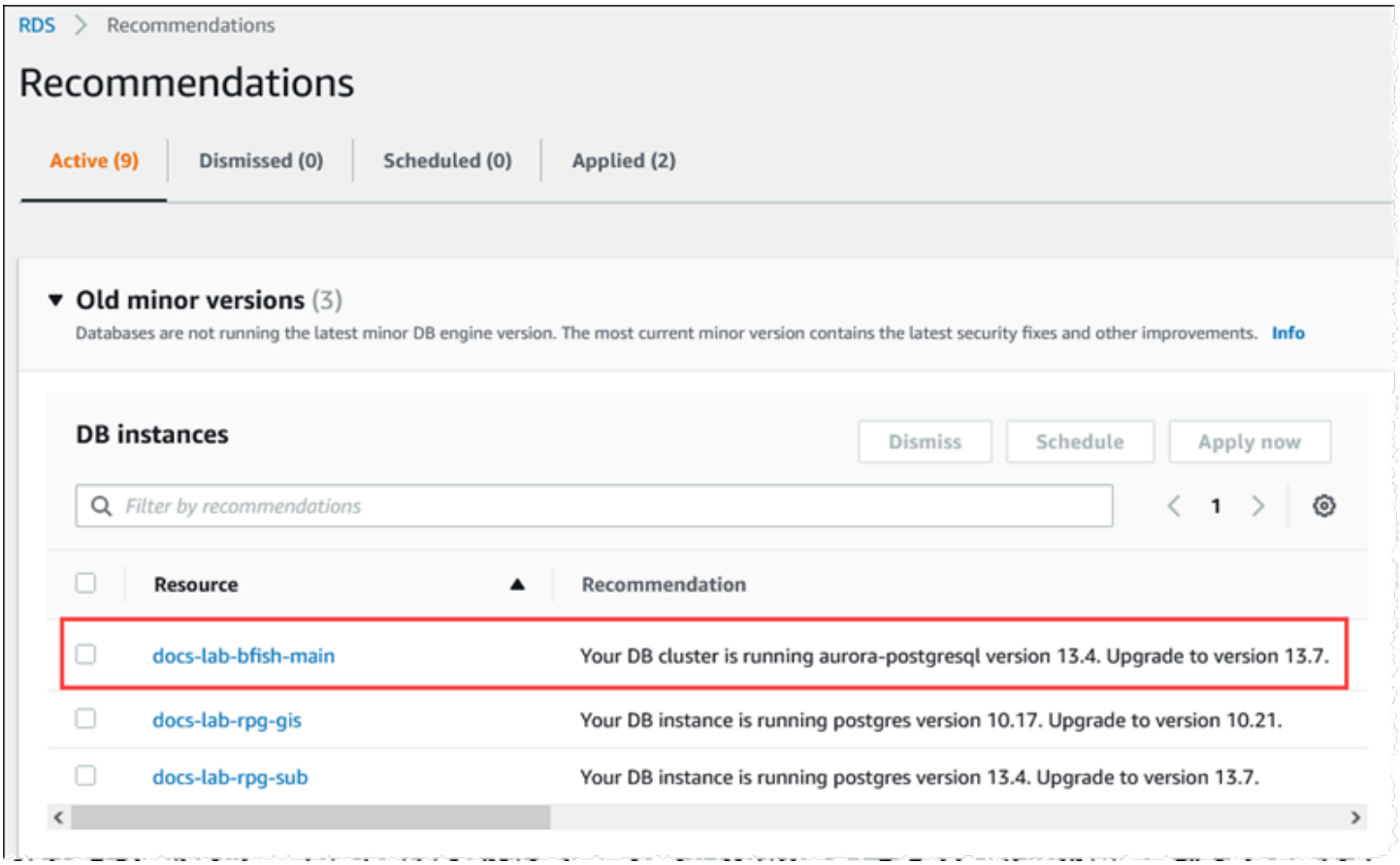
Aggiornamento di Babelfish a una nuova versione secondaria

Una nuova versione secondaria include solo le modifiche compatibili con le versioni precedenti e una patch include importanti correzioni per una versione secondaria rilasciata. Ad esempio, l'etichetta di versione del primo rilascio di Aurora PostgreSQL 13.4 era Aurora PostgreSQL 13.4.0 e successivamente sono state rilasciate diverse patch per la versione secondaria, tra cui Aurora PostgreSQL 13.4.1, 13.4.2 e 13.4.4. Le patch disponibili per ogni versione di Aurora PostgreSQL sono indicate nell'elenco Patch releases (Rilasci di patch), riportato all'inizio delle note di rilascio della versione in questione di Aurora PostgreSQL. Per un esempio, consulta PostgreSQL 14.3 nelle Note di rilascio per Aurora PostgreSQL.

Se il cluster di database Aurora PostgreSQL è configurato con l'opzione Auto minor version upgrade (Aggiornamento automatico versione secondaria), il cluster di database Babelfish per Aurora PostgreSQL viene aggiornato automaticamente durante la finestra di manutenzione del cluster. Per ulteriori informazioni sull'aggiornamento automatico della versione secondaria (AmVU) e su come utilizzarlo, consulta Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora. Se non usi l'aggiornamento automatico della versione secondaria, puoi aggiornare manualmente il cluster database Babelfish per Aurora PostgreSQL a nuove versioni secondarie tramite le attività di manutenzione o la modifica del cluster in modo da utilizzare la nuova versione.

Quando si sceglie una versione di Aurora PostgreSQL da installare e quando si visualizza un cluster Aurora PostgreSQL DB esistente in, la versione visualizza il. AWS Management Console*major minor*solo cifre. Ad esempio, nell'immagine seguente della console puoi vedere che per un cluster di

database Babelfish per Aurora PostgreSQL con Aurora PostgreSQL 13.4 si consiglia l'aggiornamento alla versione 13.7, una nuova versione secondaria di Aurora PostgreSQL.



Per ottenere dettagli completi sulla versione, incluso il *patch* livello, puoi interrogare il cluster Aurora PostgreSQL DB utilizzando la funzione Aurora PostgreSQL. `aurora_version` Per ulteriori informazioni, consulta aurora_version nella Riferimenti relativi alle funzioni Aurora PostgreSQL. Un esempio di utilizzo della funzione è disponibile nella procedura To use the PostgreSQL port to query for version information in Identificazione della versione di Babelfish.

La tabella seguente mostra le versioni di Aurora PostgreSQL e Babelfish di origine e le versioni di destinazione disponibili che supportano il processo di aggiornamento della versione secondaria.

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
16.6 (4.4.0)	16.8 (4.5.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
16.4 (4.3.0)	16.8 (4.5.0), 16.6 (4.4.0)
16.3 (4.2.0)	16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
16.2 (4.1.0)	16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
16.1 (4.0.0)	16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15,10 (3.8.0)	15,12 (3.9.0)
15,8 (3.7.0)	15,12 (3.9.0), 15.10 (3.8.0)
15.7 (3.6.0)	15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
15.6 (3.5.0)	15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
15,5 (3,4,0)	15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
15.4 (3.3.0)	15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
15,12 (3.9.0), 15.3 (3.2.0)	15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
15.2 (3.1.0)	15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14,15 (2.11.0)	14,17 (2.12.0)
14,13 (2.10.0)	14,17 (2.12.0), 14,15 (2.11.0)
14,12 (2.9.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
14.11 (2.8.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0)
14,10 (2.7.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0), 14,11 (2.8.0)
14,9 (2.6.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0), 14,11 (2.8.0), 14,10 (2.7.0)
14.8 (2.5.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0), 14,11 (2.8.0), 14,10 (2.7.0), 14,9 (2.6.0)
14.7 (2.4.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0), 14,11 (2.8.0), 14,10 (2.7.0), 14,9 (2.6.0), 14,8 (2.5.0)
14.6 (2.3.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0), 14,11 (2.8.0), 14,10 (2.7.0), 14,9 (2.6.0), 14,8 (2.5.0), 14,7 (2.4.0)
14,5 (2.2.0)	14,17 (2.12.0), 14,15 (2.11.0), 14,13 (2.10.0), 14,12 (2.9.0), 14,11 (2.8.0), 14,10 (2.7.0), 14,9 (2.6.0), 14,8 (2.5.0), 14.7 (2.4.0), 14.6 (2.3.0)
14.3 (2.1.0)	14.6 (2.3.0)
13.8 (1.4.0)	13.9 (1.5.0)
13.7 (1.3.0)	13.9 (1.5.0), 13.8 (1.4.0)

Aggiornamento di Babelfish a una nuova versione principale

Per eseguire l'aggiornamento della versione principale, è necessario prima aggiornare il cluster di database Babelfish per Aurora PostgreSQL a una versione che supporti l'aggiornamento della versione principale e per farlo, gli aggiornamenti di patch o della versione secondaria devono essere

applicati al cluster di database. Per ulteriori informazioni, consulta Aggiornamento di Babelfish a una nuova versione secondaria.

La tabella seguente mostra le versioni di Aurora PostgreSQL e di Babelfish che possono supportare l'aggiornamento della versione principale.

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
16.8 (4.5.0)	17.4 (5.1.0)
16.6 (4.4.0)	17.4 (5.1.0)
16.4 (4.3.0)	17.4 (5.1.0)
16.3 (4.2.0)	17.4 (5.1.0)
16.2 (4.1.0)	17.4 (5.1.0)
16.1 (4.0.0)	17.4 (5.1.0)
15,12 (3.9.0)	17.4 (5.1.0) 16.8 (4.5.0)
15.10 (3.8.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0)
15,8 (3.7.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
15.7 (3.6.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
15.6 (3.5.0)	17.4 (5.1.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
	16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15,5 (3,4,0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.4 (3.3.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.3 (3.2.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.2 (3.1.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
14,17 (2,12,0)	17,4 (5.1.0) 16.8 (4.5.0) 15,12 (3.9.0)
14,15 (2.11.0)	17,4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0) 15,12 (3.9.0), 15.10 (3.8.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
14,13 (2.10.0)	17,4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
14,12 (2.9.0)	17,4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
14.11 (2.8.0)	17,4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
14,10 (2,7,0)	17,4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
14.9 (2.6.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)

Versioni di origine correnti	Versioni di destinazione dell'aggiornamento più recente
14.8 (2.5.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.7 (2.4.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
14.6 (2.3.0)	17.4 (5.1.0) 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15,12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
13.9 (1.5.0)	14.6 (2.3.0)
13.8 (1.4.0)	14.6 (2.3.0)
13.7 (1.3.0)	14.6 (2.3.0)

Operazioni preliminari all'aggiornamento di Babelfish a una nuova versione principale

Un aggiornamento può comportare brevi interruzioni e per questo motivo si consiglia vivamente di eseguire o pianificare gli aggiornamenti durante la finestra di manutenzione o durante altri periodi di scarso utilizzo del sistema.

Operazioni preliminari all'aggiornamento della versione principale

1. Identifica la versione di Babelfish del cluster di database Aurora PostgreSQL esistente utilizzando i comandi descritti in Identificazione della versione di Babelfish. Le informazioni sulla versione di Aurora PostgreSQL e sulla versione di Babelfish sono gestite da PostgreSQL, quindi segui i passaggi illustrati nella procedura To use the PostgreSQL port to query for version information per i dettagli.
2. Verifica se la versione supporta l'aggiornamento della versione principale. Per l'elenco delle versioni che supportano la funzionalità di aggiornamento della versione principale, consulta Aggiornamento di Babelfish a una nuova versione secondaria ed esegui le attività necessarie prima dell'aggiornamento.

   Se, ad esempio, la versione di Babelfish è in esecuzione su un cluster di database Aurora PostgreSQL 13.5 e desideri eseguire l'aggiornamento ad Aurora PostgreSQL 15.2, devi prima applicare tutte le patch e le versioni secondarie per aggiornare il cluster ad Aurora PostgreSQL 14.6 o versione successiva. Una volta che il cluster è aggiornato alla versione 14.6 o successiva, puoi continuare il processo di aggiornamento della versione principale.
3. Crea uno snapshot manuale del cluster di database Babelfish corrente come backup, che ti consente di ripristinare il cluster alla versione di Aurora PostgreSQL e di Babelfish precedente l'aggiornamento, compresi tutti i dati. Per ulteriori informazioni, consulta Creazione di uno snapshot del cluster database. Assicurati di mantenere il gruppo di parametri cluster di database personalizzato per riutilizzarlo se decidi di ripristinare il cluster allo stato precedente l'aggiornamento. Per ulteriori informazioni, consultare Ripristino da uno snapshot cluster database e Considerazioni sui gruppi di parametri.
4. Prepara un gruppo di parametri cluster di database personalizzato per la versione del database Aurora PostgreSQL di destinazione. Duplica le impostazioni dei parametri Babelfish dal cluster di database Babelfish per Aurora PostgreSQL corrente. Per l'elenco di tutti i parametri Babelfish, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish. Per l'aggiornamento della versione principale, è necessario che i seguenti parametri abbiano le stesse impostazioni del cluster di database di origine. Tutte le impostazioni devono essere uguali per poter completare l'aggiornamento.

- rds.babelfish_status
- babelfishpg_tds.tds_default_numeric_precision
- babelfishpg_tds.tds_default_numeric_scale
- babelfishpg_tsql.database_name
- babelfishpg_tsql.default_locale
- babelfishpg_tsql.migration_mode
- babelfishpg_tsql.server_collation_name

> **Warning**
>
> Se nel gruppo di parametri cluster di database personalizzato per la nuova versione di Aurora PostgreSQL le impostazioni dei parametri Babelfish non corrispondono ai valori dei parametri del cluster che stai aggiornando, l'operazione `ModifyDBCluster` non riesce e Viene visualizzato un messaggio di `InvalidParameterCombination` errore nel AWS Management Console o nell'output del `modify-db-cluster` AWS CLI comando.

5. Utilizzate AWS Management Console o AWS CLI per creare il gruppo di parametri del cluster DB personalizzato. Scegli la famiglia di Aurora PostgreSQL applicabile per la versione di Aurora PostgreSQL che desideri per l'aggiornamento.

> **Tip**
>
> I gruppi di parametri vengono gestiti a Regione AWS livello. Quando si lavora con AWS CLI, è possibile configurare con una regione predefinita anziché specificarla `--region` nel comando. Per ulteriori informazioni sull'utilizzo di AWS CLI, consulta Configurazione rapida nella Guida per l'AWS Command Line Interface utente.

Esecuzione dell'aggiornamento alla versione principale

1. Aggiorna il cluster di database Aurora PostgreSQL a una nuova versione principale. Per ulteriori informazioni, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.
2. Riavvia l'istanza di scrittura del cluster per rendere effettive le impostazioni dei parametri.

Operazioni successive all'aggiornamento a una nuova versione principale

Dopo l'aggiornamento della versione principale a una nuova versione di Aurora PostgreSQL, il valore `IDENTITY` nelle tabelle con una colonna `IDENTITY` potrebbe essere maggiore (+32) del valore precedente l'aggiornamento. Di conseguenza, quando la riga successiva viene inserita nelle tabelle, il valore della colonna Identity generata diventa +32 e la sequenza inizia da quel numero. Questa condizione non influisce negativamente sulle funzioni del cluster di database Babelfish. Tuttavia, se lo desideri, puoi reimpostare l'oggetto sequenza in base al valore massimo della colonna e per farlo, ti devi connettere alla porta T-SQL nell'istanza di scrittura Babelfish utilizzando `sqlcmd` o un altro client SQL Server. Per ulteriori informazioni, consulta Utilizzo di un client SQL Server per la connessione al cluster database.

```
sqlcmd -S bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
     sa -P ******** -d dbname
```

Una volta stabilita la connessione, usa il seguente comando SQL per generare le istruzioni che puoi utilizzare per inizializzare l'oggetto sequenza associato. Questo comando SQL funziona per le configurazioni Babelfish con un singolo database e più database. Per ulteriori informazioni su questi due modelli di implementazione, consulta Utilizzo di Babelfish con un singolo database o più database.

```
DECLARE @schema_prefix NVARCHAR(200) = ''
IF current_setting('babelfishpg_tsql.migration_mode') = 'multi-db'
    SET @schema_prefix = db_name() + '_'
SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
 schema_name(tables.schema_id)
    + '.' + tables.name + ''', ''' + columns.name + '''),(select max(' + columns.name +
 ')
    FROM ' + schema_name(tables.schema_id) + '.' + tables.name + '));
    'FROM sys.tables tables JOIN sys.columns
    columns ON tables.object_id = columns.object_id
    WHERE columns.is_identity = 1
GO
```

La query genera una serie di istruzioni SELECT che è possibile eseguire per reimpostare il valore massimo di IDENTITY e risolvere eventuali problemi di sequenza. Di seguito è riportato l'output del database SQL Server di esempio Northwind in esecuzione su un cluster Babelfish.

```
------------------------------------------------------
```

```
SELECT setval(pg_get_serial_sequence('northwind_dbo.categories', 'categoryid'),(select
 max(categoryid)
    FROM dbo.categories));

SELECT setval(pg_get_serial_sequence('northwind_dbo.orders', 'orderid'),(select
 max(orderid)
    FROM dbo.orders));


SELECT setval(pg_get_serial_sequence('northwind_dbo.products', 'productid'),(select
 max(productid)
    FROM dbo.products));


SELECT setval(pg_get_serial_sequence('northwind_dbo.shippers', 'shipperid'),(select
 max(shipperid)
    FROM dbo.shippers));


SELECT setval(pg_get_serial_sequence('northwind_dbo.suppliers', 'supplierid'),(select
 max(supplierid)
    FROM dbo.suppliers));


(5 rows affected)
```

Esegui le istruzioni una alla volta per reimpostare i valori della sequenza.

Esempio: aggiornamento del cluster di database Babelfish a una versione principale

In questo esempio, puoi trovare la serie di AWS CLI comandi che spiega come aggiornare un cluster Aurora PostgreSQL 13.6.4 DB che esegue Babelfish versione 1.2.2 ad Aurora PostgreSQL 14.6. Per prima cosa crea un gruppo di parametri cluster di database personalizzato per Aurora PostgreSQL 14. Quindi, modifica i valori dei parametri in modo che corrispondano a quelli della versione Aurora PostgreSQL 13 di origine. Infine, esegui l'aggiornamento modificando il cluster di origine. Per ulteriori informazioni, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish. In questo argomento, è inoltre possibile trovare informazioni sull'utilizzo di per eseguire l'aggiornamento. AWS Management Console

Usa il comando CLI create-db-cluster-parameter-group per creare il gruppo di parametri del cluster DB per la nuova versione.

PerLinux, omacOS: Unix

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
    --db-parameter-group-family aurora-postgresql14 \
    --description 'New custom parameter group for upgrade to new major version' \
    --region us-west-1
```

Quando si esegue questo comando, il gruppo di parametri cluster di database personalizzato viene creato nella Regione AWS. L'output visualizzato è simile al seguente.

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14",
        "DBParameterGroupFamily": "aurora-postgresql14",
        "Description": "New custom parameter group for upgrade to new major version",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-west-1:111122223333:cluster-
pg:docs-lab-babelfish-apg-14"
    }
}
```

Per ulteriori informazioni, consulta Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.

Usa il comando CLI modify-db-cluster-parameter-group per modificare le impostazioni in modo che corrispondano al cluster di origine.

Per Windows:

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name docs-lab-
babelfish-apg-14 ^
  --parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_precision,ParameterValue=38,ApplyMethod=pen
reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_scale,ParameterValue=8,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.database_name,ParameterValue=babelfish_db,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.default_locale,ParameterValue=en-
US,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.migration_mode,ParameterValue=single-
db,ApplyMethod=pending-reboot" ^
```

```
 "ParameterName=babelfishpg_tsql.server_collation_name,ParameterValue=sql_latin1_general_cp1_ci
reboot"
```

La risposta è simile a quella riportata di seguito.

```
{
    "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14"
}
```

Utilizza il comando modify-db-clusterCLI per modificare il cluster in modo da utilizzare la nuova versione e il nuovo gruppo di parametri del cluster DB personalizzato. Specifica anche l'argomento `--allow-major-version-upgrade`, come indicato nell'esempio seguente.

```
aws rds modify-db-cluster \
--db-cluster-identifier docs-lab-bfish-apg-14 \
--engine-version 14.6 \
--db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
--allow-major-version-upgrade \
--region us-west-1 \
--apply-immediately
```

Utilizzate il comando reboot-db-instanceCLI per riavviare l'istanza writer del cluster, in modo che le impostazioni dei parametri possano avere effetto.

```
aws rds reboot-db-instance \
--db-instance-identifier docs-lab-bfish-apg-14-instance-1\
--region us-west-1
```

Utilizzo del parametro di versione del prodotto Babelfish

Un nuovo parametro Grand Unified Configuration (GUC) chiamato `babelfishpg_tds.product_version` viene introdotto dalle versioni Babelfish 2.4.0 e 3.1.0. Questo parametro consente di impostare il numero di versione del prodotto SQL Server come output di Babelfish.

Il parametro è una stringa ID di versione in 4 parti e ogni parte deve essere separata da ".".

Sintassi

```
Major.Minor.Build.Revision
```

- Versione principale: un numero compreso tra 11 e 16.
- Versione secondaria: un numero compreso tra 0 e 255.
- Versione di build: un numero compreso tra 0 e 65535.
- Revisione: 0 e qualsiasi numero positivo.

Configurazione del parametro di versione del prodotto Babelfish

È necessario utilizzare il gruppo di parametri del cluster per impostare il parametro babelfishpg_tds.product_version nella console. Per ulteriori informazioni su come modificare il parametro del cluster DB, vedere. Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora

Quando si imposta il parametro della versione del prodotto su un valore non valido, la modifica non ha effetto. Sebbene la console possa mostrare il nuovo valore, il parametro mantiene il valore precedente. Controlla il file di log del motore per i dettagli sui messaggi di errore.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name mydbparametergroup \
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

In Windows:

```
aws rds modify-db-cluster-parameter-group ^
--db-cluster-parameter-group-name mydbparametergroup ^
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Query e parametri interessati

Query/Parametro	Risultato	Tempo effettivo
SELECT @@VERSION	Restituisce la versione SQL del server definita dall'utente (babelfishpg_tsql.version value = Valore predefinito)	Subito
SELECT SERVERPROPERTY('ProductVersion')	Restituisce la versione del SQL server definita dall'utente	Subito
SELECT SERVERPROPERTY('ProductMajorVersion')	Restituisce la versione principale della versione del SQL server definita dall'utente	Subito
VERSIONtoken in PRELOGIN Response Message	Il server restituisce PRELOGIN messaggi con una versione del SQL server definita dall'utente	Ha effetto quando un utente crea una nuova sessione
SQLServerVersionin LoginAck quando si utilizza JDBC	DatabaseMetaData. getDatabaseProductVersion () restituisce la versione SQL del server definita dall'utente	Ha effetto quando un utente crea una nuova sessione

Interfaccia con il parametro babelfishpg_tsql.version

È possibile impostare l'output di @@ VERSION utilizzando i parametri babelfishpg_tsql.version e babelfishpg_tds.product_version. Gli esempi seguenti mostrano come si interfacciano questi due parametri.

- Quando il parametro babelfishpg_tsql.version è predefinito e babelfishpg_tds.product_version è 15.0.2000.8.
  - Output di @@version - 15.0.2000.8.
- Quando il parametro babelfishpg_tsql.version è impostato su 13.0.2000.8 e il parametro babelfishpg_tds.product_version è 15.0.2000.8.

- Output di @@version - 13.0.2000.8.

Babelfish per Aurora Postgree SQL

Argomenti

- Funzionalità supportate in Babelfish per versione
- Funzionalità non supportate in Babelfish
- Lavorare con Babelfish per le procedure di Aurora Postgre SQL

Funzionalità supportate in Babelfish per versione

Nelle tabelle seguenti, puoi trovare l'elenco di tutte le funzionalità supportate dalle diverse versioni di Babelfish. Per gli elenchi di funzionalità non supportate, vedere Funzionalità non supportate in Babelfish. Per informazioni sulle varie versioni di Babelfish, consulta Release Notes for Aurora PostgreSQL.

Funzionalità di Aurora e PostgreSQL

Nella tabella seguente, puoi trovare le funzionalità di Aurora e PostgreSQL supportate da diverse versioni di Babelfish.

Funzionalità di Aurora e PostgreSQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
Servizi Aurora ML	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Autenticazione del database con Kerberos utilizzando AWS Directory Service	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
Scarica e ripristina	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
L'estensione pg_stat_statement	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No

Funzionalità di Aurora e PostgreSQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
pgvector	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Applicazione di patch senza tempi di inattività (ZDP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No

Funzionalità o sintassi T-SQL

Nella tabella seguente, puoi trovare la funzionalità o la sintassi T-SQL supportate da diverse versioni di Babelfish.

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
4 parti: riferimenti ai nomi degli oggetti per le istruzioni SELECT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No
Sintassi ALTER AUTHORIZATION per cambiare il proprietario del database	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER DATABASE *<db_name>* MODIFICA NOME = *<new_db_name>*	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER FUNCTION	S	S	N	N	N	N	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER PROCEDURE																											

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
	S	S	S	S	N	N	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ALTER ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
MODIFICA UTENTE... CON LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Parola chiave AS in CREATE FUNCTION	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Clausola AT TIME ZONE	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Istanza Babelfish come server collegato	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	N	No
Operatori di confronto! < e!	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREA INDICE... ON partition_scheme_name (partition_column_name)	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE anziché Triggers (DML) nelle visualizzazioni di SQL Server	S	S	S	S	N	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
CREA UNA FUNZIONE DI PARTIZIONE	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREA UNO SCHEMA DI PARTIZIONE	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CREA TABELLA... ON partition_scheme_name (partition_column_name)	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
CREATE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Crea indici unici	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Esecuzione di procedure tra database	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
riferimenti incrociati al database SELECT, INSERT, UPDATE, DELETE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Parametri digitati con cursore solo per i parametri di input (non di output)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
Migrazione dei dati tramite l'utilità bcp client	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Tipi di dati TIMESTAMP, ROWVERSION (per informazioni sull'utilizzo, consulta Funzionalità con implementazione limitata	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Parola chiave DEFAULT nelle chiamate a procedure e funzioni memorizzate	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DBCC CHECKIDENT	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP DATABASE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
DROP IF EXISTS (per oggetti SCHEMA, DATABASE e USER)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
RILASCIA L'indice SU schema.table	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
RILASCIA INDEX schema.table.index	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
ELIMINA LA FUNZIONE DI PARTIZIONE	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ELIMINA LO SCHEMA DI PARTIZIONE	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
DROP ROLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
ENABLE/DISABLE TRIGGER	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
RICERCA NEL TESTO COMPLETO	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Ricerca nel testo completo con clausola CONTAINS	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Tipi di dati spaziali di geometria e geografia	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
GRANT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CONCEDI l'autorizzazione [,... n] ON SCHEMA	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
GUC babelfishpg_tds.product_version	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
Identificatori con punto iniziale	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
INVECE DEI trigger sulle tabelle	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
INVECE DEI trigger sulle viste	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
KILL	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
$partition.PARTITION_FUNCTION_NAME (partition_column_value)	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
PIVOT (usa la definizione nella visualizzazione nelle versioni 4.4.0 e 3.8.0, espressione di tabella comune nella versione 3.4.0, join nella versione 3.4.0)	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
REVOKE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
REVOCA l'autorizzazione [,... n] ON SCHEMA	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
clausole SELECT... OFFSET... FETCH	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SELEZIONA PER JSON AUTO	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ATTIVA (e DISATTIVA) BABELFISH_SHOWPLAN_ALL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
ATTIVA (DISATTIVA) IL PROFILO BABELFISH_STATISTICS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SET CONTEXT_INFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No
SET LOCK_TIMEOUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SET NO_BROWSE TABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
IMPOSTA il conteggio delle righe	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
SET SHOWPLAN_ALL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET STATISTICS IO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	No
Sintassi SET TRANSACTION ISOLATION LEVEL	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
SSMS: connessione con la finestra di dialogo di connessione di Object Explorer	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SSMS: migrazione dei dati con la procedura guidata Import/Export	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
SSMS: supporto parziale per l'esplora tore di oggetti	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
STDEV	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No

Funzionalità o sintassi T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
STDEVP	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
I trigger con più azioni DML possono fare riferimento a tabelle di transizione	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Suggerimenti T-SQL (metodi di join, utilizzo dell'indice, MAXDOP)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
Sintassi T-SQL tra parentesi quadre con il predicato LIKE	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
Valori stringa non quotati nei richiami di procedure memorizzate e nei valori predefiniti	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
VAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No
VARP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No

Funzioni T-SQL integrate

Nella tabella seguente, puoi trovare le funzioni T-SQL integrate supportate da diverse versioni di Babelfish.

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
APP_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
ATN2	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
CHARINDEX	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CHOOSE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
LUNGHEZZA_DEL COLLO	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
NOME_COL	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
COLUMNS_UPDATED	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
COLUMNPROPERTY (solo) CharMaxLen AllowsNull	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CONCAT_WS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
CONTEXT_INFO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CURSOR_STATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
DATABASE_ PRINCIPAL_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No		
DATEADD	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATEDIFF	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATEDIFF_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
DATEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
DATENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATEPART	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		
DATETIMEF ROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
DATETIME2DA PARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No		

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0	
DATETIMEOFFSETFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No			
DATA TRONC	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	N	N	N	N	N	N	No			
DATE_BUCKET	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	N	N	N	N	N	N	No			
EOMONTH	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	S	S	N	N	N	N	N	N	No			
ESEGUI COME CHIAMANTE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No			
proprietà estesa fn_list	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No			
PER JSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No			
FULLTEXTSERVICEPROPERTY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
HAS_DBACCESS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
HAS_PERMS_BY_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
HOST_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
HOST_ID	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	N	No
IDENTITY	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
IS_MEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IS_ROLEMEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IS_SRVROLEMEMBER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
ISJSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
JSON_MODIFY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
JSON_QUERY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
JSON_VALUE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0	
VALORE SUCCESSIVO PER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
DEFINIZIONE DELL'OGGETTO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
NOME_SCHEMA_OGGETTO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
OPENJSON	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
OPENQUERY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
ORIGINAL_LOGIN	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
PARSENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No		
PATINDEX	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
ROWCOUNT_BIG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No		
SCHEMA_NAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CONTESTO_ SESSIONE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
SESSION_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
SID_BINARY (restituisce sempre NULL)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No		
SMALLDATE TIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	S	N	N	N	N	No		
SQUARE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
STR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
STRING_AGG	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	S	S	S	N	N	N	N	N	N	N	N	No		
STRINGA_DIVISA	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
SUSER_SID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
SUSER_SNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		

Funzioni T-SQL integrate	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0	
SWITCHOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No		
SYSTEM_USER	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No		
TIMEFROMPARTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No			
TODATETIMEOFFSET	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No			
TO_CHAR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No				
TRIGGER_NESTLEVEL (solo senza argomenti)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì				
TRY_CONVERT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No				
TYPE_ID	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	No				
NOME_TIPO	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	No				
UPDATE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì				

cataloghi T-SQL INFORMATION_SCHEMA

Nella tabella seguente, puoi trovare i cataloghi T-SQL INFORMATION_SCHEMA supportati da diverse versioni di Babelfish.

cataloghi T-SQL INFORMATION_SCHEMA	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2		
CHECK_CONSTRAINTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
USO_DOMINIO_COLONNA	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
COLUMNS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
USO_COLONNA_VINCOLO	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
DOMAINS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
USO_COLONNA_CHIAVE	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	No
ROUTINE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
TABLES	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
TABLE_CONSTRAINTS																										

cataloghi T-SQL INFORMATION_SCHEMA	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2							
	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
VISUALIZZAZIONI	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			

variabili @@ definite dal sistema T-SQL

Nella tabella seguente, puoi trovare le variabili @@ definite dal sistema T-SQL supportate da diverse versioni di Babelfish.

Variabili @@ definite dal sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0			
@@CURSOR_ROWS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
@@DATEFIRST	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
@@DBTS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
@@ERROR	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
@ @ERROR =213	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			
@@FETCH_STATUS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì			

Variabili @@ definite dal sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
@@IDENTITY	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@LANGUAGE	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@LOCK_TIMEOUT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@MAX_CONNECTIONS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@MAX_PRECISION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@MICROSOFTVERSION	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@NESTLEVEL	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì		
@@PROCID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@ROWCOUNT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
@@SERVERNAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Variabili @@ definite dal sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
@@SERVICENAME	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@SPID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@@TRANCOUNT	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
@ @VERSION (nota la differenza di formato, come descritto in[Differenze T-SQL in Babelfish]().	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Procedure memorizzate del sistema T-SQL

Nella tabella seguente, puoi trovare le stored procedure del sistema T-SQL supportate da diverse versioni di Babelfish.

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
proprietà estesa sp_add	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
sp_addlinkedserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_addlinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_addrole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
sp_addrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
sp_babelfish_volatility	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sp_column_privileges	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_columns_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_columns_managed	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_cursor	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_cursor_list	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sp_cursorclose	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_cursorexecute	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorfetch	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursoropen	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursoroption	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorprepexec	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_cursorunprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_databases	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_datatype_info	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_datatype_info_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_describe_cursor	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_describe_first_result_set	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_describe_undeclared_parameters	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
proprietà sp_dropextended	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
sp_droplinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_droprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
membro sp_droprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_dropserver	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No
sp_enum_oledb_providers	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
sp_execute	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_execute_postgresql (CREA, MODIFICA, ELIMINA)	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
sp_executesql	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_fkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_getapplock	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_helpdb	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_helpdbruolo fisso	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_helplinkedsrvlogin	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No
sp_helprole	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
membro sp_helprolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_helpsrvrolemember	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_helpuser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_linkedservers	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sp_oledb_ro_usrname	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_pkeys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_prefix	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No	
sp_prepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_procedure_params_100_managed	S	S	S	S	S	N	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	
sp_releaseapplock	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_rename	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sp_renamedb	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_rename (per object_type 'DATABASE')	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sp_reset_connection	S	S	N	N	N	N	S	S	N	N	N	N	N	N	N	S	S	N	N	N	N	N	N	N	N	N	N	N	No
sp_serveroption (opzione connect_timeout)	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	N	No
sp_set_session_context	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	No
sp_special_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_sproc_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_sproc_columns_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_statistics	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_statistics_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Procedure memorizzate del sistema T-SQL	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_stored_procedure	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_table_privileges	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_tablecollations_100	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sp_tables	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
server sp_testlinked	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No
sp_unprepare	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
proprietà estesa sp_update	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No
sp_who	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	N	No
xp_qv	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione di sistema CONNECTIONPROPERTY

Nella tabella seguente, puoi trovare le proprietà T-SQL supportate nelle funzioni del sistema CONNECTIONPROPERTY supportate da diverse versioni di Babelfish.

Proprietà T-SQL supportate nella funzione di sistema CONNECTIONPROPERTY	4.4.4	4.4.3	4.3.3	4.3.2	4.2.2	4.2.1	4.1.0
auth_scheme	S	S	S	S	S	S	Sì
client_net_address	S	S	S	S	S	S	Sì
indirizzo_rete locale	S	S	S	S	S	S	Sì
porta_tcp_locale	S	S	S	S	S	S	Sì
net_transport	S	S	S	S	S	S	Sì
tipo_protocollo	S	S	S	S	S	S	Sì
physical_net_transport	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione di sistema OBJECTPROPERTY

Nella tabella seguente, puoi trovare le proprietà T-SQL supportate nelle funzioni del sistema OBJECTPROPERTY supportate da diverse versioni di Babelfish.

Proprietà T-SQL supportate nella funzione di sistema OBJECTPROPERTY	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
IsInlineFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
IsScalarFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
IsTableFunction	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Proprietà T-SQL supportate nella funzione SERVERPROPERTY

Nella tabella seguente, puoi trovare le proprietà T-SQL supportate nelle funzioni SERVERPROPERTY supportate da diverse versioni di Babelfish.

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Babelfish	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
Collation (Regola di confronto)	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CollationID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
Edizione	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
EditionID	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
EngineEdition	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
InstanceName	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No
IsAdvancedAnalyticsInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsBigDataCluster	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsFullTextInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
IsIntegratedSecurityOnly	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
IsLocalDB	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
IsPolyBaseInstalled	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
IsSingleUser	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
È XTPSupported	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
Japanese_CI_AI	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
Japanese_CI_AS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
Japanese_CS_AS	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
LicenseType	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
MachineName	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	No	

Proprietà T-SQL supportate nella funzione SERVERPROPERTY	4.4.4.4	4.4.4.3	4.3.3.3	4.3.3.2	4.2.2.2	4.2.2.1.0
ProductLevel	S	S	S	S	NO	No
ProductMajorVersion	S	S	S	S	SÌ	Sì
ProductMinorVersion	S	S	S	S	SÌ	Sì
ProductUpdateLevel	S	S	S	S	NO	No
ProductVersion	S	S	S	S	SÌ	Sì
ServerName	S	S	S	S	SÌ	Sì

Visualizzazioni di SQL Server supportate da Babelfish

Nella tabella seguente, puoi trovare le viste di SQL Server supportate da diverse versioni di Babelfish.

Visualizzazioni di SQL Server supportate da Babelfish	4.4.4.4	4.4.4.3	4.3.3.3	4.3.3.2	4.2.2.2	4.2.2.1.0
information_schema.key_column_usage	S	S	S	N	NO	No

Visualizzazioni di SQL Server supportate da Babelfish	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
information_schema.routine	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
information_schema.schemata	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
information_schema.sequenze	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No	
sys.all_columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.all_objects	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.all_parameters	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No	
sys.all_sql_modules	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.all_views	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.columns	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Visualizzazioni di SQL Server supportate da Babelfish	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.configurations	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.data_spaces	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.database_files	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.database_mirroring	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.database_principals	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.database_role_members	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.databases	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.destination_data_spaces	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No	
sys.dm_exec_connections	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	

Visualizzazioni di SQL Server supportate da Babelfish	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.dm_exec_session	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.dm_hard_database_replica_states	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.dm_os_host_info	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.dm_os_sys_info	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.endpoint	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.extended_properties	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	S	S	S	S	S	S	N	N	N	N	N	No
sys.indexes	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.partitions	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_functions	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No

Visualizzazioni di SQL Server supportate da Babelfish	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.partition_parameters	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_range_values	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.partition_schemi_	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.schemas	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.server_principals	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.server_role_members	S	S	S	S	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	No
sys.sql_modules	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.sysconfigures	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì
sys.syscurconfigs	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì

Visualizzazioni di SQL Server supportate da Babelfish	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.syslogins	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No	
sys.sysprocesses	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
utenti sys.sys	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	S	S	S	S	S	S	N	N	N	N	N	No	
sys.table_types	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.tables	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys.types	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	N	No		
sys.xml_schema_collections	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sys languages	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	Sì	
sysobjects.crdate	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	N	No		

Funzionalità non supportate in Babelfish

Negli elenchi seguenti è possibile trovare funzionalità attualmente non supportate in Babelfish. Gli aggiornamenti di Babelfish sono inclusi nelle versioni di Aurora PostgreSQL. Per ulteriori informazioni, consulta Release notes for Amazon Aurora PostgreSQL-Compatible Edition.

Argomenti

- Funzionalità attualmente non supportata
- Impostazioni non supportate
- Comandi non supportati
- Nomi di colonna o attributi non supportati
- Tipi di dati non supportati
- Tipi di oggetti non supportati
- Funzioni non supportate
- Sintassi non supportata

Funzionalità attualmente non supportata

Nella tabella è possibile trovare informazioni su determinate funzionalità attualmente non supportate.

Funzionalità o sintassi	Descrizione
Moduli di assembly e routine SQL Common Language Runtime (CLR)	Le funzionalità relative ai moduli di assemblaggio e alle routine CLR non sono supportate.
Attributi cookie	ROWGUIDCOL, SPARSE, FILESTREAM e MASKED non sono supportati.
Database contenuti	I database contenuti con login autenticati a livello di database anziché a livello di server non sono supportati.
DDL tra database	L'esecuzione di istruzioni DDL che fanno riferimento o operano su oggetti su più database non è ancora supportata.
Cursori (aggiornabili)	I cursori aggiornabili non sono supportati.

Funzionalità o sintassi	Descrizione
Cursori (globali)	I cursori GLOBAL non sono supportati.
Cursore (comportamenti di recupero)	I seguenti comportamenti di recupero del cursore non sono supportati: FETCH PRIOR, FIRST, LAST, ABSOLUTE e RELATIVE
Parametri di output di tipo cursore	Le variabili e i parametri di tipo cursore non sono supportati per i parametri di output (viene generato un errore).
Opzioni del cursore	SCROLL, KEYSET, DYNAMIC, FAST_FORWARD, SCROLL_LOCKS, OPTIMISTIC, TYPE_WARNING e FOR UPDATE
Crittografia dei dati	La crittografia dei dati non è supportata.
Applicazioni a livello di dati (DAC)	Le operazioni di importazione o esportazione di applicazioni a livello di dati (DAC) con file di pacchetto DAC (.dacpac) o di backup DAC (.bacpac) non sono supportate.
Comandi DBCC	I comandi Microsoft SQL Server Database Console Commands (DBCC) non sono supportati. DBCC CHECKIDENT è supportato in Babelfish 3.4.0 e versioni successive.
DROP IF EXISTS	Questa sintassi non è supportata per gli oggetti USER e SCHEMA. È supportato per gli oggetti TABLE, VIEW, PROCEDURE, FUNCTION e DATABASE.
Crittografia	Le funzioni e le istruzioni integrate non supportano la crittografia.
Connessioni ENCRYPT_CLIENT_CERT	Le connessioni ai certificati client non sono supportate.
Istruzione EXECUTE AS	Questa affermazione non è supportata.
Clausola EXECUTE AS SELF	Questa clausola non è supportata in funzioni, procedure o trigger.

Funzionalità o sintassi	Descrizione
Clausola EXECUTE AS USER	Questa clausola non è supportata in funzioni, procedure o trigger.
Vincoli di chiave esterna che fanno riferimento al nome del database	I vincoli di chiave esterna che fanno riferimento al nome del database non sono supportati.
FORMAT	I tipi definiti dall'utente non sono supportati.
Dichiarazioni di funzione con più di 100 parametri	Le dichiarazioni di funzione che contengono più di 100 parametri non sono supportate.
Chiamate di funzione che includono DEFAULT come valore di parametro	DEFAULT non è un valore di parametro supportato per una chiamata di funzione. DEFAULT come valore di parametro per una chiamata di funzione è supportato da Babelfish 3.4.0 e versioni successive.
Funzioni, definite esternamente	Le funzioni esterne, incluse le funzioni SQL CLR, non sono supportate.
Tabelle temporanee globali (tabelle con nomi che iniziano con ##)	Le tabelle temporanee globali non sono supportate.
Funzionalità grafico	Tutte le funzionalità dei grafici SQL non sono supportate.
Procedure memorizzate generali estese	Le procedure archiviate nel sistema che forniscono un'interfaccia da un'istanza di SQL Server a programmi esterni, per varie attività di manutenzione non sono supportate. Sono incluse `xp_cmdshell` altre procedure memorizzate nel sistema. Per ulteriori informazioni, vedere General Extended stored procedure.
Identificatori (variabili o parametri) con più caratteri @ iniziali	Gli identificatori che iniziano con più di un `@` principale non sono supportati.

Funzionalità o sintassi	Descrizione
Identificatori, nomi di tabelle o colonne che contengono caratteri [@ o]	Nomi di tabelle o colonne che contengono un determinato segno @ o parentesi quadre non sono supportati.
Indici in linea	Gli indici in linea non sono supportati.
Richiamo di una procedura il cui nome è in una variabile	L'utilizzo di una variabile come nome di procedura non è supportato.
Viste materializzate	Le viste materializzate non sono supportate.
Clausola NOT FOR REPLICATION	Questa sintassi è accettata e ignorata.
Funzioni di escape ODBC	Le funzioni di escape ODBC non sono supportate.
Chiamate di procedura che include DEFAULT come valore di parametro	DEFAULT non è un valore di parametro supportato. DEFAULT come valore di parametro per una chiamata di funzione è supportato da Babelfish 3.4.0 e versioni successive.
Dichiarazioni di procedura con più di 100 parametri	Le dichiarazioni con più di 100 parametri non sono supportate.
Procedure definite esternamente	Le procedure definite esternamente, incluse le procedure SQL CLR, non sono supportate.
Controllo delle versioni delle procedure	Il controllo delle versioni delle procedure non è supportato.
Procedure WITH RECOMPILE	WITH RECOMPILE (se usato in combinazione con le istruzioni DECLARE e EXECUTE) non è supportato.

Funzionalità o sintassi	Descrizione
Riferimenti agli oggetti remoti	L'esecuzione di stored procedure sui server collegati a Babelfish non è supportata. I nomi degli oggetti in quattro parti funzionan o solo per la lettura e non per la modifica della tabella remota. UPDATE può fare riferimento a una tabella remota nella clausola FROM senza modificarla. Per ulteriori informazioni, consulta Babelfish supporta i server collegati.
Sicurezza a livello di riga	La sicurezza a livello di riga con CREATE SECURITY POLICY e le funzioni inline con valori di tabella non è supportata.
Funzionalità Service Broker	La funzionalità Service Broker non è supportata.
SESSIONPROPERTY	Proprietà non supportate: ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS _NULL e NUMERIC_ROUNDABORT
IMPOSTA LINGUA	Questa sintassi non è supportata con alcun valore diverso da `english` o `us_english` .
SP_CONFIGURE	Questa procedura memorizzata di sistema non è supportata.
Parola chiave SQL `SPARSE`	La parola chiave SPARSE viene accettata e ignorata.
Sintassi del costruttore del valore della tabella (clausola FROM)	La sintassi non supportata è per una tabella derivata costruita con la clausola FROM.
Tabelle globali	Le tabelle temporali non sono supportate.
Le procedure temporanee non vengono interrate automatic amente	Questa funzionalità non è supportata.
Trigger, definiti esternamente	Questi trigger non sono supportati, incluso SQL Common Language Runtime (CLR).

Funzionalità o sintassi	Descrizione
Senza clausola SCHEMABIN DING	La creazione di una vista senza SCHEMABINDING non è supportata, tuttavia la vista viene creata come se WITH SCHEMABINDING fosse specificato. L'uso di SCHEMABIN DING durante la creazione di funzioni, procedure e trigger viene ignorato senza alcun avviso.

Impostazioni non supportate

Le seguenti impostazioni non sono supportate:

- ATTIVA ANSI_NULL_DFLT_OFF
- DISATTIVA ANSI_NULL_DFLT_ON
- DISATTIVA IMPOSTA ANSI_PADDING
- DISATTIVA IMPOSTA ANSI_WARNINGS
- DISATTIVA ARITHABORT
- ATTIVA ARITHIGNORE
- SET CURSOR_CLOSE_ON_COMMIT SU
- SET NUMERIC_ROUNDABORT ON
- SET PARSEONLY ON (il comando non funziona come previsto)
- SET FMT-ONLY ON (il comando non funziona correttamente. Sopprime solo l'esecuzione delle istruzioni SELECT ma non di altre.)

Comandi non supportati

Alcune funzionalità per i seguenti comandi non sono supportate:

- ADD SIGNATURE
- ALTER DATABASE, ALTER DATABASE SET
- BACKUP/RESTORE DATABASE/LOG
- RIPRISTINO DEI FILE BACPAC e DACPAC
- CREARE, MODIFICARE, ELIMINARE L'AUTORIZZAZIONE. ALTER AUTHORIZATION è supportata per gli oggetti del database.

- CREATE, ALTER, DROP AVAILABILITY GROUP
- CREATE, ALTER, DROP BROKER PRIORITY
- CREATE, ALTER, DROP COLUMN ENCRYPTION KEY
- CREATE, ALTER, DROP DATABASE ENCRYPTION KEY
- CREATE, ALTER, DROP, BACKUP CERTIFICATE
- CREATE AGGREGATE
- CREATE CONTRACT
- CHECKPOINT

Nomi di colonna o attributi non supportati

I seguenti nomi di colonna non sono supportati:

- $IDENTITY
- $ROWGUID
- IDENTITYCOL

Tipi di dati non supportati

I seguenti tipi di dati non sono supportati:

- HIERARCHYID

Tipi di oggetti non supportati

I seguenti tipi di oggetti non sono supportati:

- COLUMN MASTER KEY
- CREATE, ALTER EXTERNAL DATA SOURCE
- CREATE, ALTER, DROP DATABASE AUDIT SPECIFICATION
- CREATE, ALTER, DROP EXTERNAL LIBRARY
- CREATE, ALTER, DROP SERVER AUDIT
- CREATE, ALTER, DROP SERVER AUDIT SPECIFICATION
- CREATE, ALTER, DROP, OPEN/CLOSE SYMMETRIC KEY

- CREATE, DROP DEFAULT
- CREDENTIAL
- CRYPTOGRAPHIC PROVIDER
- DIAGNOSTIC SESSION
- Visualizzazioni indicizzate
- CHIAVE MASTER SERVICE
- SYNONYM

Funzioni non supportate

Le seguenti funzioni integrate non sono supportate:

Funzioni di aggregazione

- APPROX_COUNT_DISTINCT
- CHECKSUM_AGG
- GROUPING_ID
- STRING_AGG utilizzando la clausola WITHIN GROUP

Funzioni di crittografia

- Funzione CERTENCODED
- Funzione CERTID
- Funzione CERTPROPERTY

Funzioni sui metadati

- COLUMNPROPERTY
- TYPEPROPERTY
- Funzione SERVERPROPERTY - Le seguenti proprietà non sono supportate:
  - BuildClrVersion
  - ComparisonStyle
  - ComputerNamePhysicalNetBIOS

- HadrManagerStatus
- InstanceDefaultDataPath
- InstanceDefaultLogPath
- IsClustered
- IsHadrEnabled
- LCID
- NumLicenses
- ProcessID
- ProductBuild
- ProductBuildType
- ProductUpdateReference
- ResourceLastUpdateDateTime
- ResourceVersion
- ServerName
- SqlCharSet
- SqlCharSetName
- SqlSortOrder
- SqlSortOrderName
- FilestreamShareName
- FilestreamConfiguredLevel
- FilestreamEffectiveLevel

Funzioni di sicurezza

- CERTPRIVATEKEY
- LOGINPROPERTY

Istruzioni, operatori, altre funzioni

- Funzione EVENTDATA
- GET_TRANSMISSION_STATUS
- OPENXML

Sintassi non supportata

La seguente sintassi non è supportata:

- ALTER DATABASE
- ALTER DATABASE SCOPED CONFIGURATION
- ALTER DATABASE SCOPED CREDENTIAL
- ALTER DATABASE SET HADR
- ALTER INDEX
- MODIFICA LA FUNZIONE DI PARTIZIONE
- ALTERARE LO SCHEMA DI PARTIZIONE
- ALTER SCHEMA
- ALTER SERVER CONFIGURATION
- Clausola ALTER SERVICE, BACKUP/RESTORE SERVICE MASTER KEY
- ALTER VIEW
- BEGIN CONVERSATION TIMER
- BEGIN DISTRIBUTED TRANSACTION
- BEGIN DIALOG CONVERSATION
- BULK INSERT
- CREATE COLUMNSTORE INDEX
- CREATE EXTERNAL FILE FORMAT
- CREATE EXTERNAL TABLE
- CREATE, ALTER, DROP APPLICATION ROLE
- CREATE, ALTER, DROP ASSEMBLY
- CREATE, ALTER, DROP ASYMMETRIC KEY
- CREATE, ALTER, DROP CREDENTIAL
- CREATE, ALTER, DROP CRYPTOGRAPHIC PROVIDER
- CREATE, ALTER, DROP ENDPOINT
- CREATE, ALTER, DROP EVENT SESSION
- CREATE, ALTER, DROP EXTERNAL LANGUAGE
- CREATE, ALTER, DROP EXTERNAL RESOURCE POOL
- CREATE, ALTER, DROP FULLTEXT CATALOG

- CREATE, ALTER, DROP FULLTEXT INDEX
- CREATE, ALTER, DROP FULLTEXT STOPLIST
- CREATE, ALTER, DROP MESSAGE TYPE
- CREA, MODIFICA, RILASCIA, CHIAVE OPEN/CLOSE, BACKUP/RESTORE PRINCIPALE
- CREATE, ALTER, DROP QUEUE
- CREATE, ALTER, DROP RESOURCE GOVERNOR
- CREATE, ALTER, DROP RESOURCE POOL
- CREATE, ALTER, DROP ROUTE
- CREATE, ALTER, DROP SEARCH PROPERTY LIST
- CREATE, ALTER, DROP SECURITY POLICY
- CREATE, ALTER, DROP SELECTIVE XML INDEX clause
- CREATE, ALTER, DROP SERVICE
- CREATE, ALTER, DROP SPATIAL INDEX
- CREATE, ALTER, DROP TYPE
- CREATE, ALTER, DROP XML INDEX
- CREATE, ALTER, DROP XML SCHEMA COLLECTION
- CREATE/DROP RULE
- CREATE, DROP WORKLOAD CLASSIFIER
- CREATE, ALTER, DROP WORKLOAD GROUP
- ALTER TRIGGER
- CREA TABELLA Clausola GRANT
- CREA TABELLA Clausola IDENTITY
- CREATE USER - Questa sintassi non è supportata. L'affermazione PostgreSQL CREATE USER non crea un utente equivalente alla sintassi di SQL Server CREATE USER. Per ulteriori informazioni, consulta Differenze T-SQL in Babelfish.
- DENY
- END, MOVE CONVERSATION
- EXECUTE with AS LOGIN or AT option
- GET CONVERSATION GROUP
- GROUP BY ALL clause
- GROUP BY CUBE clause

- GROUP BY ROLLUP clause
- INSERT… DEFAULT VALUES
- MERGE
- READTEXT
- REVERT
- UNPIVOT
- SELECT TOP x PERCENT WHERE x <> 100
- SELECT TOP... WITH TIES
- SELECT... FOR BROWSE
- SELECT... FOR XML AUTO
- SELECT... FOR XML EXPLICIT
- SELECT... PER PERCORSO XML
- SEND
- SET DATEFORMAT
- SET DEADLOCK_PRIORITY
- SET FMTONLY
- SET FORCEPLAN
- SET NUMERIC_ROUNDABORT ON
- SET OFFSETS
- SET REMOTE_PROC_TRANSACTIONS
- SET SHOWPLAN_TEXT
- SET SHOWPLAN_XML
- SET STATISTICS
- SET STATISTICS PROFILE
- SET STATISTICS TIME
- SET STATISTICS XML
- SHUTDOWN statement
- UPDATE STATISTICS
- UPDATETEXT
- Using EXECUTE to call a SQL function

- VIEW... CHECK OPTION clause
- VIEW... VIEW_METADATA clause
- WAITFOR DELAY
- WAITFOR TIME
- WAITFOR, RECEIVE
- WITH XMLNAMESPACES construct
- WRITETEXT
- XPATH expressions

Lavorare con Babelfish per le procedure di Aurora Postgre SQL

Panoramica

Puoi utilizzare la seguente procedura per le istanze Amazon RDS DB che eseguono Babelfish for Aurora SQL Postgre per migliorare le prestazioni delle query:

- sp_babelfish_volatility
- sp_execute_postgresql

sp_babelfish_volatility

La volatilità della SQL funzione Postgre aiuta l'ottimizzatore per una migliore esecuzione delle query che, se utilizzata in parti di determinate clausole, ha un impatto significativo sulle prestazioni delle query.

Sintassi

```
sp_babelfish_volatility 'function_name', 'volatility'
```

Argomenti

function_name (opzionale)

Puoi specificare il valore di questo argomento con un nome composto da due parti come `schema_name.function_name` o solo `function_name`. Se si specifica solo `function_name`, il nome dello schema è lo schema predefinito per l'utente corrente.

volatility (opzionale)

I valori di volatilità di Postgre SQL validi sono, o. `stable` `volatile` `immutable` Per ulteriori informazioni, consulta https://www.postgresql.org/docs/current/xfunc-volatility.html

> **Note**
>
> Quando `sp_babelfish_volatility` viene chiamato con `function_name` che ha più definizioni, verrà generato un errore.

Set di risultati

Se i parametri non sono menzionati, il set di risultati viene visualizzato nelle seguenti colonne: `schemaname`, `functionname`, `volatility`.

Note per l'utilizzo

La volatilità SQL della funzione Postgre aiuta l'ottimizzatore a migliorare l'esecuzione delle query che, se utilizzata in alcune parti di determinate clausole, ha un impatto significativo sulle prestazioni delle query.

Esempi

Gli esempi seguenti mostrano come creare funzioni semplici e in seguito spiegano come utilizzare `sp_babelfish_volatility` con metodi diversi.

```
1> create function f1() returns int as begin return 0 end
2> go
```

```
1> create schema test_schema
2> go
```

```
1> create function test_schema.f1() returns int as begin return 0 end
2> go
```

L'esempio seguente mostra la volatilità delle funzioni:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           volatile
test_schema f1           volatile
```

L'esempio seguente mostra come modificare la volatilità delle funzioni:

```
1> exec sp_babelfish_volatility 'f1','stable'
2> go
1> exec sp_babelfish_volatility 'test_schema.f1','immutable'
```

```
2> go
```

Quando si specifica solo function_name, vengono visualizzati il nome dello schema, il nome della funzione e la volatilità di tale funzione. L'esempio seguente mostra la volatilità delle funzioni dopo la modifica dei valori:

```
1> exec sp_babelfish_volatility 'test_schema.f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
test_schema f1           immutable
```

```
1> exec sp_babelfish_volatility 'f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
```

Quando non specifichi alcun argomento, viene visualizzato un elenco di funzioni (nome dello schema, nome della funzione, volatilità delle funzioni) presenti nel database corrente:

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
test_schema f1           immutable
```

sp_execute_postgresql

È possibile eseguire istruzioni SQL Postgre da T-endpoint. SQL Questo semplifica le applicazioni in quanto non è necessario uscire da T- SQL port per eseguire queste istruzioni.

Sintassi

```
sp_execute_postgresql [ @stmt = ] statement
```

Argomenti

Istruzione [@stmt]

Il tipo di dati dell'argomento è varchar. Questo argomento accetta istruzioni in dialetto PG.

> **Note**
>
> È possibile passare come argomento solo un'istruzione in dialetto PG, in caso contrario viene generato l'errore seguente.

```
1>exec sp_execute_postgresql 'create extension pg_stat_statements; drop extension
 pg_stat_statements'
2>go
```

```
Msg 33557097, Level 16, State 1, Server BABELFISH, Line 1
expected 1 statement but got 2 statements after parsing
```

Note per l'utilizzo

CREATE EXTENSION

Crea e carica una nuova estensione nel database attuale.

```
1>EXEC sp_execute_postgresql 'create extension [ IF NOT EXISTS ] <extension name>
 [ WITH ] [SCHEMA schema_name] [VERSION version]';
2>go
```

Il seguente esempio illustra come creare un'estensione:

```
1>EXEC sp_execute_postgresql 'create extension pg_stat_statements with schema sys
 version "1.10"';
2>go
```

Per concedere l'accesso alla funzione Lambda, utilizza il seguente comando:

```
1>select * from pg_stat_statements;
2>go
```

> **Note**
>
> Se il nome dello schema non viene fornito esplicitamente durante la creazione dell'estensione, per impostazione predefinita le estensioni vengono installate nello schema public. È necessario fornire il qualificatore dello schema per accedere agli oggetti di estensione come indicato di seguito:

```
1>select * from [public].pg_stat_statements;
2>go
```

Estensioni supportate

Le seguenti estensioni disponibili con Aurora Postgre SQL funzionano con Babelfish.

- `pg_stat_statements`
- `tds_fdw`
- `fuzzystrmatch`

Limitazioni

- Gli utenti devono avere il ruolo sysadmin su T- SQL e rds_superuser su postgres per installare le estensioni.
- Non è possibile installare estensioni in schemi creati dall'utente e nemmeno negli schemi dbo e guest per i database master, tempdb e msdb.
- CASCADEl'opzione non è supportata.

ALTER EXTENSION

È possibile eseguire l'aggiornamento a una nuova versione dell'estensione utilizzando ALTER l'estensione.

```
1>EXEC sp_execute_postgresql 'alter extension <extension name> UPDATE TO
 <new_version>';
2>go
```

Limitazioni

- È possibile aggiornare la versione dell'estensione solo utilizzando l'istruzione ALTER Extension. Altre operazioni non sono supportate.

DROP EXTENSION

Rimuove l'estensione specificata. Per rimuovere l'estensione, è anche possibile usare le nostre opzioni `if exists` o `restrict`.

```
1>EXEC sp_execute_postgresql 'drop extension <extension name>';
2>go
```

Limitazioni

- CASCADEl'opzione non è supportata.

Prestazioni e scalabilità per Amazon Aurora PostgreSQL

La sezione seguente descrive come gestire le prestazioni e il dimensionamento per un cluster di database di Amazon Aurora PostgreSQL. Include anche informazioni su altre attività di manutenzione.

Argomenti

- Dimensionamento delle istanze database Aurora PostgreSQL
- Numero massimo di connessioni a un'istanza database Aurora PostgreSQL
- Limiti di storage temporaneo per Aurora PostgreSQL
- Huge Pages per Aurora PostgreSQL

- Test di Amazon Aurora PostgreSQL mediante query Fault Injection
- Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL
- Specificare il disco RAM per stats_temp_directory
- Gestione dei file temporanei con PostgreSQL

Dimensionamento delle istanze database Aurora PostgreSQL

È possibile dimensionare le istanze database Aurora PostgreSQL in due diversi modi, ovvero tramite il dimensionamento delle istanze e il dimensionamento della lettura. Per ulteriori informazioni sul dimensionamento della lettura, consulta Dimensionamento della lettura.

Puoi dimensionare il cluster di database Aurora PostgreSQL in base alle necessità, modificando la classe di istanza database per ogni istanza database nel cluster di database. Aurora PostgreSQL supporta diverse classi di istanza database ottimizzate per Aurora. Non utilizzare classi di istanza db.t2 o db.t3 per cluster Aurora più grandi di dimensioni maggiori di 40 terabyte (TB).

> **Note**
>
> Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni sulle classi di istanza T, consulta Tipi di classi di istanza database.

Il dimensionamento non è immediato. Potrebbero essere necessari 15 minuti o più per completare la modifica a una classe di istanza database diversa. Se si utilizza questo approccio per modificare la classe di istanza database, la modifica viene applicata durante la prossima finestra di manutenzione pianificata (anziché immediatamente) per evitare di influenzare gli utenti.

In alternativa alla modifica diretta della classe di istanza database, è possibile ridurre al minimo i tempi di inattività utilizzando le funzionalità ad alta disponibilità di Amazon Aurora. Innanzitutto, aggiungi una replica Aurora al cluster. Quando crei la replica, scegli la dimensione della classe di istanza database da utilizzare per il cluster. Quando la replica Aurora è sincronizzata con il cluster, viene quindi eseguito il failover sulla replica appena aggiunta. Per ulteriori informazioni, consultare Repliche di Aurora e Failover rapido con Amazon Aurora PostgreSQL.

Per le specifiche dettagliate delle classi di istanza database supportate da Aurora PostgreSQL, consulta Motori DB supportati per classi di istanza database.

Numero massimo di connessioni a un'istanza database Aurora PostgreSQL

Un cluster di database Aurora PostgreSQL alloca le risorse in base alla classe dell'istanza database e alla memoria disponibile. Ogni connessione al cluster di database consuma quantità incrementali di queste risorse, come memoria e CPU. La memoria consumata per connessione varia in base al tipo di query, al conteggio e all'utilizzo delle tabelle temporanee. Anche una connessione inattiva consuma memoria e CPU. Questo perché quando le query vengono eseguite su una connessione, viene allocata più memoria per ogni query che non viene rilasciata completamente, anche quando l'elaborazione si arresta. Pertanto, ti consigliamo di assicurarti che le tue applicazioni non utilizzino connessioni inattive: ognuna di queste spreca risorse e influisce negativamente sulle prestazioni. Per ulteriori informazioni, consulta Resources consumed by idle PostgreSQL connections.

Il numero massimo di connessioni consentite a un'istanza database di Aurora PostgreSQL è determinato dal valore del parametro `max_connections` specificato nel gruppo di parametri per quell'istanza database. L'impostazione ideale per il `max_connections` parametro è quella che supporta tutte le connessioni client necessarie all'applicazione, senza un eccesso di connessioni inutilizzate, oltre ad almeno altre 3 connessioni per supportare l'automazione. AWS Prima di modificare l'impostazione del parametro `max_connections`, è opportuno considerare quanto segue:

- Se il valore `max_connections` è troppo basso, l'istanza database di Aurora PostgreSQL potrebbe non avere connessioni sufficienti quando i client tentano di connettersi. Se ciò accade, tenta di connettersi utilizzando `psql` che genera messaggi di errore come il seguente:

  ```
  psql: FATAL: remaining connection slots are reserved for non-replication superuser
   connections
  ```

- Se il valore `max_connections` supera il numero di connessioni effettivamente necessarie, le connessioni inutilizzate possono causare un peggioramento delle prestazioni.

Il valore predefinito di `max_connections` è derivato dalla seguente funzione Aurora PostgreSQL `LEAST`:

`LEAST({DBInstanceClassMemory/9531392},5000)`.

Se desideri modificare il valore di `max_connections`, devi creare un gruppo di parametri cluster di database personalizzato e modificare il valore. Dopo aver applicato il gruppo di parametri database personalizzato al cluster, assicurati di riavviare l'istanza primaria in modo che il nuovo valore diventi effettivo. Per ulteriori informazioni, consultare Amazon Aurora PostgreSQL parametri e Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.

> **Tip**
>
> Se le applicazioni aprono e chiudono frequentemente connessioni o mantengono un numero elevato di connessioni di lunga durata aperte, ti consigliamo di utilizzare Server proxy per Amazon RDS. Proxy RDS è un proxy di database completamente gestito e ad alta disponibilità che utilizza il pooling di connessioni per condividere connessioni di database in modo sicuro ed efficiente. Per ulteriori informazioni sul Proxy RDS, consulta Proxy Amazon RDS per Aurora.

Per informazioni dettagliate su come Aurora Serverless v2 le istanze gestiscono questo parametro, vediNumero massimo connessioni per Aurora Serverless v2.

Limiti di storage temporaneo per Aurora PostgreSQL

Aurora PostgreSQL memorizza tabelle e indici nel sottosistema di archiviazione Aurora. Aurora PostgreSQL utilizza l'archiviazione temporanea separata per i file temporanei non persistenti. Sono inclusi i file utilizzati per scopi quali l'ordinamento di set di dati di grandi dimensioni durante l'elaborazione delle query o per le operazioni di creazione dell'indice. Per ulteriori informazioni, consulta l'articolo Come posso risolvere i problemi di archiviazione locale nelle istanze compatibili con Aurora PostgreSQL?.

Questi volumi di archiviazione locale sono supportati da Amazon Elastic Block Store e possono essere estesi utilizzando una classe di istanza database più grande. Per ulteriori informazioni sullo storage, consultare Archiviazione Amazon Aurora. È inoltre possibile aumentare l'archiviazione locale per gli oggetti temporanei utilizzando un tipo di istanza NVMe abilitato e oggetti temporanei abilitati per Aurora Optimized Reads-enabled. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.

> **Note**
>
> Durante il dimensionamento delle istanze database, ad esempio da db.r5.2xlarge a db.r5.4xlarge, potrebbero essere visualizzati eventi `storage-optimization`.

La tabella seguente mostra la quantità massima di storage temporaneo disponibile per ogni classe di istanza database Aurora PostgreSQL. Per ulteriori informazioni sul supporto della classe di istanza database per Aurora, consultare Classi di istanze DB Amazon Aurora.

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.x2g.16xlarge	1829
db.x2g.12xlarge	1606
db.x2g.8xlarge	1071
db.x2g.4xlarge	535
db.x2g.2xlarge	268
db.x2g.xlarge	134
db.x2g.large	67
db.r8 g. 48 x grande	3072
db.r8g.24xlarge	1536
db.r8g.16xlarge	998
db.r8 g. 12 x grande	749
db.r8g. 8x grande	499
db.r8g. 4xlarge	250
db.r8g.2xlarge	125
db.r8g.xlarge	63
db.r8g.large	31
db.r7g.16xlarge	1008
db.r7g.12xlarge	756
db.r7g.8xlarge	504
db.r7g.4xlarge	252

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.r7g.2xlarge	126
db.r7g.xlarge	63
db.r7g.large	32
db.r7i.48xlarge	3072
db.r7i.24xlarge	1500
db.r7i.16xlarge	1008
db.r7i.12xlarge	748
db.r7i.8xlarge	504
db.r7i.4xlarge	249
db.r7i.2xlarge	124
db.r7i.xlarge	62
db.r7i.large	31
db.r6g.16xlarge	1008
db.r6g.12xlarge	756
db.r6g.8xlarge	504
db.r6g.4xlarge	252
db.r6g.2xlarge	126
db.r6g.xlarge	63
db.r6g.large	32
db.r6i.32xlarge	1829

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.r6i.24xlarge	1500
db.r6i.16xlarge	1008
db.r6i.12xlarge	748
db.r6i.8xlarge	504
db.r6i.4xlarge	249
db.r6i.2xlarge	124
db.r6i.xlarge	62
db.r6i.large	31
db.r5.24xlarge	1500
db.r5.16xlarge	1008
db.r5.12xlarge	748
db.r5.8xlarge	504
db.r5.4xlarge	249
db.r5.2xlarge	124
db.r5.xlarge	62
db.r5.large	31
db.r4.16xlarge	960
db.r4.8xlarge	480
db.r4.4xlarge	240
db.r4.2xlarge	120

DB instance class (Classe istanza database)	Storage temporaneo massimo disponibile (GiB)
db.r4.xlarge	60
db.r4.large	30
db.t4g.large	16,5
db.t4g.medium	8,13
db.t3.large	16
db.t3.medium	7,5

> **Note**
>
> NVMe i tipi di istanza abilitati possono aumentare lo spazio temporaneo disponibile fino alla NVMe dimensione totale. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.

È possibile monitorare lo spazio di archiviazione temporaneo disponibile per un'istanza DB con la `FreeLocalStorage` CloudWatch metrica --> descritta in. CloudWatch Parametri Amazon per Amazon Aurora (Questo non si applica a Aurora Serverless v2.)

Per alcuni carichi di lavoro, è possibile ridurre la quantità di storage temporaneo allocando più memoria ai processi che stanno perfezionando l'operazione. Per aumentare la memoria disponibile per un'operazione, aumentando i valori dei parametri work_mem o maintenance _work_mem PostgreSQL.

Huge Pages per Aurora PostgreSQL

Le pagine di grandi dimensioni sono una caratteristica di gestione della memoria che riduce il sovraccarico quando un'istanza database lavora con grandi blocchi di memoria contigui, come quelli utilizzati dai buffer condivisi. Questa caratteristica di PostgreSQL è supportata da tutte le versioni Aurora PostgreSQL attualmente disponibili.

Il parametro `Huge_pages` è attivato per impostazione predefinita per tutte le classi di istanza database diverse dalle classi di istanza t3.medium, db.t3.large, db.t4g.medium, db.t4g.large Non è

possibile modificare il valore del parametro `huge_pages` o disattivare questa funzionalità nelle classi di istanza supportate di Aurora PostgreSQL.

Nelle istanze DB Aurora PostgreSQL che non supportano la funzionalità di memoria per pagine enormi, l'utilizzo specifico della memoria di processo potrebbe aumentare senza le corrispondenti modifiche del carico di lavoro.

Il sistema alloca segmenti di memoria condivisa come la buffer cache durante l'avvio del server. Quando non sono disponibili enormi pagine di memoria, il sistema non addebita queste allocazioni al processo postmaster. Al contrario, include la memoria nel processo che ha effettuato per la prima volta l'accesso a ogni pagina da 4 KB nel segmento di memoria condivisa.

> **Note**
>
> Le connessioni attive condividono la memoria allocata in base alle esigenze, indipendentemente dal modo in cui l'utilizzo della memoria condivisa viene monitorato tra i processi.

Test di Amazon Aurora PostgreSQL mediante query Fault Injection

Puoi testare la tolleranza ai guasti del tuo cluster di database Aurora PostgreSQL utilizzando query fault injection. Le query di errore di iniezione vengono emesse come comandi SQL per un'istanza Amazon Aurora. Le query fault injection consentono di arrestare l'istanza in maniera anomala per poter testare il failover e il ripristino. È anche possibile simulare errore della replica di Aurora Replica, errore del disco e congestione del disco. Le query di fault injection sono supportate da tutte le versioni di Aurora PostgreSQL disponibili, come riportato di seguito.

- Aurora PostgreSQL versioni 12, 13, 14 e successive
- Aurora PostgreSQL versione 11.7 e versioni successive
- Aurora PostgreSQL versione 10.11 e versioni successive

Argomenti

- Test dell'arresto anomalo di un'istanza
- Test di un errore di replica Aurora
- Test di un errore del disco
- Test di una congestione del disco

Quando una query fault injection specifica un arresto anomalo, impone un arresto anomalo dell'istanza database di Aurora PostgreSQL. Le altre query fault injection generano simulazioni di eventi di errore, ma non causano l'evento. Quando invii una query fault injection, specifichi anche la durata della simulazione dell'evento di errore.

Puoi inviare una query fault injection a una delle istanze di replica Aurora eseguendo la connessione all'endpoint per la replica Aurora. Per ulteriori informazioni, consulta Connessioni endpoint Amazon Aurora.

Test dell'arresto anomalo di un'istanza

Puoi forzare un arresto anomalo di un'istanza Aurora PostgreSQL utilizzando la funzione query fault injection `aurora_inject_crash()`.

Per questa query fault injection, non si verifica alcun failover. Se desideri testare un failover, puoi scegliere l'azione dell'istanza di failover per il tuo cluster DB nella console RDS o utilizzare il failover-db-cluster AWS CLI comando o l'operazione API Failover DBCluster RDS.

Sintassi

```
SELECT aurora_inject_crash ('instance' | 'dispatcher' | 'node');
```

Opzioni

Questa query fault injection accetta uno dei seguenti tipi di arresto anomalo. Il tipo di arresto anomalo non rileva la distinzione tra maiuscole e minuscole:

'instance'

Viene simulato un arresto anomalo del database compatibile con PostgreSQL per l'istanza Amazon Aurora.

'dispatcher'.

Viene simulato un arresto anomalo del dispatcher sull'istanza primaria per il cluster di database Aurora. Il dispatcher scrive gli aggiornamenti sul volume del cluster per un cluster di database Amazon Aurora.

'node'

Viene simulato un arresto anomalo del database compatibile con PostgreSQL e del dispatcher per l'istanza Amazon Aurora.

Test di un errore di replica Aurora

Puoi simulare l'errore di una replica Aurora mediante la funzione query fault injection `aurora_inject_replica_failure()`.

Un errore di replica Aurora blocca la replica nella replica Aurora o in tutte le repliche Aurora nel cluster di database in base alla percentuale specificata per l'intervallo di tempo specificato. Al termine di tale periodo, le repliche Aurora interessate sono automaticamente sincronizzate con l'istanza primaria.

Sintassi

```
SELECT aurora_inject_replica_failure(
   percentage_of_failure,
   time_interval,
   'replica_name'
);
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

percentage_of_failure

La percentuale di repliche da bloccare durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, non viene bloccata alcuna replica. Se specifichi 100, vengono bloccate tutte le repliche.

time_interval

L'intervallo di tempo per simulare l'errore di replica Aurora. L'intervallo è in secondi. Ad esempio, se il valore è 20, la simulazione viene eseguita per 20 secondi.

> **Note**
>
> Fai attenzione quando specifichi l'intervallo di tempo per l'evento di errore della replica Aurora. Se specifichi un intervallo troppo lungo e l'istanza di scrittura scrive una grande quantità di dati durante l'evento di errore, il cluster database Aurora potrebbe ritenere che si è verificato un arresto anomalo della replica Aurora e quindi sostituirla.

replica_name

La replica Aurora in cui inserire la simulazione dell'errore. Specifica il nome della replica Aurora per simulare un errore di una singola replica Aurora. Specifica una stringa vuota per simulare errori per tutte le repliche Aurora nel cluster database.

Per identificare i nomi delle repliche, vedere la colonna `server_id` della funzione `aurora_replica_status()`. Ad esempio:

```
postgres=> SELECT server_id FROM aurora_replica_status();
```

Test di un errore del disco

Puoi simulare un errore del disco per un cluster database Aurora PostgreSQL mediante la funzione query fault injection `aurora_inject_disk_failure()`.

Durante la simulazione di un errore del disco, il cluster di database Aurora PostgreSQL contrassegna in modo aleatorio i segmenti del disco come difettosi. Le richieste a tali segmenti sono bloccate per la durata della simulazione.

Sintassi

```
SELECT aurora_inject_disk_failure(
   percentage_of_failure,
   index,
   is_disk,
   time_interval
);
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

percentage_of_failure

La percentuale del disco da contrassegnare come difettosa durante l'evento di errore. Può essere un valore tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come difettosa. Se specifichi 100, tutto il disco è contrassegnato come difettoso.

index

Un blocco di dati logico specifico nel quale simulare l'evento di errore. Se viene superato l'intervallo di blocchi logici o di dati dei nodi di storage disponibili, viene generato un errore che indica il valore di indice massimo che puoi specificare. Per evitare questo errore, consulta Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL.

is_disk

Indica se l'errore di inserimento è in un blocco logico o in un nodo di storage. Se si specifica true, significa che gli errori di inserimento sono in un blocco logico. Se si specifica false, significa che gli errori di inserimento sono in un nodo di storage.

time_interval

La quantità di tempo necessaria per simulare l'errore del disco. L'intervallo è in secondi. Ad esempio, se il valore è 20, la simulazione viene eseguita per 20 secondi.

Test di una congestione del disco

È possibile simulare una congestione del disco per un cluster Aurora PostgreSQL DB utilizzando la funzione di query di fault injection. `aurora_inject_disk_congestion()`

Durante la simulazione della congestione del disco, il cluster di database Aurora PostgreSQL contrassegna in modo aleatorio i segmenti del disco come congestionati. Le richieste a tali segmenti vengono ritardate di un periodo di tempo compreso tra il ritardo minimo e quello massimo specificati per la durata della simulazione.

Sintassi

```
SELECT aurora_inject_disk_congestion(
   percentage_of_failure,
   index,
   is_disk,
   time_interval,
   minimum,
   maximum
);
```

Opzioni

Questa query fault injection accetta i seguenti parametri:

percentage_of_failure

La percentuale del disco da contrassegnare come congestionata durante l'evento di errore. Questo è un doppio valore compreso tra 0 e 100. Se specifichi 0, nessuna parte del disco è contrassegnata come congestionata. Se specifichi 100, tutto il disco è contrassegnato come congestionato.

index

Un blocco di dati logico o nodo storage specifico nel quale simulare l'evento di errore.

Se viene superato l'intervallo di blocchi logici o di dati dei nodi di storage disponibili, viene generato un errore che indica il valore di indice massimo che puoi specificare. Per evitare questo errore, consulta Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL.

is_disk

Indica se l'errore di inserimento è in un blocco logico o in un nodo di storage. Se si specifica true, significa che gli errori di inserimento sono in un blocco logico. Se si specifica false, significa che gli errori di inserimento sono in un nodo di storage.

time_interval

La quantità di tempo necessaria per simulare la congestione del disco. L'intervallo è in secondi. Ad esempio, se il valore è 20, la simulazione viene eseguita per 20 secondi.

minimum, maximum

Il valore minimo e massimo del ritardo di congestione in millisecondi. I valori validi sono compresi tra 0,0 e 100,0 millisecondi. I segmenti del disco contrassegnati come congestionati vengono ritardati per un periodo di tempo aleatorio compreso nell'intervallo del valore minimo e massimo per la durata della simulazione. Il valore massimo deve essere maggiore del valore minimo.

Visualizzazione dello stato del volume per un cluster di database Aurora PostgreSQL

In Amazon Aurora, un volume di cluster di database è costituito da una raccolta di blocchi logici. Ognuno di questi rappresenta 10 gigabyte di storage allocato. Questi blocchi sono denominati gruppi di protezione.

I dati in ogni gruppo di protezione sono replicati su sei dispositivi di storage fisico denominati nodi di storage. Questi nodi di storage sono allocati su tre zone di disponibilità (AZs) nella regione in cui risiede il cluster DB. Ogni nodo di storage contiene a sua volta uno o più blocchi di dati logici per il volume di cluster di database. Per ulteriori informazioni sui gruppi di protezione e sui nodi di archiviazione, vedere Introduzione al motore di storage Aurora sul blog del AWS database. Per ulteriori informazioni sui volumi del cluster Aurora, consulta Archiviazione Amazon Aurora.

Utilizza la funzione `aurora_show_volume_status()` per restituire le seguenti variabili di stato del server:

- `Disks` — Il numero totale di blocchi logici di dati per il volume del cluster di database.
- `Nodes` — Il numero totale di nodi di storage per il volume del cluster di database.

Puoi utilizzare la funzione `aurora_show_volume_status()` per evitare errori durante l'utilizzo la funzione fault injection `aurora_inject_disk_failure()`. La funzione fault injection `aurora_inject_disk_failure()` simula l'errore di un intero nodo di storage o di un singolo blocco logico di dati all'interno di un nodo di storage. Nella funzione, si specifica il valore di indice di un blocco logico di dati o nodo di storage specifico. Tuttavia, l'istruzione restituisce un errore se si specifica un valore di indice superiore al numero di blocchi di dati logici o nodi di storage utilizzati dal volume del cluster di database. Per ulteriori informazioni sulle query fault injection, consulta Test di Amazon Aurora PostgreSQL mediante query Fault Injection.

> **Note**
>
> La funzione `aurora_show_volume_status()` è disponibile per Aurora PostgreSQL versione 10.11. Per ulteriori informazioni sulle versioni di Aurora PostgreSQL, consulta Versioni e versioni del motore di Amazon Aurora Postgre SQL.

Sintassi

```
SELECT * FROM aurora_show_volume_status();
```

Esempio

```
customer_database=> SELECT * FROM aurora_show_volume_status();
 disks | nodes
```

```
-------+-------
    96 |    45
```

Specificare il disco RAM per stats_temp_directory

Puoi utilizzare il parametro di Aurora PostgreSQL, `rds.pg_stat_ramdisk_size`, per specificare la memoria di sistema allocata a un disco RAM per l'archiviazione di `stats_temp_directory` PostgreSQL. Il parametro del disco RAM è disponibile in Aurora PostgreSQL 14 e versioni precedenti.

Per alcuni carichi di lavoro, l'impostazione di questo parametro può migliorare le prestazioni e ridurre i requisiti IO. Per ulteriori informazioni sul parametro `stats_temp_directory`, consulta Run-time Statistics nella documentazione di PostgreSQL. A partire dalla versione 15 di PostgreSQL, la community PostgreSQL ha iniziato a utilizzare la memoria condivisa dinamica. Quindi, non è necessario impostare `stats_temp_directory`.

Per abilitare un disco RAM per `stats_temp_directory`, imposta il parametro `rds.pg_stat_ramdisk_size` su un valore diverso da zero nel gruppo di parametri del cluster di database utilizzato dal tuo cluster di database. Questo parametro utilizza MB, quindi è necessario specificare un valore intero. Espressioni, formule e funzioni non sono valide per il parametro `rds.pg_stat_ramdisk_size`. Assicurati di riavviare il cluster di database in modo che la modifica abbia effetto. Per informazioni sull'estensione dei parametri consulta . Per ulteriori informazioni sul riavvio del cluster database, consulta Riavvio di un cluster Amazon Aurora DB o di un'istanza Amazon Aurora DB.

Ad esempio, il AWS CLI comando seguente imposta il parametro del disco RAM su 256 MB.

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name db-cl-pg-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

Dopo il riavvio di cluster di database, esegui il seguente comando per visualizzare lo stato di `stats_temp_directory`:

```
postgres=> SHOW stats_temp_directory;
```

Il comando deve restituire i seguenti valori:

```
stats_temp_directory
---------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

Gestione dei file temporanei con PostgreSQL

In PostgreSQL, una query complessa può eseguire diverse operazioni di ordinamento o hash contemporaneamente, con ogni operazione che utilizza la memoria dell'istanza per archiviare i risultati fino al valore specificato nel parametro. `work_mem` Quando la memoria dell'istanza non è sufficiente, vengono creati file temporanei per archiviare i risultati. Questi vengono scritti su disco per completare l'esecuzione della query. Successivamente, questi file vengono rimossi automaticamente al completamento della query. In In Aurora PostgreSQL, questi file condividono l'archiviazione locale con altri file di log. Puoi monitorare lo spazio di archiviazione locale del cluster Aurora PostgreSQL DB osservando la metrica Amazon per. CloudWatch `FreeLocalStorage` Per ulteriori informazioni, consulta Risoluzione dei problemi relativi all'archiviazione locale.

Consigliamo di utilizzare i cluster di istanze Aurora Optimized per carichi di lavoro che coinvolgono più query simultanee che aumentano l'utilizzo di file temporanei. Questi cluster di utilizzano storage locale a livello di blocco basato su unità a stato solido (SSDNVMe) basate su unità a stato solido (SSD) non volatile Memory Express () per collocare i file temporanei. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.

È possibile utilizzare i seguenti parametri e funzioni per gestire i file temporanei nell'istanza.

- **`temp_file_limit`**: questo parametro annulla qualsiasi query che superi la dimensione definita in KB dal parametro temp_files. Questo limite impedisce a qualsiasi query di essere eseguita all'infinito e di consumare spazio su disco con file temporanei. È possibile stimare il valore utilizzando i risultati del parametro `log_temp_files`. È consigliabile esaminare il comportamento del carico di lavoro e impostare il limite in base alla stima. Gli esempi seguenti mostrano come viene annullata una query quando supera il limite.

  ```
  postgres=>select * from pgbench_accounts, pg_class, big_table;
  ```

  ```
  ERROR: temporary file size exceeds temp_file_limit (64kB)
  ```

- **log_temp_files**: questo parametro invia messaggi a postgresql.log quando i file temporanei di una sessione vengono rimossi. Questo parametro produce log dopo che una query è stata completata correttamente. Pertanto, potrebbe non essere utile nella risoluzione dei problemi delle query attive e con tempi di esecuzione lunghi.

  L'esempio seguente mostra che quando la query viene completata correttamente, le voci vengono registrate nel file postgresql.log mentre i file temporanei vengono eliminati.

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir**: questa funzione disponibile in RDS per PostgreSQL 13 e versioni successive fornisce visibilità sull'attuale utilizzo dei file temporanei. La query completata non viene visualizzata nei risultati della funzione. Nell'esempio seguente, è possibile visualizzare i risultati di questa funzione.

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;

query
------------------------------------------------------------------------------------------
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

Il nome del file include l'ID di elaborazione (PID) della sessione che ha generato il file temporaneo. Una query più avanzata, come nell'esempio seguente, esegue la somma dei file temporanei per ogni PID.

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements**: se attivi il parametro pg_stat_statements, puoi visualizzare l'utilizzo medio dei file temporanei per chiamata. È possibile identificare il valore query_id della query e utilizzarlo per esaminare l'utilizzo dei file temporanei, come illustrato nell'esempio seguente.

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
----------------------+---------------------------+-------+-------------------------
+---------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                  239226 |
                     388678
(1 row)
```

- **Performance Insights**: nel pannello di controllo di Approfondimenti sulle prestazioni, puoi visualizzare l'utilizzo dei file temporanei attivando le metriche temp_bytes e temp_files. Puoi quindi vedere la media di entrambe queste metriche e verificare se corrispondono al carico di lavoro delle query. La visualizzazione all'interno di Approfondimenti sulle prestazioni non evidenzia in modo specifico le query che generano file temporanei. Tuttavia, combinando le informazioni di Approfondimenti sulle prestazioni con la query mostrata per il parametro `pg_ls_tmpdir`, è possibile definire, analizzare e risolvere eventuali problemi a livello di modifiche del carico di lavoro delle query.

  Per ulteriori informazioni su come analizzare metriche e query con Performance Insights, consulta. Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights

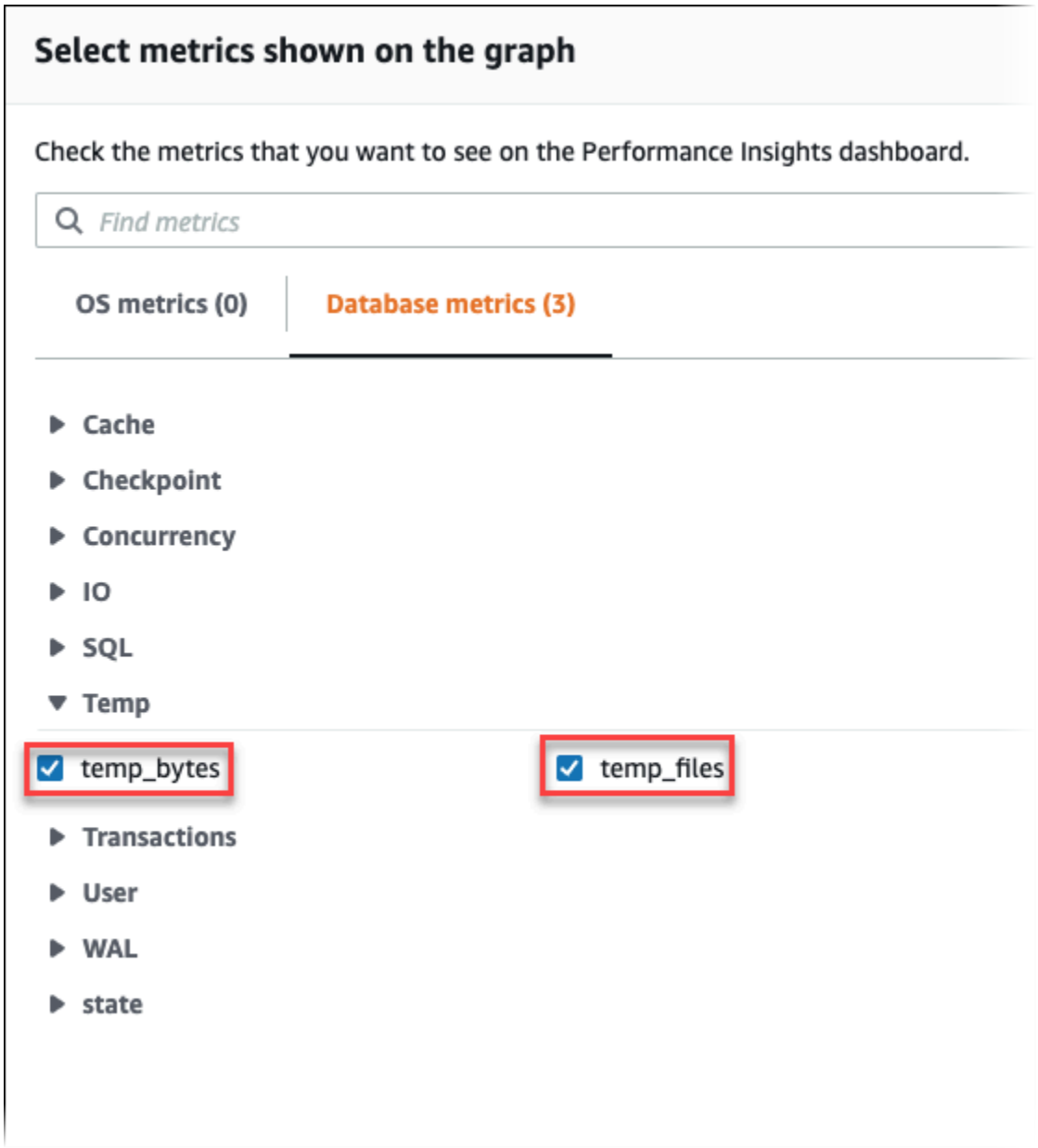
  Per un esempio di visualizzazione dell'utilizzo di file temporanei con Performance Insights, vedere Visualizzazione dell'utilizzo temporaneo dei file con Performance Insights

Visualizzazione dell'utilizzo temporaneo dei file con Performance Insights

Puoi utilizzare Performance Insights per visualizzare l'utilizzo dei file temporanei attivando le metriche temp_bytes e temp_files. La visualizzazione in Performance Insights non mostra le query specifiche che generano file temporanei, tuttavia, quando si combina Performance Insights con la query mostrata per`pg_ls_tmpdir`, è possibile risolvere i problemi, analizzare e determinare le modifiche nel carico di lavoro delle query.

1. Nel pannello di controllo di Approfondimenti sulle prestazioni, scegli Gestisci parametri.

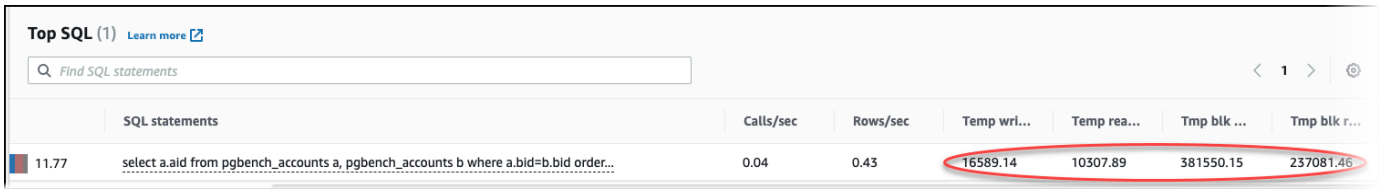
2. Seleziona Metriche del database e quindi seleziona le metriche temp_bytes e temp_files come illustrato nell'immagine seguente.



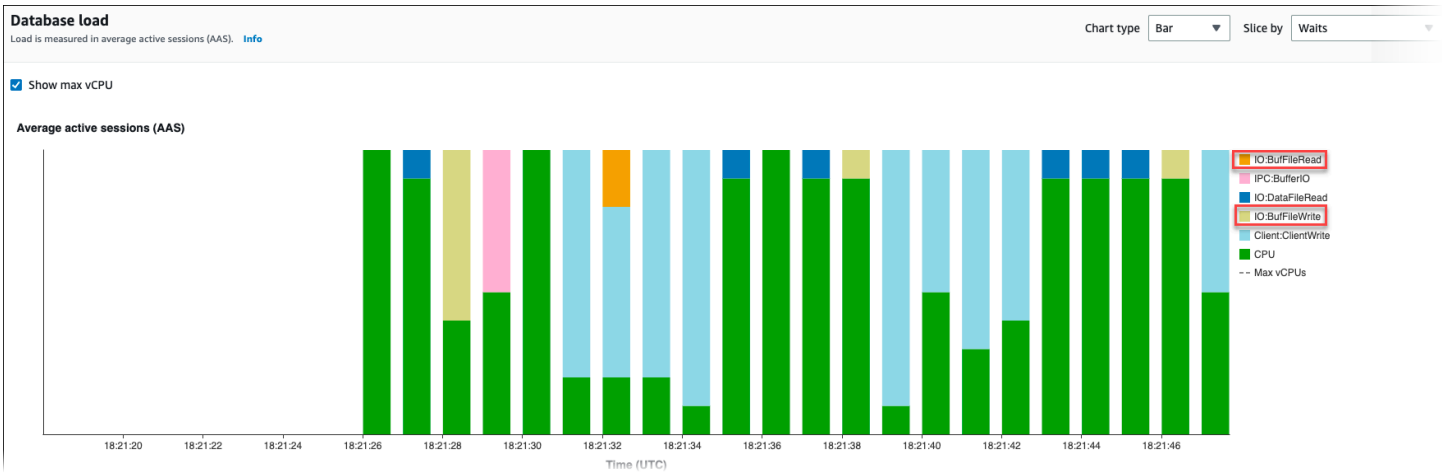
3. Nella SQL scheda Superiore, scegli l'icona Preferenze.
4. Nella finestra Preferenze, attiva le seguenti statistiche in modo che vengano visualizzate nella SQL scheda Superiore e scegli Continua.

   - Scritture temporanee al secondo
   - Letture temporanee al secondo
   - Scritture temporanee in blocco a chiamata
   - Letture temporanee in blocco a chiamata

5. Il file temporaneo viene suddiviso quando viene combinato con la query visualizzata per `pg_ls_tmpdir`, come illustrato nell'esempio seguente.



Gli eventi `IO:BufFileRead` e `IO:BufFileWrite` si verificano quando le query principali del carico di lavoro creano spesso file temporanei. Puoi utilizzare Performance Insights per identificare le principali query in attesa di `IO:BufFileRead` e `IO:BufFileWrite` esaminando Average Active Session (AAS) nelle SQL sezioni Database Load e Top.



Per ulteriori informazioni su come analizzare metriche principali ed eventi di attesa con Approfondimenti sulle prestazioni, consulta Panoramica della scheda Top SQL. Devi individuare e ottimizzare le query che causano un aumento dell'utilizzo dei file temporanei e dei relativi eventi di attesa. Per ulteriori informazioni su questi eventi di attesa e sulla correzione, vedere IO: BufFileRead and IO: BufFileWrite.

> **Note**
>
> Il parametro `work_mem` controlla quando l'operazione di ordinamento esaurisce la memoria; i risultati vengono scritti in file temporanei. Si consiglia di non modificare l'impostazione di questo parametro specificando un valore superiore al valore predefinito perché ciò causerebbe un maggiore utilizzo della memoria da parte di ciascuna sessione del database. Inoltre, una sessione che esegue unioni e ordinamenti complessi può eseguire operazioni parallele in cui ogni operazione consuma memoria.

Come best practice, in presenza di un report di grandi dimensioni con più unioni e ordinamenti, imposta questo parametro a livello di sessione utilizzando il comando `SET work_mem`. La modifica verrà quindi applicata solo alla sessione corrente e non comporterà la modifica del valore a livello globale.

Sintonizzazione degli eventi di attesa per Aurora PostgreSQL

Gli eventi di attesa sono un importante strumento di regolazione per Aurora PostgreSQL. Quando si scopre perché le sessioni sono in attesa di risorse e l'azione svolta, è possibile risolvere i colli di bottiglia in maniera più efficiente. È possibile utilizzare le informazioni contenute in questa sezione per trovare possibili cause e azioni correttive. Prima di approfondire questa sezione, è opportuno comprendere i concetti di base di Aurora, in particolare i seguenti argomenti:

- Archiviazione Amazon Aurora
- Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora

Important

Gli eventi di attesa in questa sezione sono specifici di Aurora PostgreSQL. Utilizza le informazioni contenute in questa sezione per ottimizzare solo Amazon Aurora, non RDS per PostgreSQL.
Alcuni eventi di attesa in questa sezione non hanno analoghi nelle versioni open source di questi motori di database. Altri eventi di attesa hanno lo stesso nome degli eventi nei motori open source, ma si comportano in modo diverso. Ad esempio, lo storage Amazon Aurora funziona in modo diverso dallo storage open source, pertanto gli eventi di attesa relativi allo storage indicano condizioni di risorse diverse.

Argomenti

- Concetti essenziali per l'ottimizzazione di Aurora PostgreSQL
- Eventi di attesa Aurora PostgreSQL
- Cliente: ClientRead
- Cliente: ClientWrite
- CPU

- IO:buffileRead e IO:buffileWrite
- IO: DataFileRead
- IO: XactSync
- IPC:DamRecordTxAck
- IPC: eventi di attesa paralleli
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:buffer_content (BufferContent)
- LWLock:buffer_mapping
- LWLock:bufferIO (IPC: bufferio)
- LWLock:lock_manager
- LWLock:MultiXact
- Timeout: PG Sleep

Concetti essenziali per l'ottimizzazione di Aurora PostgreSQL

Prima di sintonizzare il database Aurora PostgreSQL, assicurati di sapere cosa sono gli eventi di attesa e perché si verificano. Esamina anche l'architettura di base della memoria e del disco di Aurora PostgreSQL. Per un utile diagramma architettonico, vedere il wikibook PostgreSQL.

Argomenti

- Eventi di attesa Aurora PostgreSQL
- Memoria Aurora PostgreSQL
- Processi Aurora PostgreSQL

Eventi di attesa Aurora PostgreSQL

Un evento di attesa indica una risorsa per la quale è in attesa di una sessione. Ad esempio, l'evento di attesa `Client:ClientRead` si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client. Le risorse tipiche che una sessione attende includono quanto segue:

- Accesso a thread singolo a un buffer, ad esempio, quando una sessione sta tentando di modificare un buffer
- Una riga attualmente bloccata da un'altra sessione
- Un file di dati letto
- Scrittura di un file log

Ad esempio, per soddisfare una query, la sessione potrebbe eseguire una scansione completa della tabella. Se i dati non sono già in memoria, la sessione attende il completamento dell'I/O del disco. Quando i buffer vengono letti in memoria, potrebbe essere necessario attendere perché altre sessioni accedono agli stessi buffer. Il database registra le attese utilizzando un evento di attesa predefinito. Questi eventi sono raggruppati in categorie.

Un evento di attesa non mostra di per sé un problema di prestazioni. Ad esempio, se i dati richiesti non sono in memoria, è necessaria la lettura dei dati dal disco. Se una sessione blocca una riga per un aggiornamento, un'altra sessione attende che la riga venga sbloccata in modo che possa aggiornarla. Un commit richiede di attendere il completamento della scrittura su un file di registro. Le attese sono parte integrante del normale funzionamento di un database.

Un gran numero di eventi di attesa in genere mostrano un problema di prestazioni. In questi casi, è possibile utilizzare i dati degli eventi di attesa per determinare dove stanno trascorrendo il tempo delle sessioni. Ad esempio, se un report che in genere viene eseguito in minuti ora viene eseguito per ore, è possibile identificare gli eventi di attesa che contribuiscono maggiormente al tempo di attesa totale. Se è possibile determinare le cause degli eventi di attesa principali, a volte è possibile apportare modifiche che migliorano le prestazioni. Ad esempio, se la sessione è in attesa su una riga bloccata da un'altra sessione, è possibile terminare la sessione di blocco.

Memoria Aurora PostgreSQL

La memoria Aurora PostgreSQL è divisa in condivisa e locale.

Argomenti

- Memoria condivisa in Aurora PostgreSQL
- Memoria locale in Aurora PostgreSQL

Memoria condivisa in Aurora PostgreSQL

Aurora PostgreSQL assegna memoria condivisa all'avvio dell'istanza. La memoria condivisa è divisa in più sottoaree. Di seguito è possibile trovare una descrizione dei più importanti.

Argomenti

- Buffer condivisi
- Buffer Write ahead log (WAL)

Buffer condivisi

Il pool buffer condiviso è un'area di memoria Aurora PostgreSQL che contiene tutte le pagine che sono o sono state utilizzate dalle connessioni delle applicazioni. Una pagina è la versione di memoria di un blocco disco. Il buffer pool condiviso memorizza nella cache i blocchi di dati letti dal disco. Il pool riduce la necessità di rileggere i dati dal disco, rendendo il database più efficiente.

Ogni tabella e indice vengono memorizzati come una matrice di pagine di dimensioni fisse. Ogni blocco contiene più tuple, che corrispondono alle righe. Una tupla può essere memorizzata in qualsiasi pagina.

Il buffer pool condiviso ha memoria finita. Se una nuova richiesta richiede una pagina che non è in memoria, e non esiste più memoria, Aurora PostgreSQL sfratterà una pagina utilizzata meno frequentemente per soddisfare la richiesta. La politica di sfratto è implementata da un algoritmo di sweep dell'orologio.

Il parametro `shared_buffers` determina la quantità di memoria che il server dedica alla memorizzazione nella cache dei dati.

Buffer Write ahead log (WAL)

Un Buffer write-ahead log (WAL) conserva i dati delle transazioni che Aurora PostgreSQL successivamente scrive sullo storage persistente. Utilizzando il meccanismo WAL, Aurora PostgreSQL può effettuare le seguenti operazioni:

- Recuperare i dati dopo un errore
- Ridurre l'I/O del disco evitando scritture frequenti su disco

Quando un client modifica i dati, Aurora PostgreSQL scrive le modifiche nel buffer WAL. Quando il client emette un `COMMIT`, il processo di scrittura WAL scrive i dati delle transazioni nel file WAL.

Il parametro `wal_level` determina la quantità di informazioni scritte sul WAL.

Memoria locale in Aurora PostgreSQL

Ogni processo di back-end assegna memoria locale per l'elaborazione delle query.

Argomenti

- Area di memoria di lavoro
- Area memoria di lavoro di manutenzione
- Area buffer temporanea

Area di memoria di lavoro

L'area di memoria di lavoro contiene dati temporanei per query che eseguono ordinamenti e hash. Ad esempio, una query con clausola `ORDER BY` esegue un ordinamento. Le query utilizzano tabelle hash nei join e nelle aggregazioni hash.

Il parametro `work_mem` indica la quantità di memoria da utilizzare dalle operazioni di ordinamento interno e dalle tabelle hash prima di scrivere su file di disco temporanei. Il valore predefinito è 4 MB. È possibile eseguire più sessioni contemporaneamente e ogni sessione può eseguire operazioni di manutenzione in parallelo. Per questo motivo, la memoria di lavoro totale utilizzata può essere costituita da multipli dell'impostazione `work_mem`.

Area memoria di lavoro di manutenzione

L'area di memoria di lavoro di manutenzione memorizza nella cache i dati per le operazioni di manutenzione. Queste operazioni includono l'aspirazione, la creazione di un indice e l'aggiunta di chiavi esterne.

Il parametro `maintenance_work_mem` specifica la quantità massima di memoria da utilizzare nelle operazioni di manutenzione. Il valore predefinito è 64 MB. Una sessione di database può eseguire solo un'operazione di manutenzione alla volta.

Area buffer temporanea

L'area buffer temporanea memorizza nella cache le tabelle temporanee per ciascuna sessione del database.

Ogni sessione assegna buffer temporanei secondo necessità fino al limite specificato. Quando la sessione scade, il server cancella i buffer.

Il parametro `temp_buffers` imposta il numero massimo di buffer temporanei utilizzati da ogni sessione. Prima del primo utilizzo di tabelle temporanee all'interno di una sessione, è possibile modificare il valore `temp_buffers`.

Processi Aurora PostgreSQL

Aurora PostgreSQL utilizza più processi.

Argomenti

- Processo postmaster
- Processi di back-end
- Processi in background

Processo postmaster

Il processo postmaster è il primo processo avviato quando si avvia Aurora PostgreSQL. Il processo postmaster ha le seguenti responsabilità principali:

- Forcella e monitoraggio dei processi in background
- Ricevere le richieste di autenticazione dai processi client e autenticarle prima di consentire al database di servire le richieste

Processi di back-end

Se il postmaster autentica una richiesta del cliente, il postmaster forcherà un nuovo processo di back-end, chiamato anche processo postgres. Un processo client si connette esattamente a un processo back-end. Il processo client e il processo di backend comunicano direttamente senza intervento da parte del processo postmaster.

Processi in background

Il processo postmaster forca diversi processi che eseguono diverse attività di back-end. Alcuni dei più importanti includono quanto segue:

- Scrittore WAL

  Aurora PostgreSQL scrive i dati nel buffer WAL (write ahead logging) nei file di log. Il principio della registrazione write ahead è che il database non può scrivere modifiche ai file di dati fino a quando il database ha scritto i record di log che descrivono tali modifiche su disco. Il meccanismo WAL

riduce l'I/O del disco e consente ad Aurora PostgreSQL di utilizzare i log per recuperare il database dopo un errore.

- Background writer

  Questo processo scrive periodicamente pagine sporche (modificate) dai buffer di memoria ai file di dati. Una pagina diventa sporca quando un processo di back-end la modifica in memoria.

- Il daemon dell'Autovacuum

  Il daemon include i seguenti elementi:

  - Il lanciatore di autovacuum
  - I processi di autovacuum worker

  Se abilitata, verifica la presenza di tabelle con un numero elevato di tuple inserite, aggiornate o eliminate. Il daemon ha le seguenti responsabilità:

  - Recuperare o riutilizzare lo spazio su disco occupato da righe aggiornate o eliminate
  - Aggiornare le statistiche utilizzate dal planner
  - Proteggere contro la perdita di dati precedenti a causa di un involucro dell'ID transazione

  La funzione autovacuum automatizza l'esecuzione dei comandi `VACUUM` e `ANALYZE`. `VACUUM` ha le seguenti varianti: standard e full. Il vuoto standard funziona in parallelo con altre operazioni di database. `VACUUM FULL` richiede un blocco esclusivo sulla tabella su cui sta lavorando. Pertanto, non può essere eseguito in parallelo con le operazioni che accedono alla stessa tabella. `VACUUM` crea una notevole quantità di traffico I/O, che può causare prestazioni scadenti per altre sessioni attive.

Eventi di attesa Aurora PostgreSQL

La seguente tabella elenca gli eventi di attesa per Aurora PostgreSQL che indicano più comunemente problemi di prestazioni, con cause più comuni e azioni correttive. I seguenti eventi di attesa sono un sottoinsieme dell'elenco in Eventi di attesa Amazon Aurora PostgreSQL.

Evento di attesa	Definizione
Cliente: ClientRead	Ad esempio, l'evento di attesa si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client.

Evento di attesa	Definizione
Cliente: ClientWrite	Ad esempio, l'evento di attesa si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client.
CPU	Questo evento si verifica quando un thread è attivo nella CPU o è in attesa della CPU.
IO:buffileRead e IO:buffileWrite	Questi eventi si verificano quando Aurora PostgreSQL crea file temporanei.
IO: DataFileRead	Questo evento si verifica quando una connessione attende un processo di back-end per leggere una pagina richiesta dalla memoria perché la pagina non è disponibile nella memoria condivisa.
IO: XactSync	Questo evento si verifica quando il database è in attesa che il sottosistema di archiviazione Aurora riconosca il commit di una transazione regolare o il commit o il rollback di una transazione preparata.
IPC:DamRecordTxAck	Questo evento si verifica quando Aurora PostgreSQL in una sessione che utilizza flussi di attività del database genera un evento di flusso di attività, quindi attende che tale evento diventi duraturo.
Lock:advisory	Questo evento si verifica quando un'applicazione PostgreSQL utilizza un blocco per coordinare l'attività su più sessioni.
Lock:extend	Questo evento si verifica quando un processo di back-end è in attesa di bloccare una relazione per estenderla mentre un altro processo ha un blocco su tale relazione per lo stesso scopo.

Evento di attesa	Definizione
Lock:Relation	Questo evento si verifica quando una query è in attesa di acquisire un blocco su una tabella o vista attualmente bloccata da un'altra transazione.
Lock:transactionid	Questo evento si verifica quando una transazione è in attesa di un blocco a livello di riga.
Lock:tuple	Questo evento si verifica quando un processo di backend aspetta di acquisire un blocco su una tupla.
LWLock:buffer_content (BufferContent)	Questo evento si verifica quando una sessione è in attesa di accedere in lettura o scrittura a una pagina dati in memoria mentre un'altra sessione ha bloccato la pagina in scrittura.
LWLock:buffer_mapping	Questo evento si verifica quando un processo di backend è in attesa di associare un blocco di dati a un buffer nel pool di buffer condiviso.
LWLock:bufferIO (IPC: bufferio)	Questo evento si verifica quando Aurora PostgreSQL o RDS per PostgreSQL sono in attesa che altri processi finiscano le operazioni di input/output put (I/O) quando si tenta contemporaneamente di accedere a una pagina.
LWLock:lock_manager	Questo evento si verifica quando il motore Aurora PostgreSQL mantiene l'area di memoria del blocco condiviso per allocare, controllare e dislocare un blocco quando non è possibile un blocco rapido del percorso.

Evento di attesa	Definizione
LWLock:MultiXact	Questo tipo di evento si verifica quando Aurora PostgreSQL mantiene aperta una sessione per completare più transazioni che coinvolgono la stessa riga in una tabella. L'evento di attesa indica quale aspetto dell'elaborazione di più transazioni sta generando l'evento di attesa, ovvero LWLock:MultiXactOffsetSLRU, LWLock:MultiXactOffsetBuffer, LWLock:MultiXactMemberSLRU o LWLock:MultiXactMemberBuffer.
Timeout: PG Sleep	Questo evento si verifica quando un processo server ha chiamato la funzione `pg_sleep` e sta aspettando la scadenza del timeout del sonno.

Cliente: ClientRead

L'evento `Client:ClientRead` si verifica quando Aurora PostgreSQL è in attesa di ricevere dati dal client.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 10 e successive.

Context

Un cluster Aurora PostgreSQL DB è in attesa di ricevere dati dal client. Il cluster Aurora PostgreSQL DB deve ricevere i dati dal client prima di poter inviare più dati al client. Il tempo in cui il cluster attende prima di ricevere i dati dal client è un evento `Client:ClientRead`.

Probabili cause di aumento delle attese

Le cause comuni della comparsa dell'evento `Client:ClientRead` che appare nelle prime attese includono:

Maggiore latenza di rete

Potrebbe esserci una maggiore latenza di rete tra il cluster Aurora PostgreSQL DB e il client. Una maggiore latenza di rete aumenta il tempo necessario per la ricezione dei dati dal client del cluster DB.

Aumento del carico sul client

Potrebbe esserci una pressione della CPU o una saturazione della rete sul client. Un aumento del carico sul client può ritardare la trasmissione dei dati dal client al cluster Aurora PostgreSQL DB.

Eccesso di viaggi di andata e ritorno in rete

Un gran numero di viaggi di andata e ritorno in rete tra il cluster Aurora PostgreSQL DB e il client possono ritardare la trasmissione dei dati dal client al cluster Aurora PostgreSQL DB.

Operazione copia di grandi dimensioni

Durante un'operazione di copia, i dati vengono trasferiti dal file system del client al cluster Aurora PostgreSQL DB. L'invio di una grande quantità di dati al cluster DB può ritardare la trasmissione dei dati dal client al cluster DB.

Connessione client inattiva

Una connessione a un'istanza database di Aurora PostgreSQL è inattiva in stato di transazione ed è in attesa che un client invii più dati o un comando. Questo stato può comportare un aumento degli eventi `Client:ClientRead`.

PgBouncer utilizzato per il pool di connessioni

PgBouncer ha un'impostazione di configurazione di rete di basso livello denominata`pkt_buf`, che per impostazione predefinita è impostata su 4.096. Se il carico di lavoro consiste nell'invio

di pacchetti di query di dimensioni superiori a 4.096 byte PgBouncer, si consiglia di aumentare l'impostazione a 8.192. `pkt_buf` Se la nuova impostazione non diminuisce il numero di eventi `Client:ClientRead`, consigliamo di aumentare l'impostazione `pkt_buf` su valori più grandi, come 16.384 o 32.768. Se il testo della query è grande, l'impostazione più grande può essere particolarmente utile.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster
- Ridimensionare il client
- Utilizza istanze di generazione corrente
- Aumentare la larghezza di banda di rete
- Monitora il massimo delle prestazioni di rete
- Monitorare le transazioni nello stato «inattivo nella transazione»

Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster

Per ridurre la latenza di rete e aumentare il throughput di rete, posizionare i client nella stessa subnet di Availability Zone e Virtual Private Cloud (VPC) del cluster Aurora PostgreSQL DB. Assicurarsi che i client siano il più geograficamente vicini possibile al cluster DB.

Ridimensionare il client

Utilizzando Amazon CloudWatch o altri parametri relativi all'host, stabilisci se il tuo client è attualmente limitato dalla CPU o dalla larghezza di banda della rete o da entrambi. Se il client è vincolato, ridimensionare il client di conseguenza.

Utilizza istanze di generazione corrente

In alcuni casi, potresti non utilizzare una classe di istanza DB che supporta i frame jumbo. Se stai eseguendo l'applicazione su Amazon EC2, considera l'utilizzo di un'istanza di generazione corrente per il client. Inoltre, configura l'unità di trasmissione massima (MTU) sul sistema operativo client. Questa tecnica potrebbe ridurre il numero di round trip di rete e aumentare il throughput di rete. Per ulteriori informazioni, consulta Jumbo frames (9001 MTU) nella Amazon EC2 User Guide.

Per informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora. Per determinare la classe di istanza DB equivalente a un tipo di istanza Amazon EC2, posizionare `db.` prima del nome del tipo di istanza Amazon EC2. Ad esempio, l'istanza Amazon EC2 `r5.8xlarge` è equivalente alla classe di istanza DB `db.r5.8xlarge`.

Aumentare la larghezza di banda di rete

Usa `NetworkReceiveThroughput` i CloudWatch parametri di `NetworkTransmitThroughput` Amazon per monitorare il traffico di rete in entrata e in uscita sul cluster DB. Questi parametri possono aiutarti a determinare se la larghezza di banda della rete è sufficiente per il tuo carico di lavoro.

Se la larghezza di banda della rete non è sufficiente, aumentala. Se il AWS client o l'istanza DB stanno raggiungendo i limiti di larghezza di banda di rete, l'unico modo per aumentare la larghezza di banda è aumentare le dimensioni dell'istanza DB.

Per ulteriori informazioni sulle CloudWatch metriche, consulta. CloudWatch Parametri Amazon per Amazon Aurora

Monitora il massimo delle prestazioni di rete

Se utilizzi client Amazon EC2, Amazon EC2 fornisce il massimo per i parametri delle prestazioni di rete, inclusa la larghezza di banda aggregata in entrata e in uscita. Fornisce inoltre il monitoraggio della connessione per garantire che i pacchetti vengano restituiti come previsto e l'accesso ai servizi locali per servizi come il DNS (Domain Name System). Per monitorare questi massimi, utilizza un driver di rete avanzato e monitora le prestazioni di rete per il client.

Per ulteriori informazioni, consulta Monitoraggio delle prestazioni di rete per l'istanza Amazon EC2 nella Guida per l'utente di Amazon EC2 e Monitoraggio delle prestazioni di rete per l'istanza Amazon EC2 nella Guida per l'utente di Amazon EC2.

Monitorare le transazioni nello stato «inattivo nella transazione»

Controlla se hai un numero crescente di connessioni `idle in transaction`. Per fare ciò, monitora la colonna `state` nella tabella `pg_stat_activity`. Potrebbe essere possibile identificare l'origine della connessione eseguendo una query simile alla seguente.

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
```

```
order by 3 desc
```

Cliente: ClientWrite

L'evento `Client:ClientWrite` si verifica quando Aurora PostgreSQL è in attesa di scrivere dati sul client.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 10 e successive.

Context

Il processo client deve ricevere i dati da un cluster database Aurora PostgreSQL prima di poter inviare più dati. Il tempo in cui il cluster attende prima inviare altri dati al client è un evento `Client:ClientWrite`.

Il throughput di rete ridotto tra il cluster Aurora PostgreSQL DB e il client può causare questo evento. Anche la pressione della CPU e la saturazione della rete sul client possono causare questo evento. Pressione CPU è quando la CPU è completamente utilizzata e ci sono attività in attesa del tempo della CPU. Saturazione rete è quando la rete tra il database e il client trasporta più dati di quelli che è in grado di gestire.

Probabili cause di aumento delle attese

Le cause comuni della comparsa dell'evento `Client:ClientWrite` che appare nelle prime attese includono:

Maggiore latenza di rete

Potrebbe esserci una maggiore latenza di rete tra il cluster Aurora PostgreSQL DB e il client. Una maggiore latenza di rete aumenta il tempo necessario per la ricezione dei dati dal client.

Aumento del carico sul client

Potrebbe esserci una pressione della CPU o una saturazione della rete sul client. Un aumento del carico sul client ritarda la ricezione dei dati dal cluster Aurora PostgreSQL DB.

Ampio volume di dati inviati al client

Il cluster Aurora PostgreSQL DB potrebbe inviare una grande quantità di dati al client. Un client potrebbe non essere in grado di ricevere i dati con la stessa rapidità dell'invio del cluster. Attività come una copia di una tabella di grandi dimensioni possono comportare un aumento degli eventi `Client:ClientWrite`.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster
- Utilizza istanze di generazione corrente
- Ridurre la quantità di dati inviati al client
- Ridimensionare il client

Posizionare i client nella stessa area di disponibilità e subnet VPC del cluster

Per ridurre la latenza di rete e aumentare il throughput di rete, posizionare i client nella stessa subnet di Availability Zone e Virtual Private Cloud (VPC) del cluster Aurora PostgreSQL DB.

Utilizza istanze di generazione corrente

In alcuni casi, potresti non utilizzare una classe di istanza DB che supporta i frame jumbo. Se stai eseguendo l'applicazione su Amazon EC2, considera l'utilizzo di un'istanza di generazione corrente per il client. Inoltre, configura l'unità di trasmissione massima (MTU) sul sistema operativo client. Questa tecnica potrebbe ridurre il numero di round trip di rete e aumentare il throughput di rete. Per ulteriori informazioni, consulta Jumbo frames (9001 MTU) nella Amazon EC2 User Guide.

Per informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora. Per determinare la classe di istanza DB equivalente a un tipo di istanza Amazon EC2, posizionare `db.` prima del nome del tipo di istanza Amazon EC2. Ad esempio, l'istanza Amazon EC2 `r5.8xlarge` è equivalente alla classe di istanza DB `db.r5.8xlarge`.

Ridurre la quantità di dati inviati al client

Quando possibile, regolare l'applicazione per ridurre la quantità di dati che il cluster Aurora PostgreSQL DB invia al client. Effettuare tali regolazioni allevia la contesa della CPU e della rete sul client.

Ridimensionare il client

Utilizzando Amazon CloudWatch o altri parametri relativi all'host, stabilisci se il tuo client è attualmente limitato dalla CPU o dalla larghezza di banda della rete o da entrambi. Se il client è vincolato, ridimensionare il client di conseguenza.

CPU

Questo evento si verifica quando un thread è attivo nella CPU o è in attesa della CPU.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

L’unità di elaborazione centrale (CPU) è il componente di un computer che esegue le istruzioni. Ad esempio, le istruzioni della CPU eseguono operazioni aritmetiche e scambiano dati in memoria. Se una query aumenta il numero di istruzioni eseguite tramite il motore di database, il tempo impiegato per eseguire la query aumenta. Pianificazione CPU sta dando tempo alla CPU a un processo. La pianificazione viene orchestrata dal kernel del sistema operativo.

Argomenti

- Come sapere quando si verifica questa attesa
- Parametro DBLOADCPU

- Parametri di utilizzo di OS.CPU
- Probabile causa della pianificazione della CPU

Come sapere quando si verifica questa attesa

Questo evento di attesa CPU indica che un processo di backend è attivo nella CPU o è in attesa della CPU. Si verifica quando una query mostra le seguenti informazioni:

- La colonna `pg_stat_activity.state` ha il valore `active`.
- Le colonne `wait_event_type` e `wait_event` in `pg_stat_activity` sono entrambe `null`.

Per visualizzare i processi di backend in uso o in attesa sulla CPU, eseguire la seguente query.

```
SELECT *
FROM   pg_stat_activity
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

Parametro DBLOADCPU

Il parametro Performance Insights per la CPU è `DBLoadCPU`. Il valore di `DBLoadCPU` può differire dal valore del parametro Amazon CloudWatch `CPUUtilization`. Quest'ultimo parametro viene raccolto dall'HyperVisor per un'istanza di database.

Parametri di utilizzo di OS.CPU

I parametri del sistema operativo Performance Insights forniscono informazioni dettagliate sull'utilizzo della CPU. Ad esempio, è possibile visualizzare i seguenti parametri:

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Performance Insights segnala l'utilizzo della CPU da parte del motore di database come `os.cpuUtilization.nice.avg`.

Probabile causa della pianificazione della CPU

Dal punto di vista del sistema operativo, la CPU è attiva quando non è in esecuzione il thread inattivo. La CPU è attiva mentre esegue un calcolo, ma è attiva anche in attesa di I/O di memoria. Questo tipo di I/O domina un carico di lavoro tipico del database.

È probabile che i processi attendano di essere pianificati su una CPU quando vengono soddisfatte le seguenti condizioni:

- Il parametro CloudWatch `CPUUtilization` è vicino al 100 percento.
- Il carico medio è superiore al numero di vCPU, indicando un carico pesante. Puoi trovare il parametro `loadAverageMinute` nella sezione parametri del sistema operativo in Performance Insights.

Probabili cause di aumento delle attese

Quando l'evento di attesa della CPU si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Argomenti

- Probabili cause di picchi improvvisi
- Probabili cause di alta frequenza a lungo termine
- Casi particolari

Probabili cause di picchi improvvisi

Le cause più probabili di picchi improvvisi sono le seguenti:

- L'applicazione ha aperto troppe connessioni simultanee al database. Questo scenario è noto come «tempesta di connessione».
- Il carico di lavoro dell'applicazione è cambiato in uno dei seguenti modi:
  - Nuove query
  - Aumento delle dimensioni del set di dati
  - Manutenzione o creazione dell'indice
  - Nuove funzioni
  - Nuovi operatori

  - Un aumento dell'esecuzione parallela di query
- I piani di esecuzione delle query sono cambiati. In alcuni casi, un cambiamento può causare un aumento dei buffer. Ad esempio, la query ora utilizza una scansione sequenziale quando in precedenza utilizzava un indice. In questo caso, le query richiedono più CPU per raggiungere lo stesso obiettivo.

Probabili cause di alta frequenza a lungo termine

Le cause più probabili di eventi che si ripetono per un lungo periodo:

- Troppi processi backend sono in esecuzione contemporaneamente sulla CPU. Questi processi possono essere lavoratori paralleli.
- Le query vengono eseguite in modo subottimale perché necessitano di un numero elevato di buffer.

Casi particolari

Se nessuna delle cause probabili risulta essere una causa effettiva, potrebbero verificarsi le seguenti situazioni:

- La CPU sta scambiando i processi in entrata e in uscita.
- Il passaggio del contesto della CPU è aumentato.
- Nel codice Aurora PostgreSQL mancano gli eventi di attesa.

Operazioni

Se l'evento di attesa CPU domina l'attività del database, non indica necessariamente un problema di prestazioni. Rispondi a questo evento solo quando le prestazioni diminuiscono.

Argomenti

- Indagare se il database sta causando l'aumento della CPU
- Determina se il numero di connessioni è aumentato
- Rispondere alle modifiche del carico di lavoro

Indagare se il database sta causando l'aumento della CPU

Esamina il parametro `os.cpuUtilization.nice.avg` in Performance Insights. Se questo valore è molto inferiore all'utilizzo della CPU, i processi nondatabase sono il principale contributore della CPU.

Determina se il numero di connessioni è aumentato

Esamina il parametro `DatabaseConnections` in Amazon CloudWatch. L'azione dipende dal fatto che il numero sia aumentato o diminuito durante il periodo di aumento degli eventi di attesa della CPU.

Le connessioni sono aumentate

Se il numero di connessioni è aumentato, confrontare il numero di processi back-end che consumano la vCPU con il numero di vCPU. Gli scenari possibili sono i seguenti:

- Il numero di processi backend che consumano vCPU è inferiore al numero di vCPU.

  In questo caso, il numero di connessioni non è un problema. Tuttavia, è comunque possibile provare a ridurre l'utilizzo della CPU.
- Il numero di processi backend che consumano vCPUs è inferiore al numero di vCPU.

  In questo caso, valuta le seguenti opzioni:
  - Riduci il numero di processi back-end collegati al database. Ad esempio, implementa una soluzione di connection pooling come RDS Proxy. Per ulteriori informazioni, consultare Proxy Amazon RDS per Aurora.
  - Aggiorna le dimensioni dell'istanza per ottenere un numero maggiore di vCPUs.
  - Reindirizza alcuni carichi di lavoro di sola lettura ai nodi del lettore, se applicabile.

Le connessioni sono aumentate

Esamina il parametro `blks_hit` in Performance Insights. Cerca una correlazione tra un aumento `blks_hit` e utilizzo CPU. Gli scenari possibili sono i seguenti:

- Utilizzo CPU e `blks_hit` sono correlati.

  In questo caso, trova le istruzioni SQL principali collegate all'utilizzo della CPU e cerca le modifiche al piano. Puoi utilizzare una delle seguenti tecniche:
  - Spiegare i piani manualmente e confrontarli con il piano di esecuzione previsto.

- Cercare un aumento degli hit di blocco al secondo e dei blocchi locali al secondo. Nella sezione Top SQL della dashboard Performance Insights, scegli Preferenze.

- Utilizzo CPU e `blks_hit` non sono correlati.

  In questo caso, determinare se si verifica una delle seguenti condizioni:

  - L'applicazione si sta rapidamente connettendo e disconnettendo dal database.

    Diagnosticare questo comportamento attivando `log_connections` e `log_disconnections`, quindi analizzando i registri PostgreSQL. Considerare l'utilizzo dell'analizzatore di log `pgbadger`. Per ulteriori informazioni, consultare https://github.com/darold/pgbadger.

  - Il sistema operativo è sovraccarico.

    In questo caso, Performance Insights mostra che i processi back-end consumano la CPU per un tempo più lungo del solito. Cerca prove nei parametri `os.cpuUtilization` di Performance Insights o nel parametro CloudWatch `CPUUtilization`. Se il sistema operativo è sovraccarico, esamina i parametri di Enhanced Monitoring per effettuare ulteriori diagnosi. In particolare, guarda l'elenco dei processi e la percentuale di CPU consumata da ciascun processo.

  - Le istruzioni SQL principali consumano troppa CPU.

    Esaminare le istruzioni collegate all'utilizzo della CPU per verificare se possono utilizzare meno CPU. Esegui il comando `EXPLAIN` e concentrati sui nodi del piano che hanno il maggior impatto. Prendi in considerazione l'utilizzo di un visualizzatore del piano di esecuzione PostgreSQL. Per provare questo strumento, vedi http://explain.dalibo.com/.

Rispondere alle modifiche del carico di lavoro

Se il carico di lavoro è cambiato, cerca i seguenti tipi di modifiche:

Nuove query

Verifica se sono previste le nuove query. In tal caso, assicurarsi che siano previsti i piani di esecuzione e il numero di esecuzioni al secondo.

Aumento delle dimensioni del set di dati

Determina se il partizionamento, se non è già implementato, potrebbe essere d'aiuto. Questa strategia potrebbe ridurre il numero di pagine che una query deve recuperare.

Manutenzione o creazione dell'indice

Verificare se è previsto il programma per la manutenzione. Una best practice è pianificare le attività di manutenzione al di fuori delle attività di picco.

Nuove funzioni

Verifica se queste funzioni funzionano come previsto durante il test. In particolare, controlla se è previsto il numero di esecuzioni al secondo.

Nuovi operatori

Verifica se queste funzioni funzionano come previsto durante il test.

Un aumento dell'esecuzione di query parallele

Determina se si è verificata una delle seguenti situazioni:

- Le relazioni o gli indici coinvolti sono improvvisamente cresciuti di dimensioni in modo che differiscano in modo significativo da `min_parallel_table_scan_size` o `min_parallel_index_scan_size`.
- Cambiamenti recenti sono stati apportati a `parallel_setup_cost` o `parallel_tuple_cost`.
- Cambiamenti recenti sono stati apportati a `max_parallel_workers` o `max_parallel_workers_per_gather`.

IO:buffileRead e IO:buffileWrite

Gli eventi `IO:BufFileRead` e `IO:BufFileWrite` si verificano quando Aurora PostgreSQL crea file temporanei. Quando le operazioni richiedono più memoria rispetto ai parametri della memoria di lavoro attualmente definiti, scrivono dati temporanei sullo storage persistente. Questa operazione è talvolta chiamata «versamento su disco».

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

`IO:BufFileRead` e `IO:BufFileWrite` riguardano l'area di memoria di lavoro e l'area di memoria di lavoro di manutenzione. Per ulteriori informazioni su queste aree di memoria locale, consulta Area di memoria di lavoro e Area memoria di lavoro di manutenzione.

Il valore predefinito per `work_mem` è 4 MB. Se una sessione esegue operazioni in parallelo, ogni lavoratore che gestisce il parallelismo utilizza 4 MB di memoria. Per questo motivo, imposta `work_mem` con attenzione. Se si aumenta troppo il valore, un database che esegue molte sessioni potrebbe consumare troppa memoria. Se si imposta il valore troppo basso, Aurora PostgreSQL crea file temporanei nella memoria locale. L'I/O del disco per questi file temporanei può ridurre le prestazioni.

Se si osserva la seguente sequenza di eventi, il database potrebbe generare file temporanei:

1. Riduzione improvvisa e brusca della disponibilità
2. Ripristino rapido per lo spazio libero

Potresti vedere anche un motivo a «motosega». Questo modello può indicare che il database sta creando costantemente file di piccole dimensioni.

Probabili cause di aumento delle attese

In generale, questi eventi di attesa sono causati da operazioni che consumano più memoria rispetto a quella allocata dai parametri `work_mem` o `maintenance_work_mem`. Per compensare, le operazioni scrivono su file temporanei. Cause comuni degli eventi `IO:BufFileRead` e `IO:BufFileWrite` includono quanto segue:

Query che richiedono più memoria di quella esistente nell'area della memoria di lavoro

Le query con le seguenti caratteristiche utilizzano l'area di memoria di lavoro:

- Hash join
- `ORDER BY`Clausola

- `GROUP BY`Clausola
- `DISTINCT`
- Funzioni finestra
- `CREATE TABLE AS SELECT`
- Aggiornamento vista materializzata

Istruzioni che richiedono più memoria di quella esistente nell'area della memoria di lavoro di manutenzione

Le istruzioni seguenti utilizzano l'area di memoria di lavoro di manutenzione:

- `CREATE INDEX`
- `CLUSTER`

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Identificare il problema
- Esamina le tue query di join
- Esamina le query ORDER BY e GROUP BY
- Evitare di utilizzare l'operazione DISTINCT
- Considera l'utilizzo di funzioni finestra anziché le funzioni GROUP BY
- Indagare sulle viste materializzate e le istruzioni CTAS
- Usa pg_repack quando crei indici
- Aumenta maintenance_work_mem quando esegui cluster
- Sintonizza la memoria per impedire IO:BuffileRead e IO:BuffileWrite

Identificare il problema

Supponiamo una situazione in cui Performance Insights non è attivato e sospetti che `IO:BufFileRead` e `IO:BufFileWrite` si verificano più frequentemente del normale. Esegui questa operazione:

1. Esamina il parametro `FreeLocalStorage` in Amazon CloudWatch.
2. Cerca un motivo a motosega, che è una serie di punte frastagliate.

Un motivo a motosega indica un consumo e un rilascio rapido dello storage, spesso associato a file temporanei. Se noti questo modello, attiva Performance Insights. Quando si utilizza Performance Insights, è possibile identificare quando si verificano gli eventi di attesa e quali query sono associate. La soluzione dipende dalla query specifica che causa gli eventi.

Oppure, imposta il parametro `log_temp_files`. Questo parametro registra tutte le query che generano più della soglia KB di file temporanei. Se il valore è `0`, Aurora PostgreSQL registra tutti i file temporanei. Se il valore è `1024`, Aurora PostgreSQL registra tutte le query che producono file temporanei di dimensioni superiori a 1 MB. Per ulteriori informazioni su `log_temp_files`, consulta Creazione di log e report di errore nella documentazione di PostgreSQL.

Esamina le tue query di join

Probabilmente la tua applicazione usa join. Ad esempio, la seguente query unisce quattro tabelle.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

Una possibile causa di picchi nell'utilizzo temporaneo dei file è un problema nella query stessa. Ad esempio, una clausola interrotta potrebbe non filtrare correttamente i join. Considera il secondo inner join nell'esempio seguente.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = customer.id)
 INNER JOIN customer_address
```

```
      ON (customer_address.customer_id = customer.id AND
          order.customer_address_id = customer_address.id)
WHERE customer.id = 1234567890;
```

La query precedente unisce erroneamente `customer.id` a `customer.id`, generando un prodotto cartesiano tra ogni cliente e ogni ordine. Questo tipo di join accidentale genera file temporanei di grandi dimensioni. A seconda delle dimensioni delle tabelle, una query cartesiana può anche riempire lo spazio di archiviazione. La domanda potrebbe avere join cartesiani quando vengono soddisfatte le seguenti condizioni:

- Si notano forti e nitide riduzioni della disponibilità dello storage, seguite da un rapido ripristino.
- Non sono in fase di creazione di indici.
- Non vengono rilasciate istruzioni `CREATE TABLE FROM SELECT`.
- Nessuna vista materializzata viene aggiornata.

Per verificare se le tabelle vengono unite utilizzando le chiavi appropriate, ispezionare le direttive di mappatura relazionale di query e oggetti. Tieni presente che alcune query della tua applicazione non vengono sempre chiamate e alcune query vengono generate dinamicamente.

Esamina le query ORDER BY e GROUP BY

In alcuni casi, una clausola `ORDER BY` può comportare file temporanei eccessivi. Considera le linee guida seguenti:

- Includi solo colonne in una clausola `ORDER BY` quando devono essere ordinate. Questa linea guida è particolarmente importante per le query che restituiscono migliaia di righe e specificano molte colonne nella clausola `ORDER BY`.
- Considerando la creazione di indici per accelerare clausole `ORDER BY` quando corrispondono a colonne che hanno lo stesso ordine crescente o decrescente. Gli indici parziali sono preferibili perché sono più piccoli. Gli indici più piccoli vengono letti e attraversati più rapidamente.
- Se si creano indici per colonne che possono accettare valori nulli, considerare se si desidera che i valori nulli siano memorizzati alla fine o all'inizio degli indici.

  Se possibile, ridurre il numero di righe che devono essere ordinate filtrando il set di risultati. Se utilizzi istruzioni clausole o sottoquery `WITH`, ricorda che una query interna genera un set di risultati e lo passa alla query esterna. Maggiore è il numero di righe che una query può filtrare, minore è la necessità di ordinare la query.

- Se non è necessario ottenere il set completo di risultati, utilizzare la clausola `LIMIT`. Ad esempio, se si desidera solo le prime cinque righe, una query che utilizza la clausola `LIMIT` non continua a generare risultati. In questo modo, la query richiede meno memoria e file temporanei.

Una query che utilizza una clausola `GROUP BY` può richiedere anche file temporanei. Le query `GROUP BY` riepilogano i valori utilizzando funzioni come le seguenti:

- `COUNT`
- `AVG`
- `MIN`
- `MAX`
- `SUM`
- `STDDEV`

Per sintonizzare le query `GROUP BY`, segui i consigli per le query `ORDER BY`.

Evitare di utilizzare l'operazione DISTINCT

Se possibile, evitare di utilizzare l'operazione `DISTINCT` per rimuovere righe duplicate. Più righe non necessarie e duplicate restituite dalla tua query, più costosa diventa l'operazione `DISTINCT`. Se possibile, aggiungi filtri nella clausola `WHERE` anche se utilizzi gli stessi filtri per tabelle diverse. Filtrare la query e unirsi correttamente migliora le prestazioni e riduce l'utilizzo delle risorse. Previene inoltre report e risultati errati.

Se devi usare `DISTINCT` per più righe di una stessa tabella, prendi in considerazione la possibilità di creare un indice composito. Il raggruppamento di più colonne in un indice può migliorare il tempo necessario per valutare righe distinte. Inoltre, se utilizzi Amazon Aurora PostgreSQL versione 10 o successiva, puoi correlare le statistiche tra più colonne utilizzando il comando `CREATE STATISTICS`.

Considera l'utilizzo di funzioni finestra anziché le funzioni GROUP BY

Utilizzando `GROUP BY` si modifica il set di risultati e quindi recupera il risultato aggregato. Utilizzando le funzioni della finestra, si aggregano i dati senza modificare il set di risultati. Una funzione finestra utilizza la clausola `OVER` per eseguire calcoli tra i set definiti dalla query, correlando una riga con un'altra. È possibile utilizzare tutte le funzioni `GROUP BY` nelle funzioni di finestra, ma anche funzioni come le seguenti:

- RANK
- ARRAY_AGG
- ROW_NUMBER
- LAG
- LEAD

Per ridurre al minimo il numero di file temporanei generati da una funzione di finestra, rimuovere le duplicazioni per lo stesso set di risultati quando sono necessarie due aggregazioni distinte. Considera la query seguente.

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

È possibile riscrivere la query con la clausola `WINDOW` come segue.

```
SELECT sum(salary) OVER w as sum_salary
         , avg(salary) OVER w as_avg_salary
    FROM empsalary
  WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

Per impostazione predefinita, il planner di esecuzione Aurora PostgreSQL consolida nodi simili in modo da non duplicare le operazioni. Tuttavia, utilizzando una dichiarazione esplicita per il blocco finestra, è possibile mantenere la query più facilmente. È inoltre possibile migliorare le prestazioni impedendo la duplicazione.

Indagare sulle viste materializzate e le istruzioni CTAS

Quando una vista materializzata si aggiorna, esegue una query. Questa query può contenere un'operazione come `GROUP BY`, `ORDER BY` oppure`DISTINCT`. Durante un aggiornamento, è possibile osservare un numero elevato di file temporanei e gli eventi di attesa `IO:BufFileWrite` e `IO:BufFileRead`. Allo stesso modo, quando crei una tabella basata su una dichiarazione `SELECT`, l'istruzione `CREATE TABLE` esegue una query. Per ridurre i file temporanei necessari, ottimizza la query.

Usa pg_repack quando crei indici

Quando crei un indice, il motore ordina il set di risultati. Man mano che le tabelle aumentano di dimensioni e man mano che i valori nella colonna indicizzata diventano più diversi, i file temporanei richiedono più spazio. Nella maggior parte dei casi, non è possibile impedire la creazione di file temporanei per tabelle di grandi dimensioni senza modificare l'area della memoria di lavoro di manutenzione. Per ulteriori informazioni, consultare Area memoria di lavoro di manutenzione.

Una possibile soluzione alternativa quando si ricrea un indice di grandi dimensioni consiste nell'utilizzare lo strumento pg_repack. Per ulteriori informazioni, consulta Riorganizzare le tabelle nei database PostgreSQL con blocchi minimi nella documentazione di pg_repack.

Aumenta maintenance_work_mem quando esegui cluster

Il comando `CLUSTER` raggruppa la tabella specificata da table_name in base a un indice esistente specificato da index_name. Aurora PostgreSQL ricrea fisicamente la tabella in modo che corrisponda all'ordine di un determinato indice.

Quando lo storage magnetico era prevalente, il clustering era comune perché il throughput di storage era limitato. Ora che lo storage basato su SSD è comune, il clustering è meno popolare. Tuttavia, se si raggruppano le tabelle, è comunque possibile aumentare leggermente le prestazioni a seconda delle dimensioni della tabella, dell'indice, della query e così via.

Se esegui il comando `CLUSTER` e osservi gli eventi di attesa `IO:BufFileWrite` e `IO:BufFileRead`, sintonizza `maintenance_work_mem`. Aumenta la dimensione della memoria a una quantità abbastanza grande. Un valore elevato significa che il motore può utilizzare più memoria per l'operazione di clustering.

Sintonizza la memoria per impedire IO:BuffileRead e IO:BuffileWrite

In alcune situazioni, è necessario sintonizzare la memoria. Il tuo obiettivo è quello di bilanciare i seguenti requisiti:

- Il valore `work_mem` (vedi Area di memoria di lavoro)
- La memoria rimanente dopo aver scontato il valore `shared_buffers` (vedi Pool di buffer)
- Le connessioni massime aperte e in uso, limitate da `max_connections`

Aumentare le dimensioni dell'area di memoria di lavoro

In alcune situazioni, l'unica opzione è aumentare la memoria utilizzata dalla sessione. Se le tue query sono scritte correttamente e utilizzano i tasti corretti per i join, prendi in considerazione la possibilità di aumentare il valore `work_mem`. Per ulteriori informazioni, consultare Area di memoria di lavoro.

Per scoprire quanti file temporanei genera una query, imposta `log_temp_files` su `0`. Aumentando il valore `work_mem` al valore massimo identificato nei log, si impedisce alla query di generare file temporanei. Tuttavia, `work_mem` imposta il massimo per nodo piano per ogni connessione o operatore parallelo. Se il database ha 5.000 connessioni e se ciascuna utilizza una memoria di 256 MiB, il motore necessita di 1,2 TiB di RAM. Pertanto, la tua istanza potrebbe esaurirsi dalla memoria.

Riserva una memoria sufficiente per il buffer pool condiviso

Il database utilizza aree di memoria come il buffer pool condiviso, non solo l'area di memoria di lavoro. Considerare i requisiti di queste aree di memoria aggiuntive prima di aumentare `work_mem`. Per ulteriori informazioni sul pool di buffer, consulta Pool di buffer.

Ad esempio, supponiamo che la classe di istanza Aurora PostgreSQL sia db.r5.2xlarge. Questa classe ha 64 GiB di memoria. Per impostazione predefinita, il 75% della memoria è riservato al buffer pool condiviso. Dopo aver sottratto la quantità allocata all'area di memoria condivisa, rimangono 16.384 MB. Non allocare la memoria rimanente esclusivamente all'area della memoria di lavoro perché anche il sistema operativo e il motore richiedono memoria.

La memoria a cui è possibile allocare per `work_mem` dipende dalla classe di istanza. Se si utilizza una classe di istanza più grande, è disponibile più memoria. Tuttavia, nell'esempio precedente, non è possibile utilizzare più di 16 GiB. In caso contrario, la tua istanza diventa non disponibile quando esaurisce la memoria. Per ripristinare l'istanza dallo stato non disponibile, i servizi di automazione Aurora PostgreSQL si riavviano automaticamente.

Gestisci il numero di connessioni

Supponiamo che l'istanza del database disponga di 5.000 connessioni simultanee. Ogni connessione utilizza almeno 4 MiB di `work_mem`. L'elevato consumo di memoria delle connessioni rischia di peggiorare le prestazioni. Sono disponibili le seguenti opzioni:

- Eseguire l'aggiornamento a una classe di istanza più grande
- Diminuire il numero di connessioni simultanee al database utilizzando un proxy o un pool di connessioni.

Per i proxy, considera Amazon RDS Proxy, PGBouncer o un connection pooler basato sulla tua applicazione. Questa soluzione riduce il carico della CPU. Riduce inoltre il rischio quando tutte le connessioni richiedono l'area di memoria di lavoro. Quando esistono meno connessioni al database, è possibile aumentare il valore di `work_mem`. In questo modo, si riduce il verificarsi degli eventi di attesa `IO:BufFileRead` e `IO:BufFileWrite`. Inoltre, le query in attesa dell'area di memoria di lavoro accelerano in modo significativo.

IO: DataFileRead

L'evento `IO:DataFileRead` si verifica quando una connessione attende un processo di back-end per leggere una pagina richiesta dalla memoria perché la pagina non è disponibile nella memoria condivisa.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sugli eventi di attesa sono supportate per tutte le versioni di Aurora Postgre. SQL

Context

Tutte le interrogazioni e le operazioni di manipolazione dei dati (DML) accedono alle pagine del buffer pool. Le dichiarazioni che possono indurre letture includono `SELECT`, `UPDATE` e `DELETE`. Ad esempio, un `UPDATE` può leggere pagine da tabelle o indici. Se la pagina richiesta o aggiornata non si trova nel buffer pool condiviso, questa lettura può portare all'evento `IO:DataFileRead`.

Poiché il buffer pool condiviso è finito, può essere riempito. In questo caso, le richieste di pagine che non sono in memoria impongono al database di leggere i blocchi dal disco. Se l'evento `IO:DataFileRead` si verifica frequentemente, il buffer pool condiviso potrebbe essere troppo piccolo per adattarsi al carico di lavoro. Questo problema è particolarmente grave per le query `SELECT` che leggono un numero elevato di righe che non rientrano nel buffer pool. Per ulteriori informazioni sul pool di buffer, consulta Pool di buffer.

Probabili cause di aumento delle attese

Cause comuni dell'evento `IO:DataFileRead` includono quanto segue:

Picchi di connessione

Potresti trovare più connessioni che generano lo stesso numero di eventi IO: DataFileRead wait. In questo caso, può verificarsi un picco (aumento improvviso e grande) negli eventi `IO:DataFileRead`.

SELECTe DML istruzioni che eseguono scansioni sequenziali

L'applicazione potrebbe aver eseguito una nuova operazione. Oppure un'operazione esistente potrebbe cambiare a causa di un nuovo piano di esecuzione. In questi casi, cerca tabelle (in particolare tabelle di grandi dimensioni) che abbiano un valore `seq_scan` maggiore. Puoi trovarli interrogando `pg_stat_user_tables`. Per tenere traccia delle query che generano più operazioni di lettura, utilizzare l'estensione `pg_stat_statements`.

CTASe CREATE INDEX per set di dati di grandi dimensioni

A CTASè una `CREATE TABLE AS SELECT` dichiarazione. Se si esegue un CTAS set di dati di grandi dimensioni come origine o si crea un indice su una tabella di grandi dimensioni, l'`IO:DataFileRead`evento può verificarsi. Quando si crea un indice, il database potrebbe dover leggere l'intero oggetto utilizzando una scansione sequenziale. A CTAS genera `IO:DataFile` letture quando le pagine non sono in memoria.

Diversi lavoratori sottovuoto in esecuzione contemporaneamente

Gli operatori del vuoto possono essere attivati manualmente o automaticamente. Raccomandiamo di adottare una strategia aggressiva per il vuoto. Tuttavia, quando una tabella contiene molte righe aggiornate o cancellate, l'attesa `IO:DataFileRead` aumenta. Dopo aver recuperato lo spazio, il tempo dedicato al vuoto su `IO:DataFileRead` diminuisce.

Ingresso di grandi quantità di dati

Quando l'applicazione acquisisce quantità di dati elevate, le operazioni `ANALYZE` potrebbero verificarsi più spesso. Il processo `ANALYZE` può essere attivato da un launcher automatico o richiamato manualmente.

L'operazione `ANALYZE` legge un sottoinsieme della tabella. Il numero di pagine che devono essere scansionate viene calcolato moltiplicando 30 per il valore `default_statistics_target`. Per ulteriori informazioni, consulta la documentazione di

SQLPostgre. Il parametro `default_statistics_target` accetta valori compresi tra 1 e 10.000, dove il valore predefinito è 100.

Fame di risorse

Se le istanze utilizzano o CPU consumano la larghezza di banda della rete, l'`IO:DataFileRead`evento potrebbe verificarsi più frequentemente.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Controlla i filtri predicati per le query che generano attese
- Riduci al minimo l'effetto delle operazioni di manutenzione
- Rispondere a un numero elevato di connessioni

Controlla i filtri predicati per le query che generano attese

Supponiamo di identificare query specifiche che stanno generando eventi di attesa `IO:DataFileRead`. È possibile identificarli utilizzando le seguenti tecniche:

- Approfondimenti sulle prestazioni
- Viste catalogo come quella fornita dall'estensione `pg_stat_statements`
- La vista catalogo `pg_stat_all_tables`, se mostra periodicamente un numero maggiore di letture fisiche
- La vista `pg_statio_all_tables`, se lo mostra che i contatori `_read` sono in aumento

Si consiglia di determinare quali filtri vengono utilizzati nel predicato (clausola `WHERE`) di queste query. Seguire queste linee guida:

- Esegui il comando `EXPLAIN`. Nell'output, identificare quali tipi di scansioni vengono utilizzati. Una scansione sequenziale non indica necessariamente che ci sia un problema. Le query che utilizzano scansioni sequenziali producono naturalmente più eventi `IO:DataFileRead` rispetto alle query che utilizzano filtri.

  Scopri se la colonna elencata nella clausola `WHERE` è indicizzata. In caso contrario, prendi in considerazione la possibilità di creare un indice per questa colonna. Questo approccio evita le

scansioni sequenziali e riduce gli eventi `IO:DataFileRead`. Se una query dispone di filtri restrittivi e continua a produrre scansioni sequenziali, valutare se vengono utilizzati gli indici appropriati.

- Scopri se la query sta accedendo a una tabella molto ampia. In alcuni casi, il partizionamento di una tabella può migliorare le prestazioni, consentendo alla query di leggere solo le partizioni necessarie.
- Esamina la cardinalità (numero totale di righe) dalle operazioni di join. Nota quanto sono restrittivi i valori che stai passando nei filtri per la tua clausola `WHERE`. Se possibile, sintonizza la query per ridurre il numero di righe passate in ogni fase del piano.

Riduci al minimo l'effetto delle operazioni di manutenzione

Operazioni di manutenzione come `VACUUM` e `ANALYZE` sono importanti. Si consiglia di non spegnerli qualora vengano trovati eventi di attesa `IO:DataFileRead` relativi a queste operazioni di manutenzione. I seguenti approcci possono ridurre al minimo l'effetto di queste operazioni:

- Eseguire manualmente le operazioni di manutenzione durante le ore non di punta. Questa tecnica impedisce al database di raggiungere la soglia per le operazioni automatiche.
- Per tabelle molto grandi, prendi in considerazione il partizionamento. Questa tecnica riduce il sovraccarico delle operazioni di manutenzione. Il database accede solo alle partizioni che richiedono manutenzione.
- Quando si acquisiscono grandi quantità di dati, prendere in considerazione la possibilità di disabilitare la funzione di analisi automatica.

La funzione autovacuum viene attivata automaticamente per una tabella quando la formula seguente è vera.

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

La vista `pg_stat_user_tables` e il catalogo `pg_class` hanno più righe. Una riga può corrispondere a una riga della tabella. Questa formula presuppone che i `reltuples` sono per una tabella specifica. I parametri `autovacuum_vacuum_scale_factor` (0,20 per impostazione predefinita) e `autovacuum_vacuum_threshold` (50 tuple per impostazione predefinita) sono generalmente impostate globalmente per l'intera istanza. Tuttavia, è possibile impostare valori diversi per una tabella specifica.

Argomenti

- Ricerca delle tabelle che consumano spazio inutile
- Trova tabelle che consumano spazio inutile
- Trova tabelle idonee per l'autovacuum

Ricerca delle tabelle che consumano spazio inutile

Per trovare le tabelle che consumano spazio più del necessario, esegui la query riportata di seguito. Quando questa query viene eseguita da un utente del database che non ha il ruolo `rds_superuser`, restituisce informazioni solo sulle tabelle per cui il ruolo utente ha le autorizzazioni di lettura. Questa query è supportata dalla SQL versione 12 di Postgre e dalle versioni successive.

```
WITH report AS (
   SELECT   schemaname
            ,tblname
            ,n_dead_tup
            ,n_live_tup
            ,block_size*tblpages AS real_size
            ,(tblpages-est_tblpages)*block_size AS extra_size
            ,CASE WHEN tblpages - est_tblpages > 0
               THEN 100 * (tblpages - est_tblpages)/tblpages::float
               ELSE 0
             END AS extra_ratio, fillfactor, (tblpages-est_tblpages_ff)*block_size AS
 bloat_size
            ,CASE WHEN tblpages - est_tblpages_ff > 0
               THEN 100 * (tblpages - est_tblpages_ff)/tblpages::float
               ELSE 0
             END AS bloat_ratio
            ,is_na
    FROM (
           SELECT  ceil( reltuples / ( (block_size-page_hdr)/tpl_size ) ) +
 ceil( toasttuples / 4 ) AS est_tblpages
                  ,ceil( reltuples / ( (block_size-page_hdr)*fillfactor/
(tpl_size*100) ) ) + ceil( toasttuples / 4 ) AS est_tblpages_ff
                  ,tblpages
                  ,fillfactor
                  ,block_size
                  ,tblid
                  ,schemaname
                  ,tblname
                  ,n_dead_tup
                  ,n_live_tup
```

```
                  ,heappages
                  ,toastpages
                  ,is_na
              FROM (
                     SELECT ( 4 + tpl_hdr_size + tpl_data_size + (2*ma)
                                - CASE WHEN tpl_hdr_size%ma = 0 THEN ma ELSE
 tpl_hdr_size%ma END
                                - CASE WHEN ceil(tpl_data_size)::int%ma = 0 THEN ma ELSE
 ceil(tpl_data_size)::int%ma END
                            ) AS tpl_size
                            ,block_size - page_hdr AS size_per_block
                            ,(heappages + toastpages) AS tblpages
                            ,heappages
                            ,toastpages
                            ,reltuples
                            ,toasttuples
                            ,block_size
                            ,page_hdr
                            ,tblid
                            ,schemaname
                            ,tblname
                            ,fillfactor
                            ,is_na
                            ,n_dead_tup
                            ,n_live_tup
                           FROM (
                                SELECT  tbl.oid                       AS tblid
                                       ,ns.nspname                    AS schemaname
                                       ,tbl.relname                   AS tblname
                                       ,tbl.reltuples                 AS reltuples
                                       ,tbl.relpages                  AS heappages
                                       ,coalesce(toast.relpages, 0)   AS toastpages
                                       ,coalesce(toast.reltuples, 0)  AS toasttuples
                                       ,psat.n_dead_tup               AS n_dead_tup
                                       ,psat.n_live_tup               AS n_live_tup
                                       ,24                            AS page_hdr
                                       ,current_setting('block_size')::numeric AS
 block_size

 ,coalesce(substring( array_to_string(tbl.reloptions, ' ') FROM
 'fillfactor=([0-9]+)')::smallint, 100) AS fillfactor
                                       ,CASE WHEN version()~'mingw32' OR version()~'64-
bit|x86_64|ppc64|ia64|amd64' THEN 8 ELSE 4 END        AS ma
```

```
                                        ,23 + CASE WHEN MAX(coalesce(null_frac,0)) > 0
 THEN ( 7 + count(*) ) / 8 ELSE 0::int END              AS tpl_hdr_size
                                        ,sum( (1-coalesce(s.null_frac, 0)) *
 coalesce(s.avg_width, 1024) )                                  AS tpl_data_size
                                        ,bool_or(att.atttypid =
 'pg_catalog.name'::regtype) OR count(att.attname) <> count(s.attname)         AS is_na
                                   FROM  pg_attribute       AS att
                                   JOIN  pg_class           AS tbl    ON (att.attrelid =
 tbl.oid)
                                   JOIN  pg_stat_all_tables AS psat   ON (tbl.oid =
 psat.relid)
                                   JOIN  pg_namespace       AS ns     ON (ns.oid =
 tbl.relnamespace)
                              LEFT JOIN  pg_stats           AS s      ON
 (s.schemaname=ns.nspname AND s.tablename = tbl.relname AND s.inherited=false AND
 s.attname=att.attname)
                              LEFT JOIN  pg_class           AS toast  ON
 (tbl.reltoastrelid = toast.oid)
                                  WHERE  att.attnum > 0
                                    AND  NOT att.attisdropped
                                    AND  tbl.relkind = 'r'
                               GROUP BY  tbl.oid, ns.nspname, tbl.relname,
 tbl.reltuples, tbl.relpages, toastpages, toasttuples, fillfactor, block_size, ma,
 n_dead_tup, n_live_tup
                               ORDER BY  schemaname, tblname
                            ) AS s
                 ) AS s2
       ) AS s3
 ORDER BY bloat_size DESC
)
  SELECT *
    FROM report
   WHERE bloat_ratio != 0
 -- AND schemaname = 'public'
 -- AND tblname = 'pgbench_accounts'
;

-- WHERE NOT is_na
--   AND tblpages*((pst).free_percent + (pst).dead_tuple_percent)::float4/100 >= 1
```

Puoi verificare l'aumento delle dimensioni della tabella e dell'indice nell'applicazione. Per ulteriori informazioni, consulta Diagnosi delle dimensioni della tabella e dell'indice.

Trova tabelle che consumano spazio inutile

Per trovare tabelle che richiedono spazio inutile, esegui la query riportata.

```
-- WARNING: run with a nonsuperuser role, the query inspects
-- only indexes on tables you have permissions to read.
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
    ) AS est_pages,
    coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
```

```
       + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
MAXALIGN
           WHEN nulldatawidth = 0 THEN 0
           WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
           ELSE nulldatawidth::integer%maxalign
         END
     )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
     -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
   FROM (
     SELECT
       i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
AS table_oid,
       current_setting('block_size')::numeric AS bs, fillfactor,
       CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
         WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
THEN 8
         ELSE 4
       END AS maxalign,
       /* per page header, fixed size: 20 for 7.X, 24 for others */
       24 AS pagehdr,
       /* per page btree opaque data */
       16 AS pageopqdata,
       /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
       CASE WHEN max(coalesce(s.null_frac,0)) = 0
         THEN 2 -- IndexTupleData size
         ELSE 2 + (( 32 + 8 - 1 ) / 8)
         -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
index + 8 - 1 /8)
       END AS index_tuple_hdr_bm,
       /* data len: we remove null values save space using it fractionnal part from
stats */
       sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
nulldatawidth,
       max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
AS is_na
     FROM pg_attribute AS a
       JOIN (
         SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
           idx.reltuples, idx.relpages, idx.relam,
           indrelid, indexrelid, indkey::smallint[] AS attnum,
           coalesce(substring(
             array_to_string(idx.reloptions, ' ')
              from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
         FROM pg_index
```

```
            JOIN pg_class idx ON idx.oid=pg_index.indexrelid
            JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
            JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
          WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
        ) AS i ON a.attrelid = i.indexrelid
        JOIN pg_stats AS s ON s.schemaname = i.nspname
          AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
          -- stats from tbl
          OR  (s.tablename = i.idxname AND s.attname = a.attname))
          -- stats from functionnal cols
        JOIN pg_type AS t ON a.atttypid = t.oid
      WHERE a.attnum > 0
      GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
    ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

Trova tabelle idonee per l'autovacuum

Per trovare tabelle idonee per l'autovacuum, esegui la query riportata.

```
--This query shows tables that need vacuuming and are eligible candidates.
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
```

```
        coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
        as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC;
```

Rispondere a un numero elevato di connessioni

Quando monitori Amazon CloudWatch, potresti scoprire che la `DatabaseConnections` metrica aumenta. Questo aumento indica un numero maggiore di connessioni al database. Consigliamo quanto segue:

- Limita il numero di connessioni che l'applicazione può aprire con ciascuna istanza. Se l'applicazione dispone di una funzione di connection pool incorporata, impostare un numero ragionevole di connessioni. Basa il numero su ciò che vCPUs nella tua istanza può parallelizzare efficacemente.

  Se la tua applicazione non utilizza una funzionalità di pool di connessioni, prendi in considerazione l'utilizzo di Amazon RDS Proxy o un'alternativa. Questo approccio consente all'applicazione di aprire più connessioni con il bilanciamento del carico. Il bilanciatore può quindi aprire un numero limitato di connessioni con il database. Poiché un numero inferiore di connessioni sono in

esecuzione in parallelo, l'istanza DB esegue meno commutazione di contesto nel kernel. Le query dovrebbero progredire più velocemente, causando un minor numero di eventi di attesa. Per ulteriori informazioni, consulta Proxy Amazon RDS per Aurora.

- Quando possibile, sfrutta i nodi di lettura per Aurora SQL Postgre e leggi le repliche per Postgre. RDS SQL Quando l'applicazione esegue un'operazione di sola lettura, inviare queste richieste all'endpoint di sola lettura. Questa tecnica diffonde le richieste delle applicazioni su tutti i nodi del lettore, riducendo la pressione I/O sul nodo di scrittura.
- Prendi in considerazione la possibilità di scalare l'istanza database. Una classe di istanza a maggiore capacità offre più memoria, il che offre ad Aurora Postgre SQL un pool di buffer condiviso più ampio per contenere le pagine. Le dimensioni maggiori offrono inoltre all'istanza DB una maggiore capacità di gestione delle connessioni. vCPUs Altre vCPUs sono particolarmente utili quando le operazioni che generano eventi di `IO:DataFileRead` attesa sono scritture.

IO: XactSync

L'evento `IO:XactSync` si verifica quando un processo di backend è in attesa che il sottosistema di storage Aurora riconosca il commit di una transazione regolare o il commit o il rollback di una transazione preparata. Una transazione preparata fa parte del supporto di PostgreSQL per un commit in due fasi. Questo evento può verificarsi anche quando una query è in attesa del commit di un'altra transazione, in particolare nei casi in cui il commit automatico è disattivato. In tali scenari, potrebbe sembrare che gli aggiornamenti siano in attesa XactSync anche se non sono ancora stati eseguiti.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `IO:XactSync` indica che l'istanza sta dedicando tempo ad aspettare che il sottosistema di storage Aurora confermi che i dati delle transazioni sono stati elaborati.

Probabili cause di aumento delle attese

Quando l'evento `IO:XactSync` si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Saturazione di rete

Il traffico tra i client e l'istanza DB o il traffico verso il sottosistema di storage potrebbe essere troppo pesante per la larghezza di banda della rete.

Pressione della CPU

Un carico di lavoro pesante potrebbe impedire al daemon di archiviazione Aurora di ottenere un tempo sufficiente per la CPU.

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Monitoraggio delle risorse
- Ridimensionamento della CPU
- Aumentare la larghezza di banda di rete
- Riduzione del numero di commit

Monitoraggio delle risorse

Per determinare la causa dell'aumento degli eventi `IO:XactSync`, controlla i seguenti parametri:

- `WriteThroughput` e `CommitThroughput` — Le modifiche del throughput di scrittura o del throughput di commit possono mostrare un aumento del carico di lavoro.
- `WriteLatency` e `CommitLatency` — Le modifiche alla latenza di scrittura o alla latenza di commit possono mostrare che al sottosistema di storage viene chiesto di svolgere più lavoro.

- `CPUUtilization` — Se l'utilizzo della CPU dell'istanza è superiore al 90%, il daemon di archiviazione Aurora potrebbe non avere tempo sufficiente per la CPU. In questo caso, le prestazioni I/O diminuiscono.

Per ulteriori informazioni su questi parametri, consulta Parametri a livello di istanza per Amazon Aurora.

Ridimensionamento della CPU

Per risolvere i problemi di fame della CPU, prendi in considerazione la possibilità di passare a un tipo di istanza con maggiore capacità della CPU. Per informazioni sulla capacità della CPU per una classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Aumentare la larghezza di banda di rete

Per determinare se l'istanza sta raggiungendo i limiti di larghezza di banda di rete, verificare la presenza degli altri eventi di attesa seguenti:

- `IO:DataFileRead`, `IO:BufferRead`, `IO:BufferWrite` e `IO:XactWrite` — Le query che utilizzano grandi quantità di I/O possono generare più di questi eventi di attesa.
- `Client:ClientRead` e `Client:ClientWrite` — Le query con grandi quantità di comunicazioni client possono generare più di questi eventi di attesa.

Se la larghezza di banda di rete è un problema, considera la possibilità di passare a un tipo di istanza con maggiore larghezza di banda di rete. Per informazioni sulle prestazioni di rete per una classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Riduzione del numero di commit

Per ridurre il numero di commit, combinare le dichiarazioni in blocchi di transazione.

IPC:DamRecordTxAck

L'evento `IPC:DamRecordTxAck` si verifica quando Aurora PostgreSQL in una sessione che utilizza flussi di attività del database genera un evento di flusso di attività, quindi attende che tale evento diventi duraturo.

Argomenti

- Versioni di motori pertinenti

- Context
- Cause
- Azioni

Versioni di motori pertinenti

Queste informazioni sugli eventi di attesa sono rilevanti per tutte le versioni di Aurora PostgreSQL 10.7 e successive 10, 11.4 e versioni successive 11 e tutte le versioni 12 e 13.

Context

In modalità sincrona, la durata degli eventi del flusso di attività è favorita rispetto alle prestazioni del database. In attesa di una scrittura duratura dell'evento, la sessione blocca le altre attività del database, causando l'evento di attesa `IPC:DamRecordTxAck`.

Cause

La causa più comune per l'evento `IPC:DamRecordTxAck` che appare nelle prime attese è che la funzione Database Activity Streams (DAS) è un audit olistico. L'attività SQL superiore genera eventi di flusso di attività che devono essere registrati.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa:

- Riduci il numero di istruzioni SQL o disattiva i flussi di attività del database. In questo modo, è possibile ridurre il numero di eventi che richiedono scritture durevoli.
- Passa alla modalità asincrona. Ciò aiuta a ridurre le contese sull'evento di attesa `IPC:DamRecordTxAck`.

  Tuttavia, la funzione DAS non può garantire la durata di ogni evento in modalità asincrona.

IPC: eventi di attesa paralleli

Quanto segue `IPC:parallel wait events` indica che una sessione è in attesa della comunicazione tra processi relativa alle operazioni di esecuzione di query parallele.

- `IPC:BgWorkerStartup`- Un processo è in attesa che un processo di lavoro parallelo completi la sequenza di avvio. Ciò accade quando si inizializzano i worker per l'esecuzione di query parallele.

- `IPC:BgWorkerShutdown`- Un processo è in attesa che un processo di lavoro parallelo completi la sequenza di spegnimento. Ciò si verifica durante la fase di pulizia dell'esecuzione di query parallele.
- `IPC:ExecuteGather`- Un processo è in attesa di ricevere dati dai processi di lavoro paralleli durante l'esecuzione delle query. Ciò si verifica quando il processo leader deve raccogliere risultati dai suoi lavoratori.
- `IPC:ParallelFinish`- Un processo è in attesa che i lavoratori paralleli completino l'esecuzione e riportino i risultati finali. Ciò avviene durante la fase di completamento dell'esecuzione delle query parallele.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'esecuzione parallela delle query in PostgreSQL implica la collaborazione di più processi per elaborare una singola query. Quando si determina che un'interrogazione è adatta alla parallelizzazione, un processo leader si coordina con uno o più processi di lavoro paralleli in base all'`max_parallel_workers_per_gather`impostazione dei parametri. Il processo leader divide il lavoro tra i lavoratori, ogni lavoratore elabora la propria parte di dati in modo indipendente e i risultati vengono raccolti nel processo principale.

> **Note**
>
> Ogni worker parallelo opera come un processo separato con requisiti di risorse simili a quelli di una sessione utente completa. Ciò significa che una query parallela con 4 worker può consumare fino a 5 volte le risorse (CPU, memoria, I/O larghezza di banda) rispetto a una

query non parallela, poiché sia il processo leader che ogni processo di lavoro mantengono le proprie allocazioni di risorse. Ad esempio, impostazioni come quelle `work_mem` vengono applicate singolarmente a ciascun lavoratore, moltiplicando potenzialmente l'utilizzo totale della memoria in tutti i processi.

L'architettura di interrogazione parallela è composta da tre componenti principali:

- Processo leader: il processo principale che avvia l'operazione parallela, divide il carico di lavoro e si coordina con i processi di lavoro.
- Processi di lavoro: processi in background che eseguono parti della query in parallelo.
- Gather/Gather merge: operazioni che combinano i risultati di più processi di lavoro e li restituiscono al leader

Durante l'esecuzione parallela, i processi devono comunicare tra loro tramite meccanismi di comunicazione tra processi (IPC). Questi eventi di attesa IPC si verificano durante diverse fasi:

- Avvio del lavoratore: quando i worker paralleli vengono inizializzati
- Scambio di dati: quando i lavoratori elaborano i dati e inviano i risultati al leader
- Arresto del lavoratore: quando l'esecuzione parallela viene completata e i lavoratori vengono terminati
- Punti di sincronizzazione: quando i processi devono coordinarsi o attendere che altri processi completino le proprie attività

La comprensione di questi eventi di attesa è fondamentale per diagnosticare i problemi di prestazioni relativi all'esecuzione di query parallele, specialmente in ambienti ad alta concorrenza in cui possono essere eseguite più query parallele contemporaneamente.

Probabili cause di aumento delle attese

Diversi fattori possono contribuire a un aumento degli eventi di attesa IPC correlati al parallelo:

Elevata concorrenza di query parallele

L'esecuzione simultanea di più query parallele può portare a un conflitto di risorse e a un aumento dei tempi di attesa per le operazioni IPC. Ciò è particolarmente comune nei sistemi con elevati volumi di transazioni o carichi di lavoro analitici.

Piani di interrogazione parallela non ottimali

Se il pianificatore di query sceglie piani paralleli inefficienti, ciò può comportare una parallelizzazione non necessaria o una scarsa distribuzione del lavoro tra i lavoratori. Ciò può portare a un aumento delle attese IPC, in particolare per gli eventi IPC: e IPC:. ExecuteGather ParallelFinish Questi problemi di pianificazione spesso derivano da statistiche obsolete e dal rigonfiamento. table/index

Avvio e arresto frequenti dei lavoratori paralleli

Le interrogazioni di breve durata che avviano e interrompono frequentemente i parallel worker possono causare un aumento degli eventi AND. `IPC:BgWorkerStartup` `IPC:BgWorkerShutdown` Ciò si verifica spesso nei carichi di lavoro OLTP con molte piccole query parallelizzabili.

Vincoli delle risorse

CPU, memoria o I/O capacità limitate possono causare colli di bottiglia nell'esecuzione parallela, con conseguente aumento dei tempi di attesa in tutti gli eventi IPC. Ad esempio, se la CPU è satura, i processi di lavoro potrebbero impiegare più tempo per avviarsi o elaborare la relativa parte di lavoro.

Strutture di interrogazione complesse

Le interrogazioni con più livelli di parallelismo (ad esempio, join paralleli seguiti da aggregazioni parallele) possono portare a modelli IPC più complessi e tempi di attesa potenzialmente maggiori, soprattutto per gli eventi. `IPC:ExecuteGather`

Serie di risultati di grandi dimensioni

Le interrogazioni che producono set di risultati di grandi dimensioni possono aumentare i tempi di `IPC:ExecuteGather` attesa poiché il processo leader dedica più tempo alla raccolta e all'elaborazione dei risultati dei processi di lavoro.

La comprensione di questi fattori può aiutare a diagnosticare e risolvere i problemi di prestazioni relativi all'esecuzione di query parallele in Aurora PostgreSQL.

Azioni

Quando si vedono attese relative alle interrogazioni parallele, in genere significa che un processo di backend sta coordinando o è in attesa dei processi di lavoro paralleli. Queste attese sono comuni

durante l'esecuzione di piani paralleli. È possibile analizzare e mitigare l'impatto di queste attese monitorando l'utilizzo dei lavoratori paralleli, rivedendo le impostazioni dei parametri e ottimizzando l'esecuzione delle query e l'allocazione delle risorse.

Argomenti

- Analizza i piani di interrogazione per verificare la presenza di un parallelismo inefficiente
- Monitora l'utilizzo delle query parallele
- Rivedi e modifica le impostazioni delle interrogazioni parallele
- Ottimizza l'allocazione delle risorse
- Esamina la gestione delle connessioni
- Rivedi e ottimizza le operazioni di manutenzione
- Utilizza Query Plan Management (QPM)

Analizza i piani di interrogazione per verificare la presenza di un parallelismo inefficiente

L'esecuzione parallela delle query può spesso portare a instabilità del sistema, picchi di CPU e variazioni imprevedibili delle prestazioni delle query. È fondamentale analizzare a fondo se il parallelismo migliora effettivamente il carico di lavoro specifico. Usa EXPLAIN ANALYZE per esaminare i piani di esecuzione delle query parallele.

Disattiva temporaneamente il parallelismo a livello di sessione per confrontare l'efficienza del piano:

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

Riattiva il parallelismo e confronta:

```
RESET max_parallel_workers_per_gather;
EXPLAIN ANALYZE <your_query>;
```

Se la disabilitazione del parallelismo produce risultati migliori o più coerenti, prendi in considerazione la possibilità di disabilitarlo per query specifiche a livello di sessione utilizzando i comandi SET. Per un impatto più ampio, potresti voler disabilitare il parallelismo a livello di istanza regolando i parametri

pertinenti nel cluster o nel gruppo di parametri di istanza. Per ulteriori informazioni, consulta Amazon Aurora PostgreSQL parametri.

Monitora l'utilizzo delle query parallele

Utilizza le seguenti query per ottenere visibilità sull'attività e sulla capacità delle query parallele:

Controlla i processi di lavoro paralleli attivi:

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
    backend_type = 'parallel worker';
```

Questa query mostra il numero di processi di lavoro paralleli attivi. Un valore elevato può indicare che il tuo `max_parallel_workers` è configurato con un valore elevato e potresti prendere in considerazione la possibilità di ridurlo.

Controlla le query parallele simultanee:

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

Questa query restituisce il numero di processi leader distinti che hanno avviato query parallele. Un numero elevato indica che più sessioni eseguono query parallele contemporaneamente, il che può aumentare la richiesta di CPU e memoria.

Rivedi e modifica le impostazioni delle interrogazioni parallele

Controlla i seguenti parametri per assicurarti che siano in linea con il tuo carico di lavoro:

- `max_parallel_workers`: numero totale di worker paralleli in tutte le sessioni.

- `max_parallel_workers_per_gather`: numero massimo di lavoratori per query.

Per i carichi di lavoro OLAP, l'aumento di questi valori può migliorare le prestazioni. Per i carichi di lavoro OLTP, in genere sono preferiti valori inferiori.

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

Ottimizza l'allocazione delle risorse

Monitora l'utilizzo della CPU e valuta la possibilità di regolare il numero di v CPUs se è costantemente elevato e se l'applicazione trae vantaggio dalle query parallele. Assicurarsi che sia disponibile una memoria adeguata per le operazioni parallele.

- Utilizza le metriche di Performance Insights per determinare se il sistema è legato alla CPU.
- Ogni worker parallelo utilizza il proprio`work_mem`. Assicurati che l'utilizzo totale della memoria rientri nei limiti delle istanze.

Le interrogazioni parallele possono consumare molte più risorse rispetto alle interrogazioni non parallele, poiché ogni processo di lavoro è un processo completamente separato che ha all'incirca lo stesso impatto sul sistema di una sessione utente aggiuntiva. Questo deve essere tenuto in considerazione quando si sceglie un valore per questa impostazione, nonché quando si configurano altre impostazioni che controllano l'utilizzo delle risorse, ad esempio. `work_mem` Per ulteriori informazioni, consultare la documentazione di PostgreSQL. I limiti delle risorse, ad esempio, `work_mem` vengono applicati individualmente a ciascun lavoratore, il che significa che l'utilizzo totale può essere molto più elevato in tutti i processi rispetto a quanto sarebbe normalmente per ogni singolo processo.

Prendi in considerazione l'aumento di v CPUs o l'ottimizzazione dei parametri di memoria se il carico di lavoro è fortemente parallelizzato.

Esamina la gestione delle connessioni

In caso di esaurimento della connessione, esaminate le strategie di pool di connessioni delle applicazioni. Prendi in considerazione l'implementazione del pool di connessioni a livello di applicazione, se non è già in uso.

Rivedi e ottimizza le operazioni di manutenzione

Coordina la creazione degli indici e le altre attività di manutenzione per prevenire il conflitto di risorse. Valuta la possibilità di pianificare queste operazioni durante le ore non di punta. Evita di pianificare interventi di manutenzione impegnativi (ad esempio, compilazioni di indici paralleli) durante i periodi di elevato carico di query da parte degli utenti. Queste operazioni possono consumare lavoratori paralleli e influire sulle prestazioni per le query regolari.

Utilizza Query Plan Management (QPM)

In Aurora PostgreSQL, la funzionalità Query Plan Management (QPM) è progettata per garantire l'adattabilità e la stabilità del piano, indipendentemente dalle modifiche all'ambiente del database che potrebbero causare la regressione del piano di query. Per ulteriori informazioni, consulta Panoramica della gestione del piano di query per Aurora PostgreSQL .QPM fornisce un certo controllo sull'ottimizzatore. Rivedi i piani approvati in QPM per assicurarti che siano in linea con le attuali impostazioni di parallelismo. Aggiorna o rimuovi i piani obsoleti che potrebbero forzare un'esecuzione parallela non ottimale.

Puoi anche correggere i piani usando pg_hint_plan. Per ulteriori informazioni, consulta Correzione dei piani mediante pg_hint_plan. È possibile utilizzare il suggerimento denominato `Parallel` per imporre l'esecuzione parallela. Per ulteriori informazioni, consulta i Suggerimenti per i piani paralleli.

Lock:advisory

L'evento `Lock:advisory` si verifica quando un'applicazione PostgreSQL utilizza un blocco per coordinare l'attività su più sessioni.

Argomenti

- Versioni di motori pertinenti
- Context
- Cause
- Operazioni

Versioni di motori pertinenti

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

I blocchi di consulenza PostgreSQL sono blocchi cooperativi a livello di applicazione esplicitamente bloccati e sbloccati dal codice dell'applicazione dell'utente. Un'applicazione PostgreSQL può utilizzare un blocco per coordinare l'attività su più sessioni. A differenza dei normali blocchi a livello di oggetto o riga, l'applicazione ha il pieno controllo sulla durata del blocco. Per ulteriori informazioni consulta Blocchi di consulenza nella documentazione di PostgreSQL.

I blocchi di consulenza possono essere rilasciati prima della fine di una transazione o essere trattenuti da una sessione tra le transazioni. Ciò tuttavia non è vero per i blocchi impliciti e applicati al sistema, come un blocco esclusivo di accesso su una tabella acquisita da una dichiarazione `CREATE INDEX`.

Per una descrizione delle funzioni utilizzate per acquisire (bloccare) e rilasciare (sbloccare) i blocchi di consulenza, vedere Funzioni di Advisory Lock nella documentazione di PostgreSQL.

I blocchi di consulenza sono implementati sopra il normale sistema di blocco PostgreSQL e sono visibili nella visualizzazione di sistema `pg_locks`.

Cause

Questo tipo di blocco è controllato esclusivamente da un'applicazione che lo utilizza esplicitamente. I blocchi di consulenza acquisiti per ogni riga come parte di una query possono causare un picco di blocchi o un accumulo a lungo termine.

Questi effetti si verificano quando la query viene eseguita in un modo che acquisisce blocchi su più righe di quelle restituite dalla query. L'applicazione dovrà comunque rilasciare ogni blocco, ma se i blocchi vengono acquisiti su righe che non vengono restituite, l'applicazione non riesce a trovare tutti i blocchi.

L'esempio seguente è tratto da Blocchi di consulenza nella documentazione di PostgreSQL.

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

In questo esempio, la clausola `LIMIT` può arrestare l'output della query solo dopo che le righe sono già state selezionate internamente e i relativi valori ID bloccati. Ciò può accadere improvvisamente quando un volume di dati crescente fa sì che il pianificatore scelga un piano di esecuzione diverso che non è stato testato durante lo sviluppo. L'accumulo in questo caso avviene perché l'applicazione chiama esplicitamente `pg_advisory_unlock` per ogni valore ID bloccato. Tuttavia, in questo

caso non è possibile trovare il set di blocchi acquisiti su righe che non sono state restituite. Poiché i blocchi vengono acquisiti a livello di sessione, non vengono rilasciati automaticamente alla fine della transazione.

Un'altra possibile causa di picchi nei tentativi di blocco bloccati sono i conflitti non intenzionali. In questi conflitti, parti non correlate dell'applicazione condividono per errore lo stesso spazio ID di blocco.

Operazioni

Esaminare l'utilizzo delle applicazioni dei blocchi di consulenza e i dettagli su dove e quando nel flusso dell'applicazione viene acquisito e rilasciato ogni tipo di blocco consultivo.

Determina se una sessione sta acquisendo troppi blocchi o che una sessione di lunga durata non rilascia blocchi abbastanza presto, causando un lento accumulo di blocchi. È possibile correggere un lento accumulo di blocchi a livello di sessione terminando la sessione utilizzando `pg_terminate_backend(pid)`.

Viene visualizzato un client in attesa di un blocco di avviso in `pg_stat_activity` con `wait_event_type=Lock` e `wait_event=advisory`. È possibile ottenere valori di blocco specifici eseguendo una query nella vista di sistema `pg_locks` per lo stesso `pid`, cercando `locktype=advisory` e `granted=f`.

È quindi possibile identificare la sessione di blocco interrogando `pg_locks` per lo stesso blocco consultivo `granted=t`, come mostrato nell'esempio seguente.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
```

```
         blocking_activity.query AS blocking_statement
   FROM pg_catalog.pg_locks blocked_locks
   JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
blocked_locks.pid
   JOIN pg_catalog.pg_locks blocking_locks
       ON blocking_locks.locktype = blocked_locks.locktype
       AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
       AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
       AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
       AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
       AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
       AND blocking_locks.transactionid IS NOT DISTINCT FROM
blocked_locks.transactionid
       AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
       AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
       AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
       AND blocking_locks.pid != blocked_locks.pid
   JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
blocking_locks.pid
   WHERE NOT blocked_locks.GRANTED;
```

Tutte le funzioni API di blocco consultivo hanno due serie di argomenti, un argomento `bigint` o due argomenti `integer`:

- Per le funzioni API con un argomento `bigint`, i 32 bit superiori sono in `pg_locks.classid` e i 32 bit inferiori sono in `pg_locks.objid`.
- Per le funzioni API con due argomenti `integer`, il primo argomento è `pg_locks.classid` e il secondo argomento è `pg_locks.objid`.

Il valore `pg_locks.objsubid` indica quale modulo API è stato utilizzato: 1 significa un argomento `bigint`; 2 significa due argomenti `integer`.

Lock:extend

L'evento `Lock:extend` si verifica quando un processo di back-end è in attesa di bloccare una relazione per estenderla mentre un altro processo ha un blocco su tale relazione per lo stesso scopo.

Argomenti

- Versioni del motore supportate
- Context

- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `Lock:extend` indica che un processo di back-end è in attesa di estendere una relazione su cui un altro processo di backend mantiene un blocco mentre sta estendendo tale relazione. Poiché solo un processo alla volta può estendere una relazione, il sistema genera un evento di attesa `Lock:extend`. Le operazioni `INSERT`, `COPY`, e `UPDATE` possono generare questo evento.

Probabili cause di aumento delle attese

Quando l'evento `Lock:extend` si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Aumento degli inserti simultanei o degli aggiornamenti della stessa tabella

Potrebbe esserci un aumento del numero di sessioni simultanee con query che inseriscono o aggiornano la stessa tabella.

Larghezza di banda di rete insufficiente

La larghezza di banda di rete sull'istanza database potrebbe essere insufficiente per le esigenze di comunicazione di storage del carico di lavoro corrente. Ciò può contribuire alla latenza dello storage che causa un aumento degli eventi `Lock:extend`.

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Riduci gli inserti e gli aggiornamenti simultanei alla stessa relazione
- Aumentare la larghezza di banda di rete

Riduci gli inserti e gli aggiornamenti simultanei alla stessa relazione

Innanzitutto, determinare se c'è un aumento dei parametri `tup_inserted` e `tup_updated` e un aumento di questi eventi di attesa. In tal caso, verificare quali relazioni sono in forte contesa per le operazioni di inserimento e aggiornamento. Per determinarlo, interrogare la vista `pg_stat_all_tables` per i valori nei campi `n_tup_ins` e `n_tup_upd`. Per ulteriori informazioni sulla vista `pg_stat_all_tables`, consultare pg_stat_all_tables nella documentazione PostgreSQL.

Per ottenere ulteriori informazioni sul blocco e le query bloccate, eseguire una query `pg_stat_activity` come nel seguente esempio:

```
SELECT
    blocked.pid,
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+---------------------+--------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Dopo aver identificato le relazioni che contribuiscono ad aumentare gli eventi `Lock:extend`, utilizza le seguenti tecniche per ridurre la contesa:

- Scopri se è possibile utilizzare il partizionamento per ridurre le contese per la stessa tabella. La separazione delle tuple inserite o aggiornate in diverse partizioni può ridurre le contese. Per informazioni sulle partizioni, consulta Gestione delle partizioni PostgreSQL con l'estensione pg_partman.

- Se l'evento di attesa è dovuto principalmente all'attività di aggiornamento, considerare di ridurre il valore del fattore di riempimento della relazione. Ciò può ridurre le richieste di nuovi blocchi durante l'aggiornamento. Il fattore di riempimento è un parametro di archiviazione per una tabella che determina la quantità massima di spazio per l'imballaggio di una pagina di tabella. Viene espresso come percentuale dello spazio totale per una pagina. Per ulteriori informazioni sul parametro fillfactor, consulta CREA TABELLA nella documentazione di PostgreSQL.

> **Important**
>
> Si consiglia vivamente di testare il sistema se si modifica il fattore di riempimento perché la modifica di questo valore può influire negativamente sulle prestazioni, a seconda del carico di lavoro.

Aumentare la larghezza di banda di rete

Per vedere se c'è un aumento della latenza di scrittura, controlla il parametro `WriteLatency` in CloudWatch. Se è presente, usa i parametri Amazon CloudWatch `WriteThroughput` e `ReadThroughput` per monitorare il traffico relativo allo storage sul cluster DB. Questi parametri possono aiutarti a determinare se la larghezza di banda della rete è sufficiente per il tuo carico di lavoro.

Se la larghezza di banda della rete non è sufficiente, aumentala. Se il file client o l'istanza DB sta raggiungendo i limiti di larghezza di banda di rete, l'unico modo per aumentare la larghezza di banda è aumentare la dimensione dell'istanza DB.

Per ulteriori informazioni sui parametri di CloudWatch, consultare CloudWatch Parametri Amazon per Amazon Aurora. Per informazioni sulle prestazioni di rete per una classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Lock:Relation

L'evento `Lock:Relation` si verifica quando una query è in attesa di acquisire un blocco su una tabella o vista (relazione) attualmente bloccata da un'altra transazione.

Argomenti

- Versioni del motore supportate
- Context

- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni sugli eventi di attesa sono supportate per tutte le versioni di Aurora Postgre. SQL

Context

La maggior parte dei SQL comandi Postgre utilizza implicitamente i blocchi per controllare l'accesso simultaneo ai dati nelle tabelle. È inoltre possibile utilizzare questi blocchi esplicitamente nel codice dell'applicazione con il comando `LOCK`. Molte modalità di blocco non sono compatibili tra loro e possono bloccare le transazioni quando cercano di accedere allo stesso oggetto. Quando ciò accade, Aurora Postgre SQL genera un evento. `Lock:Relation` Di seguito sono riportati alcuni esempi comuni:

- Blocchi esclusivi come `ACCESS EXCLUSIVE` possono bloccare tutti gli accessi simultanei. Operazioni Data Definition Language (DDL) come`DROP TABLE`, `TRUNCATEVACUUM FULL`, e `CLUSTER` acquisisci `ACCESS EXCLUSIVE` blocchi in modo implicito. `ACCESS EXCLUSIVE`è anche la modalità di blocco predefinita per `LOCK TABLE` le istruzioni che non specificano una modalità in modo esplicito.
- L'utilizzo `CREATE INDEX (without CONCURRENT)` su una tabella è in conflitto con le istruzioni Data Manipulation Language (DML) e `UPDATE DELETEINSERT`, che acquisiscono `ROW EXCLUSIVE` blocchi.

Per ulteriori informazioni sui blocchi a livello di tabella e sulle modalità di blocco in conflitto, consulta Explicit Locking nella documentazione di Postgre. SQL

Il blocco di query e transazioni in genere si sblocca in uno dei seguenti modi:

- Query di blocco: l'applicazione può annullare la query o l'utente può terminare il processo. Il motore può anche forzare la fine della query a causa del timeout dell'istruzione di una sessione o di un meccanismo di rilevamento del deadlock.
- Blocco della transazione: una transazione smette di bloccarsi quando esegue un'istruzione `ROLLBACK` o `COMMIT`. I rollback si verificano automaticamente anche quando le sessioni vengono disconnesse da un client o da problemi di rete o terminano. Le sessioni possono essere terminate quando il motore di database è spento, quando il sistema è fuori memoria e così via.

Probabili cause di aumento delle attese

Quando l'evento `Lock:Relation` si verifica più frequentemente del normale, può indicare un problema di prestazioni. Le cause tipiche sono:

Sessioni simultanee aumentate con blocchi di tabella in conflitto

Potrebbe esserci un aumento del numero di sessioni simultanee con query che inseriscono o aggiornano la stessa tabella.

Operazioni di manutenzione

Le operazioni di manutenzione Health come `VACUUM` e `ANALYZE` possono aumentare significativamente il numero di blocchi in conflitto. `VACUUM FULL` acquisisce un blocco `ACCESS EXCLUSIVE`, e `ANALYZE` acquisisce un blocco `SHARE UPDATE EXCLUSIVE`. Entrambi i tipi di blocchi possono causare un evento di attesa `Lock:Relation`. Le operazioni di manutenzione dei dati delle applicazioni, come l'aggiornamento di una vista materializzata, possono anche aumentare le query e le transazioni bloccate.

Blocchi sulle istanze del lettore

È possibile che si verifichi un conflitto tra i blocchi di relazione tenuti dallo scrittore e dai lettori. Attualmente solo i blocchi di relazione `ACCESS EXCLUSIVE` vengono replicati nelle istanze del lettore. Tuttavia, il blocco di relazione `ACCESS EXCLUSIVE` sarà in conflitto con qualsiasi blocco di relazione `ACCESS SHARE` tenuto dal lettore. Questo conflitto può causare un aumento degli eventi di attesa della relazione di blocco sul lettore.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- SQLRiduci l'impatto delle istruzioni di blocco
- Riduci al minimo l'effetto delle operazioni di manutenzione
- Verifica la presenza di blocchi del lettore

SQLRiduci l'impatto delle istruzioni di blocco

Per ridurre l'impatto delle SQL istruzioni di blocco, modificate il codice dell'applicazione ove possibile. Di seguito sono riportate due tecniche comuni per ridurre i blocchi:

- Usa l'`NOWAIT`opzione: alcuni SQL comandi, come `SELECT` e `LOCK` istruzioni, supportano questa opzione. La direttiva `NOWAIT` annulla la query richiedente il blocco se il blocco non può essere acquisito immediatamente. Questa tecnica può aiutare a impedire che una sessione di blocco provochi un accumulo di sessioni bloccate dietro di essa.

  Ad esempio: si supponga che la transazione A sia in attesa di un blocco trattenuto dalla transazione B. Ora, se B richiede un blocco su una tabella bloccata dalla transazione C, la transazione A potrebbe essere bloccata fino al completamento della transazione C. Ma se la transazione B utilizza un `NOWAIT` quando richiede il blocco su C, può fallire rapidamente e garantire che la transazione A non debba attendere indefinitamente.
- Usa`SET lock_timeout`: imposta un `lock_timeout` valore per limitare il tempo di attesa di un'SQListruzione per acquisire un blocco su una relazione. Se il blocco non viene acquisito entro il timeout specificato, la transazione che richiede il blocco viene annullata. Impostare questo valore a livello di sessione.

Riduci al minimo l'effetto delle operazioni di manutenzione

Operazioni di manutenzione come `VACUUM` e `ANALYZE` sono importanti. Si consiglia di non spegnerli qualora vengano trovati eventi di attesa `Lock:Relation` relativi a queste operazioni di manutenzione. I seguenti approcci possono ridurre al minimo l'effetto di queste operazioni:

- Eseguire manualmente le operazioni di manutenzione durante le ore non di punta.
- Per ridurre le attese `Lock:Relation` causate da attività autovacuum, eseguire qualsiasi sintonizzazione automatica necessaria. Per informazioni sulla regolazione dell'autovacuum, consulta Working with Postgre SQL autovacuum on Amazon nella Amazon User Guide. RDS RDS

Verifica la presenza di blocchi del lettore

Puoi vedere come le sessioni simultanee su uno scrittore e sui lettori potrebbero tenere blocchi che si bloccano a vicenda. A questo scopo, puoi eseguire le query che restituiscono il tipo di blocco e la relazione. Nella tabella, puoi trovare una sequenza di interrogazioni relative a due di queste sessioni simultanee, una sessione di scrittura e una sessione di lettura.

Il processo di riproduzione attende la durata di 1 `max_standby_streaming_delay` prima di annullare la query del lettore. Come mostrato nell'esempio, il timeout di blocco di 100 ms è ben al di sotto del valore di default di `max_standby_streaming_delay` di 30 secondi. Il timeout di blocco si verifica prima che si verifichi un problema.

Evento di sequenza	Sessione	Comando o output

Imposta una variabile di ambiente chiamata READER con il valore specificato e tenta di connettersi all'istanza DB con questo endpoint.	Sessione lettore	CLIcomando:
		`export READER=au rorapg2.1234567891 0.us-west-1.rds.am azonaws.com`
		`psql -h $READER`
		Output:
		`psql (15devel, server 10.14) Type "help" for help.`

Imposta una variabile di ambiente chiamata WRITER e tenta di connettersi all'istanza DB con questo endpoint.	Sessione di scrittura	CLIcomando:
		`export WRITER=au rorapg1.1234567891 0.us-west-1.rds.am azonaws.com psql -h $WRITER`
		Output:
		`psql (15devel, server 10.14) Type "help" for help.`

La sessione writer crea la tabella t1 sull'istanza writer.	Sessione di scrittura	Interrogazione Postger: SQL `postgres=> CREATE TABLE t1(b integer); CREATE TABLE`
Se non ci sono domande contrastanti sullo scrittore, il	Sessione di scrittura	`ACCESS EXCLUSIVE` blocco abilitato

Evento di sequenza	Sessione	Comando o output
ACCESS EXCLUSIVE blocco viene acquisito immediata mente dallo scrittore.		
La sessione del lettore imposta un intervallo di timeout di blocco di 100 millisecondi.	Sessione lettore	Interrogazione Postgree: SQL `postgres=> SET lock_timeout=100; SET`

La sessione di lettura tenta di leggere i dati dalla tabella t1 sull'istanza del lettore.	Sessione lettore	Interrogazione Postgree: SQL
		`postgres=> SELECT * FROM t1;`
		Output di esempio:
		`b --- (0 rows)`

La sessione di scrittura elimina t1.	Sessione di scrittura	Interrogazione Postgres: SQL `postgres=> BEGIN; BEGIN postgres=> DROP TABLE t1; DROP TABLE postgres=>`

Evento di sequenza	Sessione	Comando o output

La query scade e viene annullata sul lettore.	Sessione lettore	Interrogazione SQL Postgre:
		`postgres=>  SELECT * FROM t1;`
		Output di esempio:
		`ERROR:  canceling statement due to lock timeout`
		`LINE 1: SELECT * FROM t1;`
		`^`

Per determinare la causa dell'errore. la sessione di lettura interroga e `pg_locks` `pg_stat_activity`	Sessione lettore	Interrogazione Postgree: SQL
		`postgres=>  SELECT locktype, relation, mode, backend_type`
		`postgres=>  FROM pg_locks l, pg_stat_activity t1`
		`postgres=>  WHERE l.pid=t1.pid AND relation = 't1'::regclass::oid;`

Evento di sequenza	Sessione	Comando o output

Il risultato indica che il `aurora wal replay` processo sta `ACCESS EXCLUSIVE` bloccando la tabella t1.	Sessione lettore	Risultato della query:
		`locktype | relation | mode | backend_type`
		`----------+----------+------------------+---------------`
		`relation | 68628525 | AccessExclusiveLock | aurora wal replay`
		`(1 row)`

Lock:transactionid

L'evento `Lock:transactionid` si verifica quando una transazione è in attesa di un blocco a livello di riga.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `Lock:transactionid` si verifica quando una transazione sta tentando di acquisire un blocco a livello di riga già concesso a una transazione in esecuzione contemporaneamente. La sessione che mostra il l'evento di attesa `Lock:transactionid` è bloccato a causa di questo

blocco. Dopo che la transazione di blocco termina in un'istruzione `COMMIT` o `ROLLBACK`, la transazione bloccata può procedere.

La semantica multiversione di controllo della concorrenza di Aurora PostgreSQL garantisce che i lettori non bloccano gli scrittori e gli scrittori non bloccano i lettori. Affinché si verifichino conflitti a livello di riga, le transazioni bloccate e bloccate devono emettere dichiarazioni in conflitto dei seguenti tipi:

- `UPDATE`
- `SELECT … FOR UPDATE`
- `SELECT … FOR KEY SHARE`

L'istruzione `SELECT … FOR KEY SHARE` è un caso speciale. Il database utilizza la clausola `FOR KEY SHARE` per ottimizzare le prestazioni dell'integrità referenziale. Un blocco a livello di riga su una fila può bloccare i comandi `INSERT`, `UPDATE`, e `DELETE` su altre tabelle che fanno riferimento alla riga.

Probabili cause di aumento delle attese

Quando questo evento appare più del normale, la causa è in genere un'istruzione `UPDATE`, `SELECT … FOR UPDATE`, oppure `SELECT … FOR KEY SHARE` combinate con le seguenti condizioni.

Argomenti

- Elevata concorrenza
- Inattivo in transazione
- Transazioni di lunga durata

Elevata concorrenza

Aurora PostgreSQL può utilizzare la semantica di blocco granulare a livello di riga. La probabilità di conflitti a livello di riga aumenta quando vengono soddisfatte le seguenti condizioni:

- Un carico di lavoro altamente simultaneo è conteso per le stesse righe.
- Aumenta la concorrenza.

Inattivo in transazione

A volte la colonna `pg_stat_activity.state` mostra il valore `idle in transaction`. Questo valore viene visualizzato per le sessioni che hanno avviato una transazione, ma non hanno ancora emesso un `COMMIT` o `ROLLBACK`. Se il valore `pg_stat_activity.state` non è `active`, la query mostrata in `pg_stat_activity` è la versione più recente a terminare l'esecuzione. La sessione di blocco non sta elaborando attivamente una query perché una transazione aperta contiene un blocco.

Se una transazione inattiva ha acquisito un blocco a livello di riga, potrebbe impedire ad altre sessioni di acquisirlo. Questa condizione porta al frequente verificarsi dell'evento di attesa `Lock:transactionid`. Per diagnosticare il problema, esaminare l'output da `pg_stat_activity` e `pg_locks`.

Transazioni di lunga durata

Le transazioni che vengono eseguite a lungo ricevono blocchi per un lungo periodo di tempo. Questi blocchi a tenuta lunga possono impedire l'esecuzione di altre transazioni.

Operazioni

Il blocco delle righe è un conflitto tra le istruzioni `UPDATE`, `SELECT … FOR UPDATE`, oppure `SELECT … FOR KEY SHARE`. Prima di tentare una soluzione, scopri quando queste istruzioni sono in esecuzione sulla stessa riga. Utilizzare queste informazioni per scegliere una strategia descritta nelle sezioni seguenti.

Argomenti

- Rispondere a un'elevata concorrenza
- Rispondere alle transazioni inattive
- Rispondere alle transazioni di lunga durata

Rispondere a un'elevata concorrenza

Se il problema è la concorrenza, prova una delle seguenti tecniche:

- Riduci la concorrenza nell'applicazione. Ad esempio, diminuisci il numero di sessioni attive.
- Implementa un pool di connessioni. Per informazioni su come mettere in pool le connessioni con RDS Proxy, vedere Proxy Amazon RDS per Aurora.

- Progettare l'applicazione o il modello di dati per evitare di contendere le istruzioni `UPDATE` e `SELECT … FOR UPDATE`. È inoltre possibile ridurre il numero di chiavi esterne a cui accedono le istruzioni `SELECT … FOR KEY SHARE`.

Rispondere alle transazioni inattive

Se `pg_stat_activity.state` mostra `idle in transaction`, utilizza le seguenti strategie:

- Attiva autocommit laddove possibile. Questo approccio impedisce alle transazioni di bloccare altre transazioni durante l'attesa di un `COMMIT` o `ROLLBACK`.
- Cerca percorsi di codice a cui manca `COMMIT`, `ROLLBACK`, oppure `END`.
- Assicurati che la logica di gestione delle eccezioni nell'applicazione abbia sempre un percorso per un `end of transaction` valido.
- Assicurati che l'applicazione elabori i risultati delle query dopo aver terminato la transazione con `COMMIT` o `ROLLBACK`.

Rispondere alle transazioni di lunga durata

Se le transazioni di lunga durata causano il frequente verificarsi di `Lock:transactionid`, prova le seguenti strategie:

- Tieni i blocchi di riga fuori dalle transazioni di lunga durata.
- Limita la durata delle query implementando autocommit quando possibile.

Lock:tuple

L'evento `Lock:tuple` si verifica quando un processo di backend aspetta di acquisire un blocco su una tupla.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

L'evento `Lock:tuple` indica che un backend è in attesa di acquisire un blocco su una tupla mentre un altro backend tiene un blocco in conflitto sulla stessa tupla. Nella tabella seguente viene illustrato uno scenario in cui le sessioni generano l'evento `Lock:tuple`.

Orari	Sessione 1	Sessione 2	Sessione 3
t1	Inizia una transazione.		
t2	Aggiorna la riga 1.		
t3		Aggiorna la riga 1. La sessione acquisisce un blocco esclusivo sulla tupla e quindi attende che la sessione 1 rilasci il blocco eseguendo il commit o il rollback.	
t4			Aggiorna la riga 1. La sessione attende che la sessione 2 rilasci il blocco esclusivo sulla tupla.

Oppure puoi simulare questo evento di attesa utilizzando lo strumento di benchmarking `pgbench`. Configurare un numero elevato di sessioni simultanee per aggiornare la stessa riga in una tabella con un file SQL personalizzato.

Per ulteriori informazioni sulle modalità di blocco in conflitto, vedere Blocco esplicito nella documentazione di PostgreSQL. Per ulteriori informazioni su `pgbench`, consulta pgbench nella documentazione di PostgreSQL.

Probabili cause di aumento delle attese

Quando l'evento si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti:

- Un numero elevato di sessioni simultanee sta cercando di acquisire un blocco in conflitto per la stessa tupla eseguendo istruzioni `UPDATE` o `DELETE`.
- Sessioni altamente simultanee stanno eseguendo un'istruzione `SELECT` usando le modalità di blocco `FOR UPDATE` o `FOR NO KEY UPDATE`.
- Diversi fattori spingono le applicazioni o i connection pool ad aprire più sessioni per eseguire le stesse operazioni. Mentre le nuove sessioni stanno tentando di modificare le stesse righe, il carico del DB può aumentare e `Lock:tuple` può apparire.

Per ulteriori informazioni consulta Blocchi a livello di riga nella documentazione di PostgreSQL.

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Indagare la logica dell'applicazione
- Trova la sessione responsabile del blocco
- Riduci la concorrenza quando è alta
- Risoluzione dei problemi dei colli di bottiglia

Indagare la logica dell'applicazione

Scopri se una sessione bloccante è rimasta in stato `idle in transaction` per lungo tempo. In tal caso, considera di terminare la sessione di blocco come soluzione a breve termine. È anche possibile usare la funzione `pg_terminate_backend`. Per ulteriori informazioni su questa funzione, consulta Funzioni di segnalazione server nella documentazione di PostgreSQL.

Per una soluzione a lungo termine, fai quanto seguente:

- Regola la logica dell'applicazione.
- Utilizzo del parametro `idle_in_transaction_session_timeout`. Questo parametro termina qualsiasi sessione con una transazione aperta che è rimasta inattiva per un periodo di tempo

superiore al periodo di tempo specificato. Per ulteriori informazioni, consulta la pagina Errori connessione client nella documentazione di PostgreSQL.

- Usa autocommit il più possibile. Per ulteriori informazioni, consulta la pagina CONFIGURA AUTOCOMMIT nella documentazione di PostgreSQL.

Trova la sessione responsabile del blocco

Mentre si verifica l'evento di attesa `Lock:tuple`, identifica il blocco e la sessione bloccata scoprendo quali blocchi dipendono l'uno dall'altro. Per ulteriori informazioni, consulta Informazioni sulle dipendenze dei blocchi nel wiki di PostgreSQL. Per analizzare gli eventi `Lock:tuple` passati, usa la funzione Aurora `aurora_stat_backend_waits`.

L'esempio seguente esegue una query su tutte le sessioni, filtrando su `tuple` e ordinando per `wait_time`.

```
--AURORA_STAT_BACKEND_WAITS
      SELECT a.pid,
             a.usename,
             a.app_name,
             a.current_query,
             a.current_wait_type,
             a.current_wait_event,
             a.current_state,
             wt.type_name AS wait_type,
             we.event_name AS wait_event,
             a.waits,
             a.wait_time
        FROM (SELECT pid,
                     usename,
                     left(application_name,16) AS app_name,
                     coalesce(wait_event_type,'CPU') AS current_wait_type,
                     coalesce(wait_event,'CPU') AS current_wait_event,
                     state AS current_state,
                     left(query,80) as current_query,
                     (aurora_stat_backend_waits(pid)).*
                FROM pg_stat_activity
               WHERE pid <> pg_backend_pid()
                 AND usename<>'rdsadmin') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we
WHERE we.event_name = 'tuple'
```

```
    ORDER BY a.wait_time;

  pid  | usename | app_name |                 current_query                  |
 current_wait_type | current_wait_event | current_state | wait_type | wait_event |
 waits | wait_time
-------+---------+----------+------------------------------------------------
+------------------+--------------------+--------------+-----------+------------
+-------+-----------
 32136 | sys     | psql     | /*session3*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000018
 11999 | sys     | psql     | /*session4*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000024
```

Riduci la concorrenza quando è alta

L'evento `Lock:tuple` potrebbe verificarsi costantemente, soprattutto in un tempo di carico di lavoro occupato. In questa situazione, si consideri di ridurre l'elevata concorrenza per le file molto occupate. Spesso, solo poche righe controllano una coda o la logica booleana, il che rende queste righe molto occupate.

È possibile ridurre la concorrenza utilizzando approcci diversi in base ai requisiti aziendali, alla logica dell'applicazione e al tipo di carico di lavoro. Ad esempio, puoi eseguire le operazioni seguenti:

- Riprogetta la tua tabella e la logica dei dati per ridurre la concorrenza elevata.
- Modificare la logica dell'applicazione per ridurre la concorrenza elevata a livello di riga.
- Sfrutta e riprogetta le query con i blocchi a livello di riga.
- Utilizzo della clausola `NOWAIT` con operazioni di riprova.
- Prendi in considerazione l'utilizzo di un controllo della concorrenza logico ottimistico e ibrido.
- Valuta la possibilità di modificare il livello di isolamento del database.

Risoluzione dei problemi dei colli di bottiglia

`Lock:tuple` può verificarsi con colli di bottiglia come la fame di CPU o il massimo utilizzo della larghezza di banda Amazon EBS. Per ridurre i colli di bottiglia, valuta i seguenti approcci:

- Ridimensiona il tipo di classe di istanza.
- Ottimizza le query a uso intensivo di risorse.

- Modificare la logica dell'applicazione.
- Archivia i dati a cui si accede raramente.

LWLock:buffer_content (BufferContent)

L'evento `LWLock:buffer_content` si verifica quando una sessione è in attesa di accedere in lettura o scrittura a una pagina dati in memoria mentre un'altra sessione ha bloccato la pagina in scrittura. In Aurora PostgreSQL 13 e versioni successive, questo evento di attesa viene chiamato `BufferContent`.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

Per leggere o manipolare i dati, PostgreSQL vi accede tramite buffer di memoria condivisa. Per leggere dal buffer, un processo ottiene un blocco leggero (LWLock) sul contenuto del buffer in modalità condivisa. Per scrivere sul buffer, ottiene quel blocco in modalità esclusiva. I blocchi condivisi consentono ad altri processi di acquisire contemporaneamente blocchi condivisi su quel contenuto. I blocchi esclusivi impediscono ad altri processi di ottenere qualsiasi tipo di blocco.

L'evento `LWLock:buffer_content` (`BufferContent`) indica che più processi stanno tentando di ottenere un blocco sul contenuto di un buffer specifico.

Probabili cause di aumento delle attese

Quando l'evento `LWLock:buffer_content` (`BufferContent`) si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause tipiche includono le seguenti.

Aggiornamenti simultanei aumentati degli stessi dati

Potrebbe esserci un aumento del numero di sessioni simultanee con query che inseriscono o aggiornano la stessa tabella. Questa contesa può essere più marcata sulle tabelle con molti indici.

I dati del carico di lavoro non sono in memoria

Quando i dati elaborati dal carico di lavoro attivo non sono in memoria, questi eventi di attesa possono aumentare. Questo effetto è dovuto al fatto che i processi che contengono blocchi possono mantenerli più a lungo mentre eseguono operazioni di I/O su disco.

Uso eccessivo di vincoli di chiave esterna

I vincoli di chiave esterna possono aumentare la quantità di tempo che un processo mantiene su un blocco del contenuto del buffer. Questo effetto è dovuto al fatto che le operazioni di lettura richiedono un blocco del contenuto del buffer condiviso sulla chiave di riferimento mentre quella chiave viene aggiornata.

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa. Potresti identificare eventi `LWLock:buffer_content` (`BufferContent`) utilizzando Amazon RDS Performance Insights o interrogando la vista `pg_stat_activity`.

Argomenti

- Migliora l'efficienza in memoria
- Riduzione dell'utilizzo di vincoli di chiave esterna
- Rimuovere gli indici inutilizzati

Migliora l'efficienza in memoria

Per aumentare la probabilità che i dati del carico di lavoro attivo si trovino in memoria, partizionare tabelle o scalare la classe di istanza. Per informazioni sulle classi di istanza database, consulta Classi di istanze DB Amazon Aurora.

Riduzione dell'utilizzo di vincoli di chiave esterna

Indagare sui carichi di lavoro con un numero elevato di eventi di attesa `LWLock:buffer_content` (`BufferContent`) per l'utilizzo di vincoli di chiave esterna. Rimuovere i vincoli di chiave esterna non necessari.

Rimuovere gli indici inutilizzati

Per carichi di lavoro con un numero elevato di eventi di attesa `LWLock:buffer_content` (`BufferContent`), identificare gli indici inutilizzati e rimuoverli.

LWLock:buffer_mapping

Questo evento si verifica quando un processo di backend è in attesa di associare un blocco di dati a un buffer nel pool di buffer condiviso.

> **Note**
>
> Questo evento appare come `LWLock:buffer_mapping` in Aurora PostgreSQL versione 12 e versioni successive e `LWLock:BufferMapping` nella versione 13 e successive.

Argomenti

- Versioni del motore supportate
- Context
- Cause
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

Il pool buffer condiviso è un'area di memoria Aurora PostgreSQL che contiene tutte le pagine che sono o sono state utilizzate dai processi. Quando un processo ha bisogno di una pagina, legge la pagina nel buffer pool condiviso. Il parametro `shared_buffers` imposta le dimensioni del buffer condiviso e riserva un'area di memoria per memorizzare la tabella e le pagine indice. Se si modifica questo parametro, assicurarsi di riavviare il database. Per ulteriori informazioni, consultare Buffer condivisi.

L'evento di attesa `LWLock:buffer_mapping` si verifica nei seguenti scenari:

- Un processo ricerca nella tabella buffer una pagina e acquisisce un blocco di mappatura buffer condiviso.
- Un processo carica una pagina nel buffer pool e acquisisce un esclusivo blocco di mappatura del buffer.
- Un processo rimuove una pagina dal pool e acquisisce un blocco esclusivo di mappatura del buffer.

Cause

Quando questo evento appare più del normale, probabilmente indicando un problema di prestazioni, il database sta eseguendo il paging in entrata e in uscita dal buffer pool condiviso. Le cause tipiche sono:

- Query di grandi dimensioni
- Indici e tabelle gonfie
- Scansioni complete della tabella
- Dimensioni del pool condiviso più piccole del set di lavoro

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Monitora i parametri relativi al buffer
- Valuta la tua strategia di indicizzazione
- Riduci il numero di buffer che devono essere allocati rapidamente

Monitora i parametri relativi al buffer

Quando `LWLock:buffer_mapping` aspetta il picco, indaga il rapporto di hit del buffer. È possibile utilizzare questi parametri per comprendere meglio cosa sta accadendo nella cache del buffer. Esamina i seguenti parametri:

`BufferCacheHitRatio`

Il parametro Amazon CloudWatch misura la percentuale di richieste gestite dalla cache del buffer di un'istanza database nel cluster di database. È possibile che questo parametro diminuisca in anticipo all'evento di attesa `LWLock:buffer_mapping`.

`blks_hit`

Questo parametro contatore Performance Insights indica il numero di blocchi recuperati dal buffer pool condiviso. Dopo che appare l'evento di attesa `LWLock:buffer_mapping`, potresti osservare un picco in `blks_hit`.

`blks_read`

Questo parametro del contatore Performance Insights indica il numero di blocchi che richiedevano la lettura di I/O nel buffer pool condiviso. Potresti osservare un picco in `blks_read` in vista dell'evento di attesa `LWLock:buffer_mapping`.

Valuta la tua strategia di indicizzazione

Per confermare che la strategia di indicizzazione non sta peggiorando le prestazioni, verifica quanto segue:

Index bloat

Assicurati che il bloat di indice e tabella non porti alla lettura di pagine non necessarie nel buffer condiviso. Se le tabelle contengono righe inutilizzate, considera l'archiviazione dei dati e la rimozione delle righe dalle tabelle. È quindi possibile ricostruire gli indici per le tabelle ridimensionate.

Indici per query utilizzate di frequente

Per determinare se disponi degli indici ottimali, monitora i parametri del motore DB in Performance Insights. Il parametro `tup_returned` mostra il numero di righe lette. Il parametro `tup_fetched` mostra il numero di righe restituite al client. Se `tup_returned` è significativamente più grande di `tup_fetched`, i dati potrebbero non essere indicizzati correttamente. Inoltre, le statistiche della tabella potrebbero non essere aggiornate.

Riduci il numero di buffer che devono essere allocati rapidamente

Per ridurre gli eventi di attesa `LWLock:buffer_mapping`, cercare di ridurre il numero di buffer che devono essere allocati rapidamente. Una strategia consiste nell'eseguire operazioni di batch di dimensioni ridotte. Potresti essere in grado di ottenere batch più piccoli partizionando le tabelle.

LWLock:bufferIO (IPC: bufferio)

L'`LWLock:BufferIO`evento si verifica quando Aurora PostgreSQL o RDS for PostgreSQL sono in attesa che altri processi completino le loro operazioni mentre tentano contemporaneamente di accedere a una pagina. input/output (I/O Il suo scopo è quello di leggere la stessa pagina nel buffer condiviso.

Argomenti

- Versioni di motori pertinenti
- Context
- Cause
- Azioni

Versioni di motori pertinenti

Queste informazioni relative all'evento di attesa sono rilevanti per tutte le versioni di Aurora PostgreSQL. Per Aurora PostgreSQL 12 e versioni precedenti questo evento di attesa è denominato lwlock:buffer_io mentre in Aurora PostgreSQL versione 13 è denominato lwlock:bufferio. In Aurora PostgreSQL versione 14, l'evento di attesa BufferIO è stato spostato da `LWLock` al tipo di evento di attesa `IPC` (IPC:BufferIO).

Context

Ogni buffer condiviso ha un blocco I/O associato all'evento di attesa `LWLock:BufferIO`, ogni volta che un blocco (o una pagina) deve essere recuperato all'esterno del buffer pool condiviso.

Questo blocco viene utilizzato per gestire più sessioni che richiedono tutte l'accesso allo stesso blocco. Questo blocco deve essere letto dall'esterno del buffer pool condiviso, definito dal parametro `shared_buffers`.

Non appena la pagina viene letta all'interno del buffer pool condiviso, il blocco `LWLock:BufferIO` viene rilasciato.

> **Note**
>
> L'evento di attesa `LWLock:BufferIO` precede l'evento di attesa IO: DataFileRead. L'evento di attesa `IO:DataFileRead` si verifica mentre i dati vengono letti dallo storage.

Per ulteriori informazioni sui blocchi leggeri, consultaPanoramica dei blocchi.

Cause

Le cause comuni della comparsa dell'evento `LWLock:BufferIO` che appare nelle prime attese includono:

- Più backend o connessioni che tentano di accedere alla stessa pagina in attesa di un'operazione di I/O
- Il rapporto tra le dimensioni del buffer pool condiviso (definito dal parametro `shared_buffers`) e il numero di buffer necessari per il carico di lavoro corrente
- La dimensione del buffer pool condiviso non è ben bilanciata con il numero di pagine sfrattate da altre operazioni
- Indici grandi o gonfi che richiedono al motore di leggere più pagine del necessario nel buffer pool condiviso
- Mancanza di indici che costringe il motore DB a leggere più pagine dalle tabelle del necessario
- Picchi improvvisi per le connessioni al database che tentano di eseguire operazioni sulla stessa pagina

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa:

- Osserva le CloudWatch metriche di Amazon per verificare la correlazione tra forti diminuzioni degli eventi `BufferCacheHitRatio` e di `LWLock:BufferIO` attesa. Questo effetto può significare che hai una piccola impostazione dei buffer condivisi. Potrebbe essere necessario aumentarlo o scalare la classe di istanza DB. È possibile dividere il carico di lavoro in più nodi di lettore.
- Verifica se hai indici inutilizzati, quindi rimuovili.
- Utilizzare tabelle partizionate (che hanno anche indici partizionati). Ciò aiuta a mantenere basso il riordino dell'indice e ne riduce l'impatto.

- Evitare di indicizzare inutilmente le colonne.
- Evita improvvisi picchi di connessione al database utilizzando un connection pool.
- Limitare il numero massimo di connessioni al database come best practice.

LWLock:lock_manager

Questo evento si verifica quando il motore Aurora PostgreSQL mantiene l'area di memoria del blocco condiviso per allocare, controllare e dislocare un blocco quando non è possibile un blocco rapido del percorso.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per Aurora PostgreSQL versione 9.6 e successive.

Context

Quando si emette un'istruzione SQL, Aurora PostgreSQL registra i blocchi per proteggere la struttura, i dati e l'integrità del database durante le operazioni simultanee. Il motore può raggiungere questo obiettivo utilizzando un blocco veloce del percorso o un blocco del percorso che non è veloce. Un blocco del percorso che non è veloce è più costoso e crea più sovraccarico di un blocco veloce del percorso.

Blocco rapido del percorso

Per ridurre il sovraccarico dei blocchi che vengono presi e rilasciati frequentemente, ma che raramente sono in conflitto, i processi di backend possono utilizzare il blocco rapido del percorso. Il database utilizza questo meccanismo per i blocchi che soddisfano i seguenti criteri:

- Usano il metodo di blocco DEFAULT.

- Rappresentano un blocco su una relazione di database anziché una relazione condivisa.
- Sono blocchi deboli che difficilmente entrino in conflitto.
- Il motore può verificare rapidamente che non siano presenti blocchi in conflitto.

Il motore non può utilizzare il blocco rapido del percorso quando una di queste condizioni è vera:

- Il blocco non soddisfa i criteri precedenti.
- Non sono disponibili più slot per il processo di backend.

Per ulteriori informazioni sul blocco rapido del percorso, consulta percorso rapido nel gestore di blocco PostgreSQL README e pg-locks nella documentazione di PostgreSQL.

Esempio di problema di scalabilità per il blocco manager

In questo esempio, una tabella denominata `purchases` memorizza cinque anni di dati, partizionati per giorno. Ogni partizione ha due indici. Si verifica la seguente sequenza di eventi:

1. Si interrogano dati su diversi giorni, il che richiede al database di leggere molte partizioni.
2. Il database crea una voce di blocco per ogni partizione. Se gli indici di partizione fanno parte del percorso di accesso dell'ottimizzatore, il database crea anche una voce di blocco per loro.
3. Quando il numero di voci di blocco richieste per lo stesso processo di back-end è superiore a 16, ovvero il valore di `FP_LOCK_SLOTS_PER_BACKEND`, il gestore di blocco utilizza il metodo di blocco del percorso non veloce.

Le applicazioni moderne potrebbero avere centinaia di sessioni. Se le sessioni simultanee eseguono una query sul genitore senza un'adeguata scrematura delle partizioni, il database potrebbe creare centinaia o addirittura migliaia di blocchi di percorso non veloci. In genere, quando questa concorrenza è superiore al numero di vCPU, appare l'evento di attesa `LWLock:lock_manager`.

> **Note**
>
> L'evento di attesa `LWLock:lock_manager` non è correlato al numero di partizioni o indici in uno schema di database. Al contrario, è correlato al numero di blocchi di percorso non veloci che il database deve controllare.

Probabili cause di aumento delle attese

Quando l'evento di attesa `LWLock:lock_manager` si verifica più del normale, probabilmente indicando un problema di prestazioni, le cause più probabili di picchi improvvisi sono le seguenti:

- Le sessioni attive simultanee eseguono query che non utilizzano blocchi di percorso veloci. Queste sessioni superano anche la vCPU massima.
- Un gran numero di sessioni attive simultanee sta accedendo a una tabella fortemente partizionata. Ogni partizione ha più indici.
- Il database sta vivendo una tempesta di connessione. Per impostazione predefinita, alcune applicazioni e software del connection pool creano più connessioni quando il database è lento. Questa pratica peggiora il problema. Ottimizza il software del pool di connessioni in modo che non si verifichino tempeste di connessione.
- Un numero elevato di sessioni esegue una query su una tabella padre senza potare le partizioni.
- Un DDL (Data Definition Language), DML (Data Manipolation Language) o un comando di manutenzione blocca esclusivamente una relazione occupata o tuple a cui si accede frequentemente o modificate.

Operazioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

Argomenti

- Usare la potatura delle partizioni
- Rimuovere indici non necessari
- Ottimizza le tue query per bloccare rapidamente i percorsi
- Sintonizzati per altri eventi di attesa
- Riduzione dei colli di bottiglia hardware
- Utilizzare un connection pooler
- Aggiorna la versione Aurora PostgreSQL

Usare la potatura delle partizioni

Potatura delle partizioni è una strategia di ottimizzazione delle query che esclude le partizioni non necessarie dalle scansioni di tabelle, migliorando così le prestazioni. La potatura delle partizioni è attivata per impostazione predefinita. Se è spento, accenderlo come segue.

```
SET enable_partition_pruning = on;
```

Le query possono trarre vantaggio dalla potatura delle partizioni quando la clausola `WHERE` contiene la colonna utilizzata per il partizionamento. Per ulteriori informazioni, consulta Potatura delle partizioni nella documentazione di PostgreSQL.

Rimuovere indici non necessari

Il database potrebbe contenere indici inutilizzati o usati raramente. In tal caso, considera la possibilità di eliminarli. Eseguire una delle operazioni seguenti:

- Scopri come trovare indici non necessari consultanto Indici non utilizzati nel wiki di PostgreSQL.
- Esegui PG Collector. Questo script SQL raccoglie le informazioni del database e le presenta in un report HTML consolidato. Controlla la sezione «Indici non utilizzati». Per ulteriori informazioni, consulta pg-collector nel repository AWS Labs GitHub.

Ottimizza le tue query per bloccare rapidamente i percorsi

Per scoprire se le tue query utilizzano il blocco rapido dei percorsi, esegui una query nella colonna `fastpath` della tabella `pg_locks`. Se le query non utilizzano il blocco rapido dei percorsi, provare a ridurre il numero di relazioni per query a meno di 16.

Sintonizzati per altri eventi di attesa

Se `LWLock:lock_manager` è il primo o il secondo nell'elenco delle prime attese, controlla se nell'elenco vengono visualizzati anche i seguenti eventi di attesa:

- `Lock:Relation`
- `Lock:transactionid`
- `Lock:tuple`

Se gli eventi precedenti appaiono in alto nell'elenco, prendi in considerazione prima l'ottimizzazione di questi eventi di attesa. Questi eventi possono essere un driver per `LWLock:lock_manager`.

Riduzione dei colli di bottiglia hardware

Potresti avere un collo di bottiglia hardware, come la fame di CPU o il massimo utilizzo della larghezza di banda Amazon EBS. In questi casi, valuta la riduzione dei colli di bottiglia hardware. Prendi in considerazione le seguenti azioni:

- Ridimensiona la tua classe di istanza.
- Ottimizza le query che consumano grandi quantità di CPU e memoria.
- Cambia la logica dell'applicazione.
- Archivia i tuoi dati.

Per ulteriori informazioni su CPU, memoria e larghezza di banda di rete EBS, vedere Tipi di istanza Amazon RDS.

Utilizzare un connection pooler

Se il numero totale di connessioni attive supera la vCPU massima, più processi del sistema operativo richiedono CPU di quella supportata dal tipo di istanza. In questo caso, valuta l'utilizzo o l'ottimizzazione di un connection pool. Per ulteriori informazioni sul numero di vCPU per ogni tipo di istanza, consulta Tipi di istanza di Amazon RDS.

Per ulteriori informazioni sul connection pool, consulta le risorse seguenti:

- Proxy Amazon RDS per Aurora
- pgbouncer
- Connection pool e origini dati nella Documentazione di PostgreSQL

Aggiorna la versione Aurora PostgreSQL

Se la versione attuale di Aurora PostgreSQL è inferiore a 12, esegui l'aggiornamento alla versione 12 o superiore. Le versioni 12 e 13 di PostgreSQL hanno un meccanismo di partizione migliorato. Per ulteriori informazioni sui miglioramenti nella versioni 12, consulta PostgreSQL 12 Release Notes. Per ulteriori informazioni sulle versioni di Aurora PostgreSQL, consulta Aggiornamenti del motore di database per Amazon Aurora Postgre SQL.

LWLock:MultiXact

Gli eventi `LWLock:MultiXactMemberBufferLWLock:MultiXactOffsetBuffer`,`LWLock:MultiXactMemberSL` e `LWLock:MultiXactOffsetSLRU` wait indicano che una sessione è in attesa di recuperare un elenco di transazioni che modifica la stessa riga in una determinata tabella.

- `LWLock:MultiXactMemberBuffer`: un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un membro multixact.
- `LWLock:MultiXactMemberSLRU`: un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un membro multixact.
- `LWLock:MultiXactOffsetBuffer`: un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un offset multixact.
- `LWLock:MultiXactOffsetSLRU`: un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un offset multixact.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di aumento delle attese
- Azioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Context

Un multixact è una struttura di dati che memorizza un elenco di transazioni IDs (XIDs) che modificano la stessa riga della tabella. Quando una singola transazione fa riferimento a una riga in una tabella, l'ID della transazione viene memorizzato nella riga di intestazione della tabella. Quando più transazioni fanno riferimento alla stessa riga in una tabella, l'elenco delle transazioni IDs viene memorizzato nella struttura di dati multixact. Gli eventi di attesa multixact indicano che una sessione sta recuperando dalla struttura dei dati l'elenco delle transazioni che fanno riferimento a una determinata riga di una tabella.

Probabili cause di aumento delle attese

Le tre cause più comuni dell'uso di multixact sono le seguenti:

- Sottotransazioni da punti di salvataggio espliciti: la creazione esplicita di un punto di salvataggio nelle transazioni genera nuove transazioni per la stessa riga. Ad esempio, utilizzando `SELECT FOR UPDATE`, `SAVEPOINT` e quindi `UPDATE`.

  Alcuni driver, object relational mapper (ORMs) e layer di astrazione dispongono di opzioni di configurazione per includere automaticamente tutte le operazioni con punti di salvataggio. Questo comportamento può generare molti eventi di attesa multixact in alcuni carichi di lavoro. L'opzione `autosave` del driver PostgreSQL JDBC ne è un esempio. Per ulteriori informazioni, consulta pgJDBC nella documentazione di PostgreSQL JDBC. Un altro esempio è il driver PostgreSQL ODBC e l'opzione `protocol`. Per ulteriori informazioni, consulta psqlODBC Configuration Options (Opzioni di configurazione di psqlODBC) nella documentazione del driver PostgreSQL ODBC.
- Sottotransazioni dalle clausole PL/pgSQL EXCEPTION — Ogni clausola che scrivi nelle funzioni o procedure PL/pgSQL ne crea una internamente. `EXCEPTION SAVEPOINT`
- Chiavi esterne. Più transazioni acquisiscono un blocco di condivisione nel record padre.

Quando una determinata riga è inclusa in un'operazione di transazione multipla, l'elaborazione della riga richiede il recupero della transazione dagli elenchi. IDs `multixact` Se le ricerche non riescono a ottenere il multixact dalla cache di memoria, la struttura dei dati deve essere letta dal livello di archiviazione di Aurora. Questa operazione I/O dall'archiviazione significa che le query SQL possono richiedere più tempo. Gli errori nella cache di memoria possono iniziare a verificarsi con l'utilizzo intensivo, a causa del numero elevato di transazioni multiple. Tutti questi fattori contribuiscono ad un aumento di questo evento di attesa.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa. Alcune di queste azioni possono aiutare a ridurre immediatamente gli eventi di attesa. Tuttavia, altre potrebbero richiedere indagini e correzioni per aumentare il carico di lavoro.

Argomenti

- Esegui il congelamento sottovuoto sui tavoli con questo evento di attesa
- Aumenta la frequenza di aspirazione automatica sui tavoli con questo evento di attesa
- Aumentare i parametri di memoria

- Riduci le transazioni di lunga durata
- Azioni a lungo termine

Esegui il congelamento sottovuoto sui tavoli con questo evento di attesa

Se questo evento di attesa si verifica improvvisamente e influisce sull'ambiente di produzione, puoi utilizzare uno dei seguenti metodi temporanei per ridurne il numero.

- Utilizzate VACUUM FREEZE sulla tabella o sulla partizione di tabella interessata per risolvere immediatamente il problema. Per ulteriori informazioni, vedere VACUUM.
- Utilizzate la clausola VACUUM (FREEZE, INDEX_CLEANUP FALSE) per eseguire un'aspirazione rapida saltando gli indici. Per ulteriori informazioni, vedere Aspirare una tabella il più rapidamente possibile.

Aumenta la frequenza di aspirazione automatica sui tavoli con questo evento di attesa

Dopo aver scansionato tutte le tabelle in tutti i database, VACUUM alla fine rimuoverà i multixact e i loro valori multixact più vecchi verranno aggiornati. Per ulteriori informazioni, vedere Multixacts e Wraparound. Per ridurre al minimo gli eventi LWLock: MultiXact wait, è necessario eseguire VACUUM tutte le volte che è necessario. A tale scopo, assicurati che VACUUM nel cluster Aurora PostgreSQL DB sia configurato in modo ottimale.

Se l'utilizzo di VACUUM FREEZE sulla tabella o sulla partizione di tabella interessata risolve il problema dell'evento di attesa, consigliamo di utilizzare uno scheduler, ad esempio`pg_cron`, per eseguire VACUUM invece di regolare l'autovacuum a livello di istanza.

Affinché l'autovacuum avvenga più frequentemente, puoi ridurre il valore del parametro di archiviazione nella tabella interessata. `autovacuum_multixact_freeze_max_age` Per ulteriori informazioni, vedere autovacuum_multixact_freeze_max_age.

Aumentare i parametri di memoria

È possibile ottimizzare l'utilizzo della memoria per le cache multixact regolando i seguenti parametri. Queste impostazioni controllano la quantità di memoria riservata a queste cache, il che può aiutare a ridurre gli eventi di attesa multixact nel carico di lavoro. Ti consigliamo di iniziare con i seguenti valori:

Per Aurora PostgreSQL 17 e versioni successive:

- `multixact_offset_buffers`= 128

- `multixact_member_buffers`= 256

Per Aurora PostgreSQL 16 e versioni precedenti:

- `multixact_offsets_cache_size`= 128
- `multixact_members_cache_size`= 256

> **Note**
>
> In Aurora PostgreSQL 17, i nomi dei parametri venivano modificati da e `multixact_offset_buffers` da `multixact_offsets_cache_size` a per allinearli alla community PostgreSQL 17. `multixact_members_cache_size` `multixact_member_buffers`

È possibile impostare questi parametri a livello di cluster in modo che tutte le istanze del cluster rimangano coerenti. Ti consigliamo di testare e modificare i valori per adattarli al meglio ai requisiti specifici del carico di lavoro e alla classe di istanza. È necessario riavviare l'istanza di writer per rendere effettive le modifiche ai parametri.

I parametri sono espressi in termini di voci di cache multixact. Ogni voce della cache utilizza memoria`8 KB`. Per calcolare la memoria totale riservata, moltiplica il valore di ogni parametro per`8 KB`. Ad esempio, se imposti un parametro su 128, la memoria riservata totale sarebbe`128 * 8 KB = 1 MB`.

Riduci le transazioni di lunga durata

Una transazione di lunga durata fa sì che il vuoto conservi le informazioni fino al completamento della transazione o fino alla chiusura della transazione di sola lettura. Ti consigliamo di monitorare e gestire in modo proattivo le transazioni di lunga durata. Per ulteriori informazioni, consulta Il database ha una connessione di transazione inattiva da molto tempo. Prova a modificare l'applicazione per evitare o ridurre al minimo l'uso di transazioni di lunga durata.

Azioni a lungo termine

Esamina il tuo carico di lavoro per scoprire la causa dello spillover multixact. È necessario risolvere il problema per scalare il carico di lavoro e ridurre l'attesa.

- È necessario analizzare il DDL (Data Definition Language) utilizzato per creare le tabelle. Assicurati che le strutture e gli indici delle tabelle siano ben progettati.

- Se le tabelle interessate hanno chiavi esterne, stabilite se sono necessarie o se esiste un altro modo per applicare l'integrità referenziale.
- Quando una tabella ha indici inutilizzati di grandi dimensioni, autovacuum può non adattarsi al carico di lavoro e potrebbe impedirne l'esecuzione. Per evitare ciò, controlla gli indici non utilizzati e rimuovili completamente. Per ulteriori informazioni, consulta Gestire l'autovacuum con indici di grandi dimensioni.
- Riduci l'uso dei savepoint nelle tue transazioni.

Timeout: PG Sleep

L'evento `Timeout:PgSleep` si verifica quando un processo server ha chiamato la funzione `pg_sleep` e sta aspettando la scadenza del timeout del sonno.

Argomenti

- Versioni del motore supportate
- Probabili cause di aumento delle attese
- Operazioni

Versioni del motore supportate

Queste informazioni relative all'evento di attesa sono supportate per tutte el versioni di Aurora PostgreSQL.

Probabili cause di aumento delle attese

Questo evento di attesa si verifica quando un'applicazione, una funzione memorizzata o un utente emette un'istruzione SQL che chiama una delle seguenti funzioni:

- `pg_sleep`
- `pg_sleep_for`
- `pg_sleep_until`

Le funzioni precedenti ritardano l'esecuzione fino a quando non è trascorso il numero di secondi specificato. Ad esempio: `SELECT pg_sleep(1)` si ferma per 1 secondo. Per ulteriori informazioni, consulta Ritardo dell'esecuzione nella documentazione di PostgreSQL.

Operazioni

Identificare la dichiarazione che stava eseguendo la funzione `pg_sleep`. Determina se l'uso della funzione è appropriato.

Ottimizzazione con informazioni proattive su Amazon SQL Guru DevOps

DevOpsGuru proactive Insights rileva le condizioni sui cluster Aurora SQL Postgre DB che possono causare problemi e ti informa prima che si verifichino. Proactive Insights può avvisarti di un periodo di inattività prolungato nella connessione delle transazioni. Per ulteriori informazioni sulla risoluzione dei problemi di inattività prolungati nelle connessioni di transazione, consulta Il database ha una connessione di transazione inattiva da molto tempo

DevOpsGuru può fare quanto segue:

- Evitare molti problemi comuni relativi al database controllando la configurazione del database rispetto alle impostazioni consigliate comuni.
- Ricevere gli avvisi per le criticità relative al parco istanze che, se non controllate, possono portare a problemi più gravi in seguito.
- Ricevere gli avvisi per i nuovi problemi individuati.

Ogni approfondimento proattivo contiene un'analisi della causa del problema e i suggerimenti per le azioni correttive.

Per ulteriori informazioni su Amazon DevOps Guru per AmazonRDS, consultaAnalisi delle anomalie prestazionali di Aurora con DevOps Amazon Guru per Amazon RDS.

Il database ha una connessione di transazione inattiva da molto tempo

Una connessione al database è nello stato `idle in transaction` da più di 1800 secondi.

Argomenti

- Versioni del motore supportate
- Context
- Probabili cause di questo problema

- Azioni
- Parametri rilevanti

Versioni del motore supportate

Queste informazioni dettagliate sono supportate per tutte le versioni di RDS Postgre . SQL

Context

Una transazione nello stato `idle in transaction` può contenere blocchi che impediscono l'esecuzione di altre query. Può anche impedire al `VACUUM` (incluso l'autovacuum) di cancellare le righe inutilizzate, con conseguente aumento delle dimensioni dell'indice o della tabella o del wraparound dell'ID della transazione.

Probabili cause di questo problema

Una transazione iniziata in una sessione interattiva con BEGIN o START TRANSACTION non è terminata utilizzando un comando, o. COMMIT ROLLBACK END Lo stato della transazione diventa pertanto `idle in transaction`.

Azioni

Puoi individuare le transazioni inattive eseguendo la query `pg_stat_activity`.

Nel tuo SQL client, esegui la seguente query per elencare tutte le connessioni in `idle in transaction` stato e ordinarle per durata:

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

Consigliamo azioni diverse a seconda delle cause degli approfondimenti.

Argomenti

- Terminare la transazione
- Interrompere la connessione

- Configurare il parametro idle_in_transaction_session_timeout
- Controlla lo stato AUTOCOMMIT
- Controllare la logica delle transazioni nel codice dell'applicazione

Terminare la transazione

Quando si avvia una transazione in una sessione interattiva con BEGIN o STARTTRANSACTION, passa allo `idle in transaction` stato. Rimane in questo stato fino a quando non si termina la transazione emettendo un END comandoCOMMIT,ROLLBACK, o si disconnette completamente la connessione per ripristinare la transazione.

Interrompere la connessione

Interrompi la connessione con una transazione inattiva utilizzando la seguente query:

```
SELECT pg_terminate_backend(pid);
```

pid è l'ID di processo della connessione.

Configurare il parametro idle_in_transaction_session_timeout

Configura il parametro `idle_in_transaction_session_timeout` nel gruppo di parametri. Il vantaggio della configurazione di questo parametro è che non richiede un intervento manuale per terminare la transazione inattiva da tempo. Per ulteriori informazioni su questo parametro, consulta la documentazione di SQL Postgre.

Il seguente messaggio verrà riportato nel file di SQL registro di Postgre dopo l'interruzione della connessione, quando una transazione rimane nello stato idle_in_transaction per un periodo più lungo del tempo specificato.

```
FATAL: terminating connection due to idle in transaction timeout
```

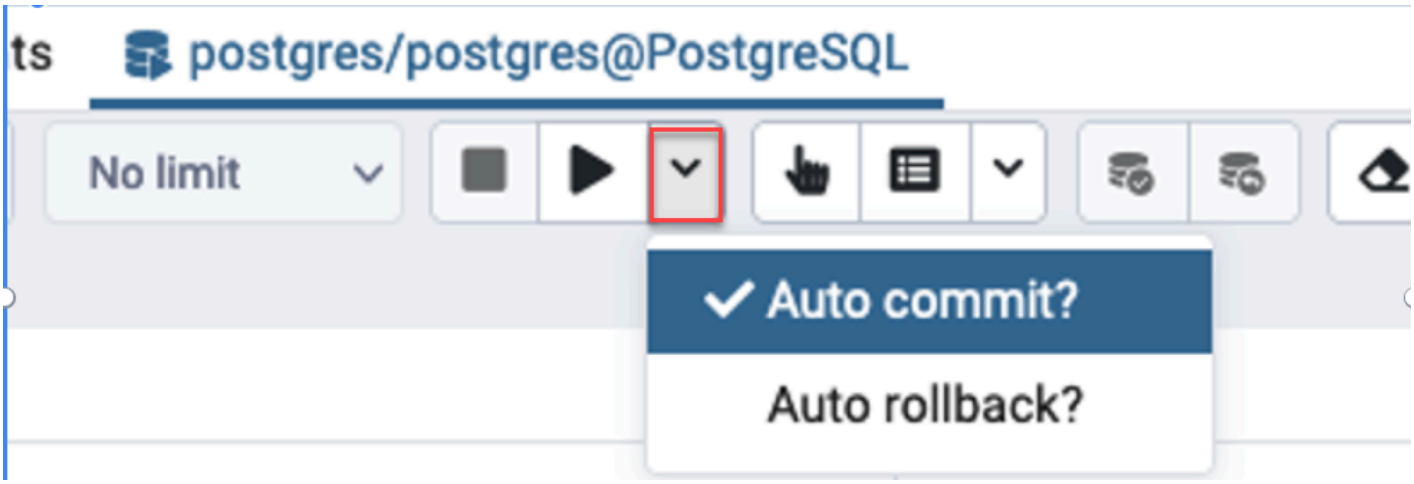
Controlla lo stato AUTOCOMMIT

AUTOCOMMITè attivata per impostazione predefinita. Tuttavia accidentalmente se viene disattivato nel client, assicurati di riattivarlo.

- Nel client psql, esegui il comando seguente:

```
postgres=> \set AUTOCOMMIT on
```

- In pgadmin, attivalo scegliendo l'AUTOCOMMITopzione dalla freccia rivolta verso il basso.



Controllare la logica delle transazioni nel codice dell'applicazione

Controlla la logica dell'applicazione per individuare eventuali problemi. Prendi in considerazione le seguenti azioni:

- Controlla se il commit JDBC automatico è impostato su true nella tua applicazione. Inoltre, considera l'utilizzo di comandi `COMMIT` espliciti nel codice.
- Controlla la logica di gestione degli errori per vedere se chiude una transazione dopo gli errori.
- Controlla se l'applicazione impiega molto tempo per elaborare le righe restituite da una query mentre la transazione è aperta. In tal caso, valuta la possibilità di codificare l'applicazione per chiudere la transazione prima di elaborare le righe.
- Controlla se una transazione contiene molte operazioni di lunga durata. In tal caso, dividi la singola transazione in più transazioni.

Parametri rilevanti

A questo approfondimento sono correlati i seguenti parametri PI:

- idle_in_transaction_count - Numero di sessioni nello stato `idle in transaction`.
- idle_in_transaction_max_time - La durata della transazione in esecuzione più lunga nello stato `idle in transaction`.

Best practice con Amazon Aurora PostgreSQL

Di seguito, sono disponibili diverse best practice per gestire il cluster database Amazon Aurora PostgreSQL. Assicurati di esaminare anche le attività di manutenzione di base. Per ulteriori informazioni, consulta Prestazioni e scalabilità per Amazon Aurora PostgreSQL.

Argomenti

- Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per le istanze database Aurora PostgreSQL
- Diagnosi delle dimensioni della tabella e dell'indice
- Migliore gestione della memoria in Aurora PostgreSQL
- Failover rapido con Amazon Aurora PostgreSQL
- Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL
- Gestione dell’abbandono della connessione Aurora PostgreSQL con pooling
-
- Utilizzo dei CloudWatch parametri di Amazon per analizzare l'utilizzo delle risorse per Aurora PostgreSQL
- Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL
- Risoluzione dei problemi di storage in Aurora PostgreSQL

Evitare il rallentamento delle prestazioni, il riavvio automatico e il failover per le istanze database Aurora PostgreSQL

Se si eseguono uno o più carichi di lavoro pesanti che eccedono le risorse allocate dell'istanza database, è possibile che le risorse su cui si esegue l'applicazione e il database Aurora si esauriscano. Per ottenere parametri sulla tua istanza di database come l'utilizzo della CPU, l'utilizzo della memoria e il numero di connessioni al database utilizzate, puoi fare riferimento ai parametri forniti da Amazon, Performance CloudWatch Insights e Enhanced Monitoring. Per ulteriori informazioni sul monitoraggio dell'istanza database, consultare Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Se il carico di lavoro esaurisce le risorse in uso, è possibile che l'istanza database rallenti, venga riavviata o esegua un failover su un'altra istanza database. Per evitare che questo si verifichi, occorre

monitorare l'utilizzo delle risorse, esaminare il carico di lavoro in esecuzione sull'istanza database ed eseguire le ottimizzazioni dove necessario. Se le ottimizzazioni non migliorano le metriche dell'istanza e non mitigano l'esaurimento delle risorse, valutare la possibilità di aumentare l'istanza database prima che raggiunga i suoi limiti. Per ulteriori informazioni sulle classi di istanza database disponibili e le relative specifiche, consultare Classi di istanze DB Amazon Aurora.

Diagnosi delle dimensioni della tabella e dell'indice

Puoi usare PostgreSQL Multiversion Concurrency Control (MVCC) per preservare l'integrità dei dati. PostgreSQL MVCC funziona salvando una copia interna delle righe aggiornate o eliminate (chiamate anche tuple) fino al commit o al rollback di una transazione. Questa copia interna salvata è invisibile agli utenti. Tuttavia, le dimensioni della tabella possono aumentare quando le copie invisibili non vengono pulite regolarmente dalle utilità VACUUM o AUTOVACUUM. Se non controllato, l'aumento delle dimensioni della tabella può comportare un aumento dei costi di archiviazione e rallentare la velocità di elaborazione.

In molti casi, le impostazioni predefinite per VACUUM o AUTOVACUUM su Aurora sono sufficienti per gestire l'aumento indesiderato delle dimensioni della tabella. Tuttavia, verifica la presenza di un aumento delle dimensioni se l'applicazione presenta le seguenti condizioni:

- Elabora un gran numero di transazioni in un tempo relativamente breve tra i processi VACUUM.
- Funziona male e esaurisce lo spazio di archiviazione.

Per iniziare, raccogli le informazioni accurate su quanto spazio viene utilizzato dalle tuple inattive e su quanto puoi recuperare ripulendo le dimensioni della tabella e dell'indice. Per farlo, usa l'estensione `pgstattuple` per raccogliere statistiche sul cluster Aurora. Per ulteriori informazioni, consulta pgstattuple. I privilegi di utilizzo dell'estensione `pgstattuple` sono limitati al ruolo `pg_stat_scan_tables` e ai superuser del database.

Per creare l'estensione `pgstattuple` su Aurora, connetti una sessione client al cluster, ad esempio psql o pgAdmin, e usa il seguente comando:

```
CREATE EXTENSION pgstattuple;
```

Crea l'estensione in ogni database da profilare. Dopo aver creato l'estensione, usa l'interfaccia della linea di comando (CLI) per misurare la quantità di spazio inutilizzato che puoi recuperare. Prima di raccogliere le statistiche, modifica il gruppo di parametri del cluster impostando AUTOVACUUM su

0. Un'impostazione pari a 0 impedisce ad Aurora di eliminare automaticamente tutte le tuple inattive dell'applicazione, il che può influire sulla precisione dei risultati. Immetti il seguente comando per creare una tabella semplice:

```
postgres=> CREATE TABLE lab AS SELECT generate_series (0,100000);
SELECT 100001
```

Nell'esempio seguente, eseguiamo la query con AUTOVACUUM attivato per il cluster di database. Il valore di `dead_tuple_count` è pari a 0, che indica che AUTOVACUUM ha cancellato i dati obsoleti o le tuple dal database PostgreSQL.

Per utilizzare `pgstattuple` per raccogliere informazioni sulla tabella, specifica il nome di una tabella o un identificatore di oggetto (OID) nella query:

```
postgres=> SELECT * FROM pgstattuple('lab');
```

```
table_len  | tuple_count | tuple_len | tuple_percent | dead_tuple_count |
 dead_tuple_len | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056    | 100001      | 2800028   | 77.16         | 0                | 0
  | 0                  | 16616      | 0.46
(1 row)
```

Nella seguente query, disattiviamo AUTOVACUUM ed eseguiamo un comando che elimina 25.000 righe dalla tabella. Di conseguenza, il valore di `dead_tuple_count` aumenta a 25000.

```
postgres=> DELETE FROM lab WHERE generate_series < 25000;

DELETE 25000
```

```
SELECT * FROM pgstattuple('lab');
```

```
table_len | tuple_count | tuple_len | tuple_percent | dead_tuple_count | dead_tuple_len
 | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056 | 75001 | 2100028 | 57.87 | 25000 | 700000 | 19.29 | 16616 | 0.46
(1 row)
```

Per recuperare le tuple inattive, avvia un processo VACUUM.

Osservazione delle dimensioni senza interrompere l'applicazione

Le impostazioni su un cluster Aurora sono ottimizzate per fornire le best practice per la maggior parte dei carichi di lavoro. Tuttavia, potresti voler ottimizzare un cluster per adattarlo meglio alle tue applicazioni e ai tuoi modelli di utilizzo. In questo caso, puoi utilizzare l'estensione `pgstattuple` senza interrompere un'applicazione in esecuzione. A tale scopo, esegui i seguenti passaggi:

1. Clona la tua istanza Aurora.
2. Modifica il file dei parametri per disattivare AUTOVACUUM nel clone.
3. Esegui una query `pgstattuple` durante il test del clone con un carico di lavoro di esempio o con pgbench, che è un programma per eseguire test di benchmark su PostgreSQL. Per ulteriori informazioni, consulta pgbench.

Dopo aver eseguito le applicazioni e visualizzato il risultato, usa pg_repack o VACUUM FULL sulla copia ripristinata e confronta le differenze. Se noti un calo significativo dei valori di dead_tuple_count, dead_tuple_len o dead_tuple_percent, modifica il programma di vacuum sul cluster di produzione per ridurre al minimo l'aumento delle dimensioni.

Impedimento dell'aumento delle dimensioni nelle tabelle temporanee

Se l'applicazione crea tabelle temporanee, assicurati che vengano rimosse quando non sono più necessarie. I processi Autovacuum non individuano le tabelle temporanee. Se non selezionate, le tabelle temporanee possono creare rapidamente un aumento delle dimensioni del database. Inoltre,

l'aumento delle dimensioni può estendersi alle tabelle di sistema, che sono le tabelle interne che tracciano gli oggetti e gli attributi di PostgreSQL, come pg_attribute e pg_depend.

Quando una tabella temporanea non è più necessaria, è possibile utilizzare un'istruzione TRUNCATE per svuotarla e liberare spazio. Quindi, esegui manualmente il vacuum delle tabelle pg_attribute e pg_depend. Il vacuum di queste tabelle garantisce che la creazione e il troncamento o l'eliminazione delle tabelle temporanee in modo continuo non comporti l'aggiunta di tuple né contribuisca all'aumento delle dimensioni del sistema.

Puoi evitare questo problema durante la creazione di una tabella temporanea includendo la seguente sintassi che elimina le nuove righe quando il contenuto viene sottoposto a commit:

```
CREATE TEMP TABLE IF NOT EXISTS table_name(table_description) ON COMMIT DELETE ROWS;
```

La clausola `ON COMMIT DELETE ROWS` tronca la tabella temporanea quando la transazione viene sottoposta a commit.

Impedimento dell'aumento delle dimensioni negli indici

Quando si modifica un campo indicizzato in una tabella, l'aggiornamento dell'indice genera una o più tuple inattive in quell'indice. Per impostazione predefinita, il processo autovacuum elimina l'aumento delle dimensioni negli indici, ma tale pulizia richiede una notevole quantità di tempo e risorse. Per specificare le preferenze di pulizia dell'indice quando crei una tabella, includi la clausola vacuum_index_cleanup. Per impostazione predefinita, al momento della creazione della tabella, la clausola è impostata su AUTO, il che significa che il server decide se l'indice deve essere pulito quando esegue il vacuum della tabella. È possibile impostare la clausola su ON per attivare la pulizia dell'indice per una tabella specifica o su OFF per disattivare la pulizia dell'indice per la tabella. Tieni presente che la disattivazione della pulizia dell'indice può far risparmiare tempo, ma può potenzialmente portare a un aumento delle dimensioni dell'indice.

È possibile controllare manualmente la pulizia dell'indice quando esegui il VACUUM di una tabella dalla linea di comando. Per eseguire il vacuum di una tabella e rimuovere le tuple inattive dagli indici, includi la clausola INDEX_CLEANUP con il valore ON e il nome della tabella:

```
acctg=> VACUUM (INDEX_CLEANUP ON) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Per eseguire il vacuum di una tabella senza pulire gli indici, specifica il valore OFF:

```
acctg=> VACUUM (INDEX_CLEANUP OFF) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Migliore gestione della memoria in Aurora PostgreSQL

Aurora PostgreSQL ora include funzionalità avanzate di gestione della memoria per ottimizzare le prestazioni e la resilienza del database con carichi di lavoro diversi. Questi miglioramenti aiutano Aurora PostgreSQL a mantenere disponibilità e reattività costanti, anche durante i periodi di elevata richiesta di memoria.

Questa funzionalità è disponibile e abilitata per impostazione predefinita nelle seguenti versioni di Aurora PostgreSQL per le istanze con provisioning:

- 15.3 e tutte le versioni secondarie successive
- 14.8 e versioni secondarie successive
- 13.11 e versioni secondarie successive
- 12.15 e versioni secondarie successive
- 11.20 e versioni secondarie successive

Questa funzionalità è disponibile e abilitata per impostazione predefinita nel seguente programma Aurora PostgreSQL per Aurora Serverless istanze:

- 16.3 e tutte le versioni secondarie successive
- 15.7 e tutte le versioni secondarie successive
- 14.12 e versioni secondarie successive
- 13.15 e versioni secondarie successive

Quando i carichi di lavoro dei clienti utilizzano tutta la memoria libera disponibile, il sistema operativo può riavviare il database per proteggere le risorse, con conseguente indisponibilità temporanea. I nuovi miglioramenti alla gestione della memoria in Aurora PostgreSQL annullano in modo proattivo determinate transazioni quando il sistema subisce un'elevata pressione della memoria, contribuendo a mantenere la stabilità del database.

Le caratteristiche principali di una migliore gestione della memoria sono le seguenti:

- Annullamento delle transazioni del database che richiedono più memoria quando il sistema sta raggiungendo un carico critico della memoria.
- Si dice che il sistema sia sottoposto a un carico critico della memoria quando esaurisce tutta la memoria fisica e sta per esaurire lo spazio di scambio. In queste circostanze, qualsiasi transazione che richiede memoria verrà annullata nel tentativo di ridurre immediatamente il carico della memoria nell'istanza database.
- Le utilità di avvio di PostgreSQL essenziali e i worker in background come i worker autovacuum sono sempre protetti.

Gestione dei parametri di gestione della memoria

Per attivare la gestione della memoria

Questa funzionalità è attivata per impostazione predefinita. Viene visualizzato un messaggio di errore, come quello illustrato nell'esempio seguente, quando una transazione viene annullata a causa di memoria insufficiente.

```
ERROR: out of memory Detail: Failed on request of size 16777216.
```

Per disattivare la gestione della memoria

Per disattivare questa funzionalità, connettiti al cluster Aurora PostgreSQL DB con psql e usa l'istruzione SET per i valori dei parametri come indicato di seguito.

> **Note**
>
> Ti consigliamo di mantenere abilitata la gestione della memoria. Questo aiuta a prevenire potenziali out-of-memory errori che potrebbero portare al riavvio del database indotto dal carico di lavoro a causa dell'esaurimento della memoria.

La tabella seguente mostra come disattivare la funzionalità di gestione della memoria per le diverse versioni di Aurora PostgreSQL:

Versioni Aurora PostgreSQL	Parametro	Predefinita	Comando per disattivare la gestione della memoria a livello di sessione
11,20, 11,21, 12,15, 12,16, 13,11, 13,12, 14,8, 14,9, 15,3, 15,4	`rds.memory_allocation_guard`	`false`	`SET rds.memory_allocation_guard = true;`
12.17, 13.13, 14.10, 15.5 e versioni successive	`rds.enable_memory_management`	`true`	`SET rds.enable_memory_management = false;`

> **Note**
>
> Il `rds.memory_allocation_guard` parametro è obsoleto in Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 e versioni successive.

L'impostazione dei valori di questi parametri nel gruppo di parametri del cluster DB impedisce l'annullamento delle query. Per ulteriori informazioni sul gruppo di parametri del cluster DB, vedere.

Limitazione

- Questa funzionalità non è supportata nella classe di istanze db.t3.medium.

Failover rapido con Amazon Aurora PostgreSQL

Di seguito, vengono fornite informazioni su come garantire che il failover si verifichi il più rapidamente possibile. Per ripristinare rapidamente dopo il failover, è possibile utilizzare la gestione della cache del cluster per il cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL.

Alcuni dei passaggi da eseguire affinché il failover venga eseguito rapidamente includono i seguenti:

- Impostare i keepalive TCP (Transmission Control Protocol) con intervalli temporali brevi, per interrompere le query in esecuzione più lunghe prima che scada il timeout di lettura in caso di errore.
- Impostare i timeout per la memorizzazione nella cache DNS (Domain Name System) Java in modo aggressivo. Questo garantisce che l'endpoint di sola lettura di Aurora possa scorrere correttamente nodi di sola lettura nei successivi tentativi di connessione.
- Impostare le variabili di timeout usate nella stringa di connessione JDBC più basse possibile. Utilizzare oggetti di connessione separati per query di breve e lunga durata.
- Utilizzare gli endpoint Aurora di lettura e scrittura forniti per connetterti al cluster.
- Utilizzare le operazioni API RDS per testare la risposta dell'applicazione in caso di errori lato server. Inoltre, utilizzare uno strumento di eliminazione pacchetti per testare la risposta dell'applicazione per gli errori lato client.
- Usa il driver AWS JDBC per sfruttare appieno le funzionalità di failover di Aurora PostgreSQL. Per ulteriori informazioni sul driver AWS JDBC e istruzioni complete per il suo utilizzo, consulta l'archivio dei driver JDBC di Amazon Web Services (AWS). GitHub

Questi sono descritti in modo dettagliato di seguito.

Argomenti

- Impostazione dei parametri di keepalive TCP
- Configurare l'applicazione per il Failover rapido
- Verifica del Failover
- Esempio di failover rapido in Java

Impostazione dei parametri di keepalive TCP

Quando si imposta una connessione TCP, ad essa viene associato un set di timer. Quando il timer keepalive raggiunge lo zero, un pacchetto di esplorazione keepalive viene inviato all'endpoint di connessione. Se la sonda riceve una risposta, si può presumere che la connessione sia ancora attiva e in esecuzione.

Attivare i parametri keepalive TCP e impostarli in modo aggressivo garantisce che se il client non è in grado di connettersi al database, le eventuali connessioni attive vengono chiuse rapidamente. L'applicazione può quindi connettersi a un nuovo endpoint.

I seguenti parametri keepalive TCP devono essere impostati:

- `tcp_keepalives_idle`controlla il tempo, in secondi, dopo il quale viene inviato un pacchetto keepalive quando nessun dato è stato inviato dal socket. ACKs non sono considerati dati. Consigliamo la seguente impostazione:

  ```
  tcp_keepalives_idle = 1
  ```

- `tcp_keepalives_interval` controlla il tempo, in secondi, tra l'invio di successivi pacchetti keepalive dopo l'invio del pacchetto iniziale. Imposta questo tempo utilizzando il parametro `tcp_keepalives_idle`. Consigliamo la seguente impostazione:

  ```
  tcp_keepalives_interval = 1
  ```

- `tcp_keepalives_count` è il numero di esplorazioni keepalive non riconosciute che si verificano prima che l'applicazione venga notificata. Consigliamo la seguente impostazione:

  ```
  tcp_keepalives_count = 5
  ```

Queste impostazioni dovrebbero notificare l'applicazione entro cinque secondi quando il database smette di rispondere. Se i pacchetti keepalive vengono spesso eliminati all'interno della rete dell'applicazione, è possibile impostare un valore `tcp_keepalives_count` più elevato. Questo consente di avere più buffer in reti meno affidabili, sebbene aumenti il tempo necessario per rilevare un errore effettivo.

Per impostare parametri keepalive TCP su Linux

1. Esegui il test della modalità di configurazione dei parametri keepalive TCP.

   A questo scopo, utilizza la riga di comando con i seguenti comandi. Questa è una configurazione consigliata a livello di sistema. In altre parole, influenza anche tutte le altre applicazioni che creano socket con l'opzione `SO_KEEPALIVE` attivata.

   ```
   sudo sysctl net.ipv4.tcp_keepalive_time=1
   sudo sysctl net.ipv4.tcp_keepalive_intvl=1
   sudo sysctl net.ipv4.tcp_keepalive_probes=5
   ```

2. Una volta trovata una configurazione che funziona per l'applicazione, mantenere le impostazioni aggiungendo le seguenti righe a `/etc/sysctl.conf`, comprese le eventuali modifiche apportate:

   ```
   tcp_keepalive_time = 1
   tcp_keepalive_intvl = 1
   ```

```
tcp_keepalive_probes = 5
```

Configurare l'applicazione per il Failover rapido

Di seguito, è disponibile una discussione su diverse modifiche di configurazione per Aurora PostgreSQL che è possibile apportare per un failover rapido. Per ulteriori informazioni sulla configurazione e la configurazione del driver JDBC PostgreSQL, vedere la documentazione PostgreSQL JDBC Driver.

Argomenti

- Ridurre i timeout della cache DNS
- Impostare una stringa di connessione Aurora PostgreSQL per un Failover rapido
- Altre opzioni per ottenere la stringa host

Ridurre i timeout della cache DNS

Quando l'applicazione prova a stabilire una connessione dopo un failover, il nuovo scrittore Aurora PostgreSQL sarà un lettore precedente Per individuarlo, utilizzare l'endpoint di sola lettura Aurora prima della completa propagazione degli aggiornamenti DNS. Impostare Time to Live (TTL) java DNS su un valore basso, ad esempio meno di 30 secondi, consente di effettuare il ciclo tra i nodi del lettore nei successivi tentativi di connessione.

```
// Sets internal TTL to match the Aurora RO Endpoint TTL
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
// If the lookup fails, default to something like small to retry
java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
```

Impostare una stringa di connessione Aurora PostgreSQL per un Failover rapido

Per usare il failover rapido di Aurora PostgreSQL, accertarsi che la stringa di connessione dell'applicazione contenga un elenco di host anziché un singolo host. Di seguito è riportata una stringa di connessione di esempio che può essere utilizzata per connettersi a un cluster Aurora PostgreSQL: In questo esempio, gli host sono in grassetto.

```
jdbc:postgresql://myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
/postgres?user=<primaryuser>&password=<primarypw>&loginTimeout=2
```

```
&connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60
&tcpKeepAlive=true&targetServerType=primary
```

Per una migliore disponibilità e per evitare una dipendenza dall'API RDS, è opportuno mantenere un file con cui eseguire la connessione. Questo file contiene una stringa host che viene letta dall'applicazione quando si stabilisce una connessione al database. Questa stringa host contiene tutti gli endpoint Aurora disponibili per il cluster. Per altre informazioni sugli endpoint Aurora, consultare Connessioni endpoint Amazon Aurora.

Ad esempio, potrebbe essere necessario archiviare gli endpoint in un file locale come mostrato di seguito.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

L'applicazione legge da questo file per compilare la sezione host della stringa di connessione JDBC. Rinominare il cluster DB fa sì che questi endpoint siano modificati. Assicurati che l'applicazione gestisca questo evento nel caso in cui si verifichi.

Un'altra opzione è usare un elenco di nodi di istanze database, come illustrato di seguito.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node3.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node4.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

Il vantaggio di questo approccio è che il driver di connessione PostgreSQL JDBC esegue un ciclo di tutti i nodi in questo elenco per individuare una connessione valida. Al contrario, quando si usano gli endpoint Aurora, vengono provati solo due nodi per ogni tentativo di connessione. Tuttavia, l'utilizzo dei nodi di istanza database presenta uno svantaggio. Se si aggiungono o si rimuovono nodi dal cluster e l'elenco degli endpoint di istanza diventa obsoleto, il driver di connessione potrebbe non trovare mai l'host corretto cui connettersi.

Per garantire che l'applicazione non rimanga in attesa troppo a lungo prima di connettersi a un host, imposta i seguenti parametri in modo aggressivo.

- `targetServerType`: controlla se il driver si connette a un nodo di scrittura o lettura. Per garantire che le applicazioni si riconnetteranno solo a un nodo di scrittura, imposta il valore `targetServerType` su `primary`.

I valori per il parametro `targetServerType` includono `primary`, `secondary`, `any` e `preferSecondary`. Il valore `preferSecondary` esegue prima un tentativo per stabilire una connessione a un lettore. Se non è possibile stabilire una connessione con il lettore, si connette allo scrittore.

- `loginTimeout`: controlla per quanto tempo l'applicazione rimane in attesa prima di accedere al database dopo che è stata stabilita una connessione socket.
- `connectTimeout` – Controlla quanto tempo il connettore attende per stabilire una connessione con il database.

In base al livello di aggressività desiderato per l'applicazione, puoi modificare altri parametri dell'applicazione per accelerare il processo di connessione:

- `cancelSignalTimeout`: in alcune applicazioni, potrebbe essere necessario inviare un segnale di annullamento "massimo sforzo" su una query scaduta. Se questo segnale di annullamento si trova nel percorso di failover, valuta se impostarlo in modo aggressivo per evitare che venga inviato a un host morto.
- `socketTimeout` – Questo parametro controlla quanto tempo il connettore attende per le operazioni di lettura. Questo parametro può essere usato come un "timeout di query" globale per assicurare che nessuna query attenda più a lungo di questo valore. Una buona pratica è quella di disporre di due gestori di connessione. Un gestore di connessione che esegue query di breve durata e imposta questo valore più basso. Un altro gestore di connessione, per query a esecuzione prolungata, con questo valore impostato molto più alto. Con questo approccio, è possibile fare affidamento sui parametri keepalive TCP per interrompere le query a esecuzioni prolungata se il server non funziona.
- `tcpKeepAlive`: attivare questo parametro per garantire che i parametri keepalive TCP impostati siano rispettati.
- `loadBalanceHosts` – Quando è impostato su `true`, l'applicazione del parametro si connette a un host casuale scelto da un elenco di host candidati.

Altre opzioni per ottenere la stringa host

È possibile ottenere la stringa host da diverse fonti, compresa la funzione `aurora_replica_status` e usando l'API di Amazon RDS.

In molti casi, occorre determinare chi è lo scrittore del cluster o trovare altri nodi di lettura nel cluster. A questo scopo, l'applicazione può connettersi a qualsiasi istanza database nel cluster database ed eseguire la query della funzione `aurora_replica_status`. Questa funzione può essere utilizzata per ridurre il tempo necessario per trovare un host cui connettersi. Tuttavia, in alcuni scenari di errore di rete, la `aurora_replica_status` funzione potrebbe mostrare informazioni incomplete out-of-date o incomplete.

Un buon modo per assicurarsi che l'applicazione possa individuare un nodo cui connettersi è provare a connettersi all'endpoint di scrittura del cluster e quindi all'endpoint di lettura del cluster, fino a quando non è possibile stabilire una connessione leggibile. Questi endpoint non cambiano a meno che il cluster database non venga rinominato. Quindi, in genere possono essere lasciati come membri statici dell'applicazione o archiviati in un file di risorsa che viene letto dall'applicazione.

Una volta stabilita una connessione usando uno di questi endpoint, è possibile ottenere informazioni sul resto del cluster. A questo scopo, richiama la funzione `aurora_replica_status`. Ad esempio, il comando seguente recupera informazioni con `aurora_replica_status`.

```
postgres=> SELECT server_id, session_id, highest_lsn_rcvd, cur_replay_latency_in_usec,
 now(), last_update_timestamp
FROM aurora_replica_status();

server_id | session_id | highest_lsn_rcvd | cur_replay_latency_in_usec | now |
 last_update_timestamp
-----------+--------------------------------------+-----------------
+---------------------------+-------------------------------+-----------------------
mynode-1 | 3e3c5044-02e2-11e7-b70d-95172646d6ca | 594221001 | 201421 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-2 | 1efdd188-02e4-11e7-becd-f12d7c88a28a | 594221001 | 201350 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-3 | MASTER_SESSION_ID | | | 2017-03-07 19:50:24.695322+00 | 2017-03-07
 19:50:23+00
(3 rows)
```

Ad esempio, la sezione host della stringa di connessione potrebbe iniziare con entrambi gli endpoint del cluster di scrittura e di lettura, come mostrato di seguito.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

In questo scenario, l'applicazione prova a stabilire una connessione a qualsiasi tipo di nodo, primario o secondario. Una volta connessa, una buona prassi è quella di esaminare innanzitutto lo stato di lettura-scrittura del nodo. A questo scopo, occorre eseguire la query del risultato del comando `SHOW transaction_read_only`.

Se il valore restituito della query è `OFF`, allora la connessione al nodo primario è andata a buon fine. Tuttavia, si supponga che il valore restituito sia `ON` e che l'applicazione richieda una connessione di lettura/scrittura. In questo caso, si può richiamare la funzione `aurora_replica_status` per determinare il `server_id` che ha `session_id='MASTER_SESSION_ID'`. Questa funzione fornisce il nome del nodo primario. Ciò è possibile il `endpointPostfix` descritto di seguito.

È importante sapere quando si esegue una connessione a una replica con dati obsoleti. Quando ciò accade, la `aurora_replica_status` funzione potrebbe mostrare out-of-date informazioni. Una soglia di obsolescenza può essere impostata a livello di applicazione. Per la verifica, osservare la differenza tra l'ora del server e il valore `last_update_timestamp`. In generale, l'applicazione dovrebbe evitare di alternarsi tra due host a causa delle informazioni in conflitto restituite dalla funzione `aurora_replica_status`. L'applicazione dovrebbe provare prima tutti gli host conosciuti anziché seguire i dati restituiti da `aurora_replica_status`.

Elenco delle istanze utilizzando l'operazione Descrivi DBClusters API, esempio in Java

È possibile trovare l'elenco delle istanze a livello di codice utilizzando, in particolare AWS SDK per Java, l'operazione Descrivi API. DBClusters

Di seguito è riportato un piccolo esempio di come eseguire questa operazione in Java 8:

```
AmazonRDS client = AmazonRDSClientBuilder.defaultClient();
DescribeDBClustersRequest request = new DescribeDBClustersRequest()
   .withDBClusterIdentifier(clusterName);
DescribeDBClustersResult result =
rdsClient.describeDBClusters(request);

DBCluster singleClusterResult = result.getDBClusters().get(0);

String pgJDBCEndpointStr =
singleClusterResult.getDBClusterMembers().stream()
   .sorted(Comparator.comparing(DBClusterMember::getIsClusterWriter)
   .reversed()) // This puts the writer at the front of the list
   .map(m -> m.getDBInstanceIdentifier() + endpointPostfix + ":" +
 singleClusterResult.getPort()))
   .collect(Collectors.joining(","));
```

Qui, `pgJDBCEndpointStr` contiene un elenco formattato di endpoint, come mostrato di seguito.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

La variabile `endpointPostfix` può essere una costante impostata dall'applicazione. Oppure, può essere ottenuta dall'applicazione eseguendo la query dell'operazione API `DescribeDBInstances` per una singola istanza del cluster. Questo valore rimane costante all'interno di un Regione AWS e per un singolo cliente. Pertanto, salva una chiamata API per mantenere semplicemente questa costante in un file di risorsa letto dall'applicazione. Nell'esempio precedente, è impostato sul valore seguente.

```
.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com
```

Ai fini della disponibilità, una buona norma è quella di usare per impostazione predefinita gli endpoint Aurora del cluster database se l'API non risponde o impiega troppo tempo a rispondere. Viene garantito l'aggiornamento degli endpoint nel tempo necessario per aggiornare il record DNS. L'aggiornamento del record DNS con un endpoint richiede in genere meno di 30 secondi. L'endpoint può essere memorizzato in un file di risorse utilizzato dall'applicazione.

Verifica del Failover

In tutti i casi è necessario disporre di un cluster di database contenente due o più istanze database.

Dal lato server, alcune operazioni API possono causare un'interruzione che può essere usata per testare la risposta delle applicazioni:

- Failover DBCluster: questa operazione tenta di promuovere una nuova istanza DB nel cluster DB a masterizzatore.

  Il seguente esempio di codice mostra come è possibile utilizzare `failoverDBCluster` per causare un'interruzione. Per ulteriori dettagli sulla configurazione di un client Amazon RDS, consulta Using the AWS SDK for Java.

  ```
  public void causeFailover() {

      final AmazonRDS rdsClient = AmazonRDSClientBuilder.defaultClient();

      FailoverDBClusterRequest request = new FailoverDBClusterRequest();
      request.setDBClusterIdentifier("cluster-identifier");
  ```

```
    rdsClient.failoverDBCluster(request);
}
```

- RiavvioDBInstance: il failover non è garantito con questa operazione API. Tuttavia, arresta il database dello scrittore. Può essere utilizzata per testare la risposta dell'applicazione all'eliminazione delle connessioni. Il parametro `ForceFailover` non si applica ai motori Aurora. Utilizzare invece l'operazione API `FailoverDBCluster`.
- Modifica DBCluster: la modifica del `Port` parametro causa un'interruzione quando i nodi del cluster iniziano ad ascoltare su una nuova porta. In generale, l'applicazione può rispondere innanzitutto a questo errore assicurandosi che solo l'applicazione controlli le modifiche alle porte. Inoltre, accertarsi che possa aggiornare in modo appropriato gli endpoint da cui dipende. A questo scopo, è necessario che qualcuno aggiorni manualmente la porta quando vengono apportate modifiche a livello di API. Oppure, è possibile utilizzare l'API RDS nell'applicazione per determinare se la porta è cambiata.
- Modifica DBInstance: la modifica del `DBInstanceClass` parametro causa un'interruzione.
- Elimina DBInstance: l'eliminazione del primario (writer) fa sì che una nuova istanza DB venga promossa a writer nel cluster DB.

Dal lato applicazione o client, se si usa Linux, è possibile testare in che modo l'applicazione risponde a improvvise eliminazioni di pacchetti. Ciò può essere fatto a seconda che porta, host o se i pacchetti keepalive TCP vengono inviati o ricevuti utilizzando il comando iptables.

Esempio di failover rapido in Java

Il seguente esempio di codice mostra come un'applicazione potrebbe configurare un gestore di driver di Aurora PostgreSQL.

L'applicazione richiama la funzione `getConnection` quando è richiesta una connessione. Una chiamata a `getConnection` può non riuscire a trovare un host valido. Un esempio è quando non viene trovato alcuno scrittore ma il parametro `targetServerType` è impostato su `primary`. In questo caso, l'applicazione chiamante deve semplicemente riprovare a richiamare la funzione.

Per evitare di spingere il comportamento dei tentativi continui nell'applicazione, puoi impacchettare questo tentativo di chiamata in un pool di connessioni. Con la maggior parte dei pool di connessioni, puoi specificare una stringa di connessione JDBC. Pertanto l'applicazione può richiamare `getJdbcConnectionString` e passarla al pool di connessioni. Ciò significa che con Aurora PostgreSQL si può utilizzare un failover più rapido.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import org.joda.time.Duration;

public class FastFailoverDriverManager {
   private static Duration LOGIN_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CONNECT_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CANCEL_SIGNAL_TIMEOUT = Duration.standardSeconds(1);
   private static Duration DEFAULT_SOCKET_TIMEOUT = Duration.standardSeconds(5);

   public FastFailoverDriverManager() {
       try {
            Class.forName("org.postgresql.Driver");
       } catch (ClassNotFoundException e) {
            e.printStackTrace();
       }

       /*
         * RO endpoint has a TTL of 1s, we should honor that here. Setting this
 aggressively makes sure that when
         * the PG JDBC driver creates a new connection, it will resolve a new different
 RO endpoint on subsequent attempts
         * (assuming there is > 1 read node in your cluster)
         */
        java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
       // If the lookup fails, default to something like small to retry
       java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
   }

   public Connection getConnection(String targetServerType) throws SQLException {
       return getConnection(targetServerType, DEFAULT_SOCKET_TIMEOUT);
   }

   public Connection getConnection(String targetServerType, Duration queryTimeout)
 throws SQLException {
```

```
       Connection conn =
DriverManager.getConnection(getJdbcConnectionString(targetServerType, queryTimeout));

      /*
        * A good practice is to set socket and statement timeout to be the same thing
since both
        * the client AND server will stop the query at the same time, leaving no
running queries
        * on the backend
        */
       Statement st = conn.createStatement();
       st.execute("set statement_timeout to " + queryTimeout.getMillis());
       st.close();

      return conn;
  }

  private static String urlFormat = "jdbc:postgresql://%s"
          + "/postgres"
          + "?user=%s"
          + "&password=%s"
          + "&loginTimeout=%d"
          + "&connectTimeout=%d"
          + "&cancelSignalTimeout=%d"
          + "&socketTimeout=%d"
          + "&targetServerType=%s"
          + "&tcpKeepAlive=true"
          + "&ssl=true"
          + "&loadBalanceHosts=true";
  public String getJdbcConnectionString(String targetServerType, Duration
queryTimeout) {
      return String.format(urlFormat,
                getFormattedEndpointList(getLocalEndpointList()),
                CredentialManager.getUsername(),
                CredentialManager.getPassword(),
                LOGIN_TIMEOUT.getStandardSeconds(),
                CONNECT_TIMEOUT.getStandardSeconds(),
                CANCEL_SIGNAL_TIMEOUT.getStandardSeconds(),
                queryTimeout.getStandardSeconds(),
                targetServerType
      );
  }

  private List<String> getLocalEndpointList() {
```

```
        /*
          * As mentioned in the best practices doc, a good idea is to read a local
 resource file and parse the cluster endpoints.
          * For illustration purposes, the endpoint list is hardcoded here
          */
         List<String> newEndpointList = new ArrayList<>();
         newEndpointList.add("myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");
         newEndpointList.add("myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");

        return newEndpointList;
    }

    private static String getFormattedEndpointList(List<String> endpoints) {
        return IntStream.range(0, endpoints.size())
                .mapToObj(i -> endpoints.get(i).toString())
                .collect(Collectors.joining(","));
    }
}
```

Ripristino rapido dopo il failover con Cluster Cache Management per Aurora PostgreSQL

Per il ripristino rapido dell'istanza database di scrittura nei cluster Aurora PostgreSQL in caso di failover, utilizzare la gestione della cache del cluster per Amazon Aurora PostgreSQL. La gestione della cache del cluster garantisce il mantenimento delle prestazioni dell'applicazione in caso di failover.

In una tipica situazione di failover, è possibile che si verifichi un degrado delle prestazioni temporaneo ma di grandi dimensioni dopo il failover. Questo degrado si verifica perché all'avvio dell'istanza database di failover, la cache del buffer è vuota. Una cache vuota è anche nota col nome di cache fredda. Una cache fredda degrada le prestazioni perché l'istanza database deve leggere dal disco a una velocità minore, invece di sfruttare i valori memorizzati nella cache del buffer.

Con la gestione della cache del cluster, si imposta una specifica istanza database di lettura come destinazione del failover. La gestione della cache del cluster garantisce che i dati nella cache dell'istanza di lettura designata siano mantenuti sincronizzati con i dati nella cache dell'istanza database di scrittura. La cache dell'istanza di lettura designata pre-popolata con i valori prende il nome di cache calda. Se si verifica un failover, l'istanza di lettura designata utilizza immediatamente

i valori presenti nella sua cache calda nel momento in cui viene promossa al ruolo di nuova istanza database di scrittura. Questo approccio garantisce all'applicazione prestazioni di recupero decisamente migliori.

La gestione della cache del cluster richiede che l'istanza del lettore designata disponga dello stesso tipo di classe di istanza e dimensione (`db.r5.2xlarge` o `db.r5.xlarge`, ad esempio) di scrittura. Tieni presente questo aspetto quando crei cluster database Aurora PostgreSQL in modo che il cluster possa essere ripristinato durante un failover. Per un elenco dei tipi e delle dimensioni delle classi di istanza, consulta Specifiche hardware per le classi di istanza database per Aurora.

> **Note**
>
> La gestione della cache del cluster non è supportata per i cluster DB secondari Aurora PostgreSQL che fanno parte dei database globali Aurora. Per i cluster primari in cui è supportata questa funzionalità, si consiglia di non eseguire alcun carico di lavoro sul lettore tier-0 designato.

Indice

- Configurazione della gestione della cache del cluster
  - Abilitazione della gestione della cache del cluster
  - Impostazione della priorità del livello di promozione per l'istanza database di scrittura
  - Impostazione della priorità del livello di promozione per un'istanza database di lettura
- Monitoraggio della cache
- Risoluzione dei problemi relativi alla configurazione CCM

Configurazione della gestione della cache del cluster

Per configurare la gestione della cache del cluster, eseguire i processi seguenti in ordine.

Argomenti

- Abilitazione della gestione della cache del cluster
- Impostazione della priorità del livello di promozione per l'istanza database di scrittura
- Impostazione della priorità del livello di promozione per un'istanza database di lettura

> **Note**
>
> Dopo aver completato queste fasi, attendi almeno 1 minuto affinché la gestione della cache del cluster sia completamente operativa.

Abilitazione della gestione della cache del cluster

Per abilitare la gestione della cache del cluster, attenersi alla procedura descritta di seguito.

Console

Per abilitare la gestione della cache del cluster

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco, selezionare il gruppo di parametri per il cluster di database di Aurora PostgreSQL.

   Il cluster di database deve utilizzare un gruppo di parametri diverso da quello predefinito, poiché non è possibile modificare i valori in un gruppo di parametri predefinito.
4. Per Parameter group actions (Operazioni del gruppo di parametri), scegliere Edit (Modifica).
5. Impostare il valore del parametro del cluster `apg_ccm_enabled` su 1.
6. Seleziona Save changes (Salva modifiche).

AWS CLI

Per abilitare la gestione della cache del cluster per un cluster Aurora PostgreSQL DB, usa il comando con i seguenti parametri obbligatori: AWS CLI `modify-db-cluster-parameter-group`

- `--db-cluster-parameter-group-name`
- `--parameters`

Example

PerLinux, o: macOS Unix

```
aws rds modify-db-cluster-parameter-group \
```

```
    --db-cluster-parameter-group-name my-db-cluster-parameter-group \
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my-db-cluster-parameter-group ^
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Impostazione della priorità del livello di promozione per l'istanza database di scrittura

Per la gestione della cache del cluster, assicurarsi che la priorità della promozione sia tier-0 per l'istanza database di scrittura del cluster di database di Aurora PostgreSQL. Il livello della priorità di promozione è un valore che specifica l'ordine in cui un'istanza di lettura di Aurora viene promossa al ruolo di istanza database di scrittura dopo un errore. I valori validi sono 0–15, dove 0 è la priorità massima e 15 è la priorità minima. Per ulteriori informazioni sul livello di promozione, consultare Tolleranza ai guasti di un cluster DB Aurora.

Console

Per impostare la priorità di promozione per l'istanza database di scrittura su tier-0

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere l'istanza database Writer (di scrittura) del cluster di database di Aurora PostgreSQL.
4. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
5. Nel pannello Additional configuration (Configurazione aggiuntiva) scegliere tier-0 (livello 0) per la Failover priority (Priorità failover).
6. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
7. Per applicare immediatamente le modifiche dopo il salvataggio, scegliere Apply immediately (Applica immediatamente).
8. Scegliere Modify DB Instance (Modifica istanza database) per salvare le modifiche.

AWS CLI

Per impostare la priorità del livello di promozione su 0 per l'istanza Writer DB che utilizza il AWS CLI, chiama il modify-db-instancecomando con i seguenti parametri richiesti:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-instance \
    --db-instance-identifier writer-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier writer-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Impostazione della priorità del livello di promozione per un'istanza database di lettura

È necessario impostare solo un'istanza Reader DB per la gestione della cache del cluster. Per farlo, scegli un lettore dal cluster Aurora PostgreSQL che sia la stessa classe di istanza e che abbia le stesse dimensioni dell'istanza database di scrittura. Ad esempio, se lo scrittore utilizza `db.r5.xlarge`, scegliere un lettore che utilizzi lo stesso tipo di classe di istanza e la stessa dimensione. Quindi impostare il livello della priorità di promozione su 0.

Il livello della priorità di promozione è un valore che specifica l'ordine in cui un'istanza di lettura di Aurora viene promossa al ruolo di istanza database di scrittura dopo un errore. I valori validi sono compresi nell'intervallo tra 0 e 15, dove 0 è la priorità massima e 15 è la priorità minima.

Console

Per impostare la priorità di promozione per l'istanza database di lettura su tier-0

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere l'istanza database Reader (Di lettura) del cluster di database di Aurora PostgreSQL che sia della stessa classe di istanza dell'istanza database di scrittura.
4. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB Instance (Modifica istanza database).
5. Nel pannello Additional configuration (Configurazione aggiuntiva) scegliere tier-0 (livello 0) per la Failover priority (Priorità failover).
6. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
7. Per applicare immediatamente le modifiche dopo il salvataggio, scegliere Apply immediately (Applica immediatamente).
8. Scegliere Modify DB Instance (Modifica istanza database) per salvare le modifiche.

AWS CLI

Per impostare la priorità del livello di promozione su 0 per l'istanza Reader DB utilizzando il AWS CLI, chiama il modify-db-instancecomando con i seguenti parametri obbligatori:

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-instance \
    --db-instance-identifier reader-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier reader-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Monitoraggio della cache

Dopo aver configurato la gestione della cache del cluster, è possibile monitorare lo stato di sincronizzazione tra la cache del buffer dell'istanza database di scrittura e la cache del buffer calda di scrittura designata. Per esaminare il contenuto della cache del buffer sia sull'istanza database di scrittura sia sull'istanza database di lettura designata, utilizza il modulo PostgreSQL `pg_buffercache`. Per ulteriori informazioni, consulta la documentazione di PostgreSQL `pg_buffercache`.

Utilizzo della funzione **`aurora_ccm_status`**

La gestione della cache del cluster mette a disposizione anche la funzione `aurora_ccm_status`. Utilizza la funzione `aurora_ccm_status` sull'istanza database di scrittura per ottenere le seguenti informazioni sull'avanzamento del popolamento della cache sull'istanza database di lettura designata:

- `buffers_sent_last_minute` – Quanti buffer sono stati inviati all'istanza database di lettura designata nell'ultimo minuto.
- `buffers_found_last_minute`: il numero di buffer con accesso frequente identificati durante l'ultimo minuto.
- `buffers_sent_last_scan` – Quanti buffer sono stati inviati all'istanza database di lettura designata durante l'ultima scansione completa della cache.
- `buffers_found_last_scan` – Quanti buffer sono stati identificati come soggetti ad accesso frequente ed è stato necessario inviare durante l'ultima scansione completa della cache. I buffer già memorizzati sull'istanza database di lettura designata non vengono inviati.
- `buffers_sent_current_scan` – Quanti buffer sono stati inviati finora durante la scansione corrente.
- `buffers_found_current_scan` – Quanti buffer sono stati identificati come soggetti ad accesso frequente nella scansione corrente.
- `current_scan_progress` – Quanti buffer sono stati analizzati finora durante la scansione corrente.

L'esempio seguente mostra come utilizzare la funzione `aurora_ccm_status` per convertire parte del suo output in velocità e percentuale di popolamento della cache calda.

```
SELECT buffers_sent_last_minute*8/60 AS warm_rate_kbps,
   100*(1.0-buffers_sent_last_scan::float/buffers_found_last_scan) AS warm_percent
   FROM aurora_ccm_status();
```

Risoluzione dei problemi relativi alla configurazione CCM

Quando si abilita il parametro `apg_ccm_enabled` cluster, la gestione della cache del cluster viene attivata automaticamente a livello di istanza sull'istanza DB writer e su un'istanza DB reader sul cluster Aurora PostgreSQL DB. L'istanza writer e reader devono utilizzare lo stesso tipo e dimensione della classe di istanza. La priorità del loro livello di promozione è impostata su`0`. Le altre istanze di lettura nel cluster DB devono avere un livello di promozione diverso da zero e la gestione della cache del cluster è disabilitata per tali istanze.

I seguenti motivi possono causare problemi nella configurazione e disabilitare la gestione della cache del cluster:

- Quando non esiste un'istanza DB a lettore singolo impostata sul livello di promozione 0.
- Quando l'istanza DB Writer non è impostata sul livello di promozione 0.
- Quando più di un'istanza DB Reader sono impostate sul livello di promozione 0.
- Quando le istanze DB di Writer e One Reader con livello di promozione 0 non hanno le stesse dimensioni.

Gestione dell’abbandono della connessione Aurora PostgreSQL con pooling

In caso di connessione e disconnessione frequente delle applicazioni client, è possibile che il tempo di risposta del cluster database Aurora PostgreSQL rallenti. In questo caso si dice che nel cluster si verifica un abbandono della connessione. Ogni nuova connessione all'endpoint del cluster database Aurora PostgreSQL consuma risorse, riducendo pertanto quelle che possono essere utilizzate per elaborare il carico di lavoro effettivo. È consigliabile gestire il problema dell’abbandono della connessione seguendo alcune delle best practice descritte di seguito.

Per prima cosa, i tempi di risposta sui cluster database Aurora PostgreSQL che hanno frequenze elevate di abbandono della connessione possono essere migliorati. A questo scopo, puoi utilizzare un pool di connessioni come RDS Proxy. Un pool di connessioni fornisce una cache di connessioni

pronte all'uso per i client. Quasi tutte le versioni di Aurora PostgreSQL supportano RDS Proxy. Per ulteriori informazioni, consulta Amazon RDS Proxy con Aurora PostgreSQL.

Se la tua versione specifica di Aurora PostgreSQL non supporta RDS Proxy, puoi utilizzare un altro pool di connessioni compatibile con PostgreSQL, ad esempio. PgBouncer Per PgBouncerulteriori informazioni, consulta il sito Web.

Per verificare se il cluster database Aurora PostgreSQL può trarne vantaggio dal pooling delle connessioni, puoi controllare il file `postgresql.log` per verificare la presenza di connessioni e disconnessioni. Puoi anche utilizzare Performance Insights per individuare il numero di abbandoni della connessione che si verificano nel cluster database Aurora PostgreSQL. Di seguito sono disponibili informazioni su entrambi gli argomenti.

Registrazione di connessioni e disconnessioni

I parametri `log_connections` e `log_disconnections` PostgreSQL possono acquisire connessioni e disconnessioni all'istanza di scrittura del cluster database Aurora PostgreSQL. Per impostazione predefinita, questi parametro sono disattivati. Per attivare questi parametri, utilizza un gruppo di parametri personalizzati e attivali modificando il valore in 1. Per ulteriori informazioni sui gruppi di parametri, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora. Per verificare le impostazioni, esegui la connessione all'endpoint del cluster database per Aurora PostgreSQL utilizzando psql ed esegui le query come segue.

```
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_connections';
 setting
---------
on
(1 row)
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_disconnections';
setting
---------
on
(1 row)
```

Con entrambi questi parametri attivati, il registro acquisisce tutte le nuove connessioni e disconnessioni. Per ogni nuova connessione autorizzata, vengono visualizzati utente e database. Al momento della disconnessione, viene registrata anche la durata della sessione, come mostrato nell'esempio seguente.

```
2022-03-07 21:44:53.978 UTC [16641] LOG: connection authorized: user=labtek
 database=labdb application_name=psql
2022-03-07 21:44:55.718 UTC [16641] LOG: disconnection: session time: 0:00:01.740
 user=labtek database=labdb host=[local]
```

Per verificare l'abbandono della connessione nell'applicazione, attiva questi parametri se non è già stato fatto. Quindi raccogli i dati nel registro PostgreSQL per l'analisi eseguendo l'applicazione con un carico di lavoro e un periodo di tempo realistici. Puoi visualizzare i file di registro nella console RDS. Scegli l'istanza di scrittura del cluster database Aurora PostgreSQL, quindi seleziona la scheda Logs & events (Log ed eventi). Per ulteriori informazioni, consulta Visualizzazione ed elenco dei file di log del database.

Oppure, puoi scaricare il file di registro dalla console e utilizzare la seguente sequenza di comandi. Questa sequenza individua il numero totale di connessioni autorizzate e interrotte al minuto.

```
grep "connection authorized\|disconnection: session time:"
 postgresql.log.2022-03-21-16|\
awk {'print $1,$2}' |\
sort |\
uniq -c |\
sort -n -k1
```

Nell'output di esempio, è possibile visualizzare un picco nelle connessioni autorizzate seguito da disconnessioni a partire dalle 16:12:10.

```
.....
,.......
.........
5 2022-03-21 16:11:55 connection authorized:
9 2022-03-21 16:11:55 disconnection: session
5 2022-03-21 16:11:56 connection authorized:
5 2022-03-21 16:11:57 connection authorized:
5 2022-03-21 16:11:57 disconnection: session
32 2022-03-21 16:12:10 connection authorized:
30 2022-03-21 16:12:10 disconnection: session
31 2022-03-21 16:12:11 connection authorized:
27 2022-03-21 16:12:11 disconnection: session
27 2022-03-21 16:12:12 connection authorized:
27 2022-03-21 16:12:12 disconnection: session
41 2022-03-21 16:12:13 connection authorized:
47 2022-03-21 16:12:13 disconnection: session
```

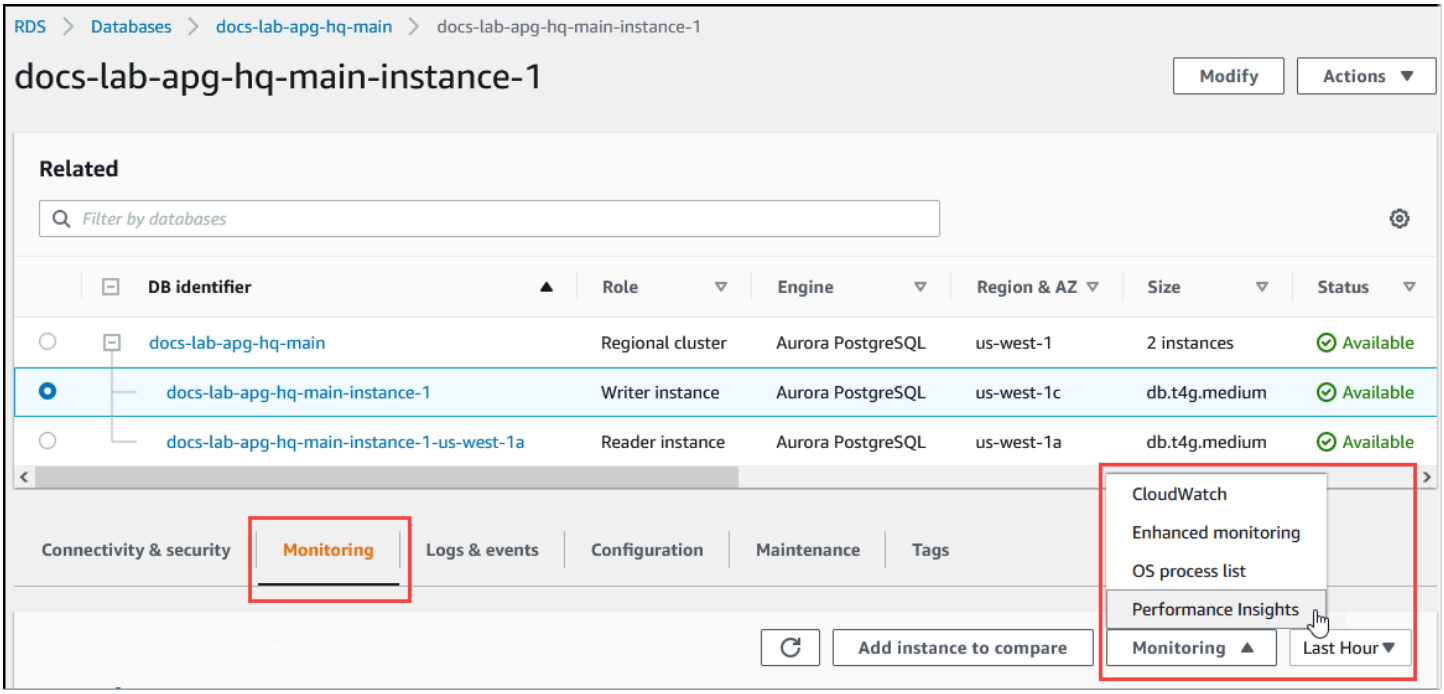
```
46 2022-03-21 16:12:14 connection authorized:
41 2022-03-21 16:12:14 disconnection: session
24 2022-03-21 16:12:15 connection authorized:
29 2022-03-21 16:12:15 disconnection: session
28 2022-03-21 16:12:16 connection authorized:
24 2022-03-21 16:12:16 disconnection: session
40 2022-03-21 16:12:17 connection authorized:
42 2022-03-21 16:12:17 disconnection: session
40 2022-03-21 16:12:18 connection authorized:
40 2022-03-21 16:12:18 disconnection: session
.....
,.......
.........
1 2022-03-21 16:14:10 connection authorized:
1 2022-03-21 16:14:10 disconnection: session
1 2022-03-21 16:15:00 connection authorized:
1 2022-03-21 16:16:00 connection authorized:
```

Con queste informazioni, puoi decidere se il carico di lavoro può trarre vantaggio da un pool di connessioni. Per analisi più dettagliate, puoi utilizzare Performance Insights.

Rilevamento dell’abbandono della connessione con Performance Insights

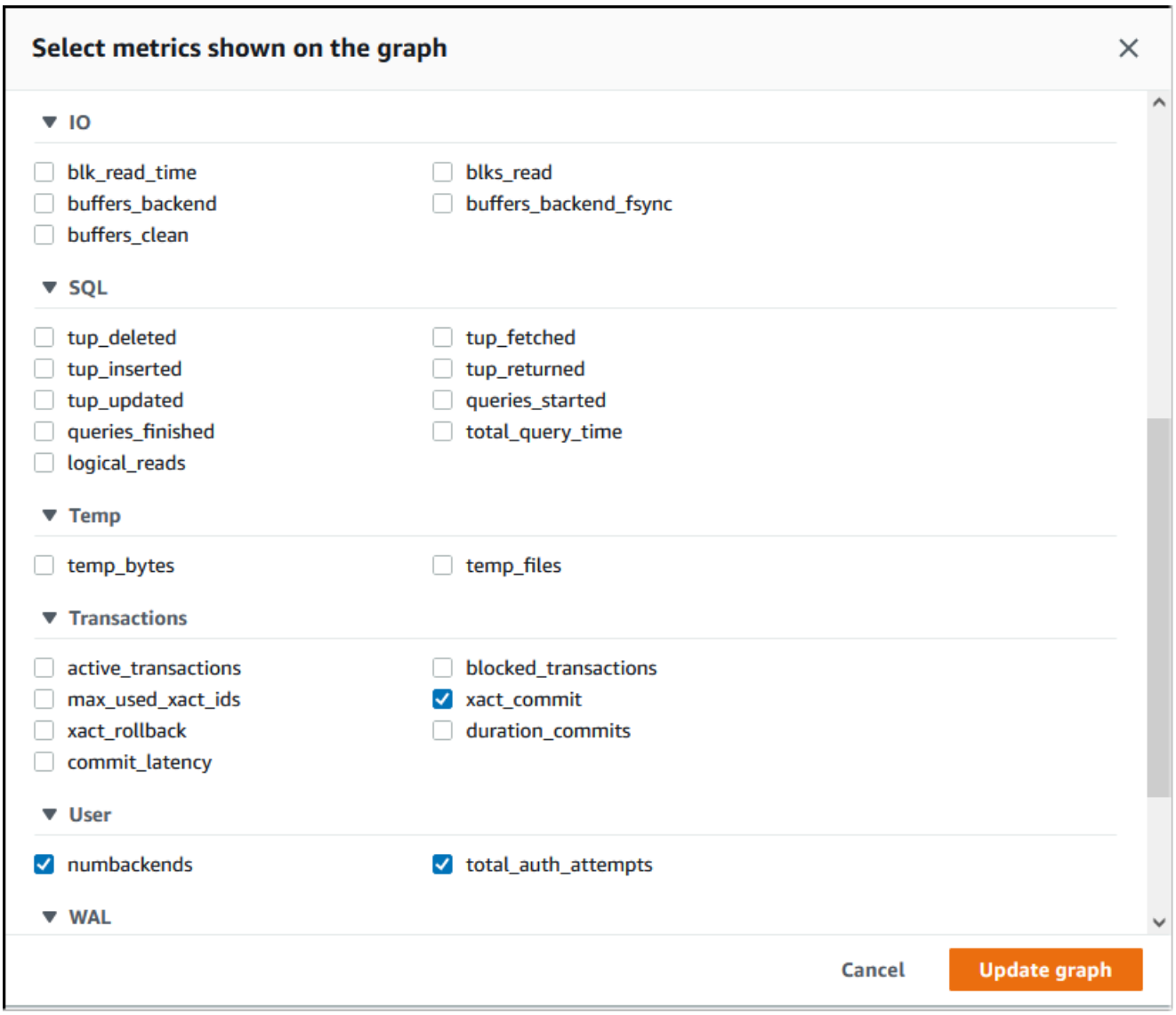
Puoi utilizzare Performance Insights per valutare il numero di abbandoni della connessione sul cluster database Aurora edizione compatibile con PostgreSQL. Durante la creazione di un cluster database Aurora PostgreSQL, l'impostazione per Performance Insights è attivata per impostazione predefinita. Se questa scelta è stata deselezionata al momento della creazione del cluster database, modifica il cluster per attivare la funzionalità. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

Con Performance Insights in esecuzione sul cluster database Aurora PostgreSQL, puoi scegliere le metriche che desideri monitorare. Puoi accedere a Performance Insights dal riquadro di navigazione nella console. Puoi anche accedere a Performance Insights dalla scheda Monitoring (Monitoraggio) dell'istanza di scrittura per il cluster database Aurora PostgreSQL, come mostrato nell'immagine seguente.



Dalla console Performance Insights, scegli Manage metrics (Gestisci metriche). Per analizzare l'attività di connessione e disconnessione del cluster database Aurora PostgreSQL, scegli le seguenti metriche. Queste sono tutte metriche di PostgreSQL.

- `xact_commit`: il numero di transazioni confermate.
- `total_auth_attempts`: il numero di tentativi di connessione dell'utente autenticate al minuto.
- `numbackends`: il numero di backend attualmente connessi al database.

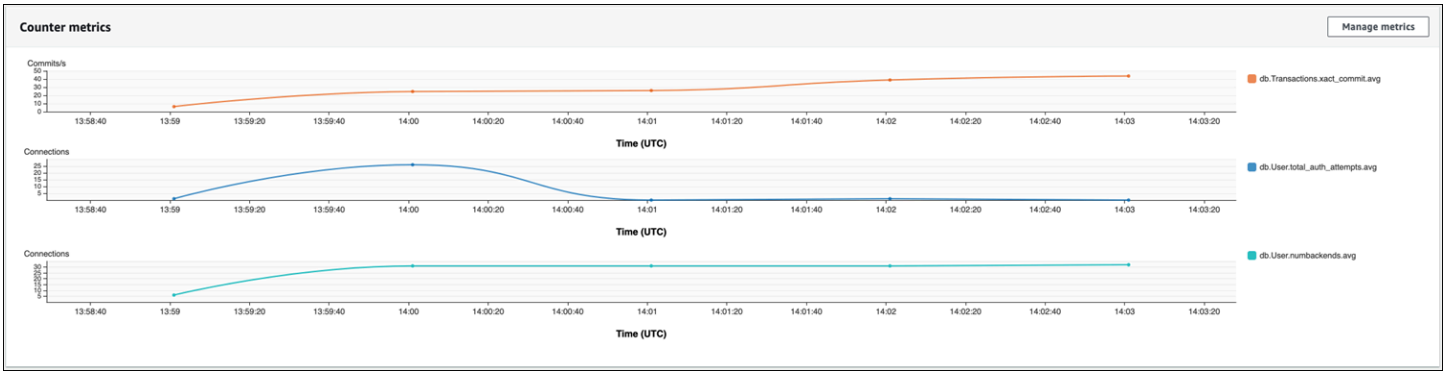


Per salvare le impostazioni e visualizzare l'attività di connessione, scegli Update graph (Aggiorna grafico).

Nell'immagine seguente è visibile l'impatto dell'esecuzione di pgbench con 100 utenti. La linea che mostra le connessioni ha una pendenza costante verso l'alto. Per ulteriori informazioni su pgbench e su come utilizzarlo, consulta pgbench nella documentazione di PostgreSQL.

L'immagine mostra che l'esecuzione di un carico di lavoro con un minimo di 100 utenti senza un pool di connessioni può causare un aumento significativo del numero di `total_auth_attempts` per tutta la durata dell'elaborazione del carico di lavoro. Tieni presente che è meglio mantenersi il più vicino `total_auth_attempts` possibile allo zero.

Con il pooling delle connessioni RDS Proxy, i tentativi di connessione aumentano all'inizio del carico di lavoro. Dopo aver impostato il pool di connessioni, la media diminuisce. Le risorse utilizzate dall'uso di transazioni e backend rimangono coerenti per tutta l'elaborazione del carico di lavoro.



Per ulteriori informazioni sull'utilizzo di Performance Insights con il cluster database Aurora PostgreSQL, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora. Per analizzare le metriche, consulta Per analizzare il parametro utilizzando il pannello di controllo di Performance Insights.

Dimostrazione dei vantaggi del pooling delle connessioni

Come accennato in precedenza, se si determina che il cluster database Aurora PostgreSQL DB presenta un problema di abbandono della connessione, è possibile utilizzare RDS Proxy per migliorare le prestazioni. Di seguito, è disponibile un esempio che mostra le differenze nell'elaborazione di un carico di lavoro con e senza il pooling delle connessioni. L'esempio utilizza pgbench per modellare il carico di lavoro transazionale.

Analogamente a psql, pgbench è un'applicazione client PostgreSQL che può essere installata ed eseguita dal computer client locale. Puoi anche installarlo ed eseguirlo dall' EC2 istanza Amazon che usi per gestire il tuo cluster Aurora PostgreSQL DB. Per ulteriori informazioni, consulta pgbench nella documentazione di PostgreSQL.

Per procedere con questo esempio, occorre creare innanzitutto l'ambiente pgbench nel database. Il seguente comando è il modello base per inizializzare le tabelle pgbench nel database specificato. Questo esempio utilizza l'account utente principale predefinito,`postgres`, per l'accesso. Modificarlo secondo necessità per il cluster database Aurora PostgreSQL. L'ambiente pgbench viene creato in un database sull'istanza di scrittura del cluster.

> **Note**
>
> Il processo di inizializzazione di pgbench elimina e ricrea le tabelle denominate `pgbench_accounts`, `pgbench_branches`, `pgbench_history` e `pgbench_tellers`. Assicurarsi che il database scelto per *dbname* durante l'inizializzazione di pgbench non utilizzi questi nomi.

```
pgbench -U postgres -h db-cluster-instance-1.111122223333.aws-region.rds.amazonaws.com
 -p 5432 -d -i -s 50 dbname
```

Per pgbench, specifica i seguenti parametri.

-d

Genera un report di debug durante l'esecuzione di pgbench.

-h

Specifica l'endpoint dell'istanza di scrittura del cluster database Aurora PostgreSQL.

-i

Inizializza l'ambiente pgbench nel database per i test di benchmark.

-p

Identifica la porta utilizzata per le connessioni al database. L'impostazione predefinita per Aurora PostgreSQL è in genere 5432 o 5433.

-s

Specifica il fattore di dimensionamento da utilizzare per compilare le tabelle con righe. Il fattore di dimensionamento predefinito è 1, che genera 1 riga nella tabella `pgbench_branches`, 10 righe nella tabella `pgbench_tellers` e 100000 righe nella tabella `pgbench_accounts`.

-U

Specifica l'account utente per l'istanza di scrittura del cluster database Aurora PostgreSQL.

Dopo aver impostato l'ambiente pgbench, puoi eseguire test di benchmarking con e senza pooling delle connessioni. Il test predefinito consiste in una serie di cinque comandi SELECT, UPDATE e INSERT per transazione che vengono eseguiti ripetutamente per il tempo specificato. Puoi specificare il fattore di dimensionamento, il numero di client e altri dettagli per modellare i tuoi casi d'uso.

Ad esempio, il comando seguente esegue il benchmark per 60 secondi (opzione -T, per il tempo) con 20 connessioni simultanee (l'opzione -c). L'opzione -C determina l'esecuzione del test utilizzando una nuova connessione ogni volta, anziché una volta per sessione client. Questa impostazione fornisce un'indicazione del sovraccarico delle connessioni.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 -C labdb
Password:**********
pgbench (14.3, server 13.3)
  starting vacuum...end.
  transaction type: <builtin: TPC-B (sort of)>
    scaling factor: 50
    query mode: simple
    number of clients: 20
    number of threads: 1
    duration: 60 s
    number of transactions actually processed: 495
    latency average = 2430.798 ms
    average connection time = 120.330 ms
    tps = 8.227750 (including reconnection times)
```

L'esecuzione di pgbench sull'istanza di scrittura di un cluster database Aurora PostgreSQL senza riutilizzare le connessioni mostra che vengono elaborate solo circa 8 transazioni al secondo. Ciò fornisce un totale di 495 transazioni durante il test di 1 minuto.

Se si riutilizzano le connessioni, la risposta del cluster database Aurora PostgreSQL per il numero di utenti è quasi 20 volte più veloce. Con il riutilizzo, viene elaborato un totale di 9.042 transazioni rispetto alle 495 nello stesso periodo di tempo e per lo stesso numero di connessioni utente. La differenza è che in quello seguente, ogni connessione viene riutilizzata.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 labdb
Password:*********
pgbench (14.3, server 13.3)
      starting vacuum...end.
      transaction type: <builtin: TPC-B (sort of)>
      scaling factor: 50
      query mode: simple
      number of clients: 20
      number of threads: 1
      duration: 60 s
      number of transactions actually processed: 9042
      latency average = 127.880 ms
      initial connection time = 2311.188 ms
      tps = 156.396765 (without initial connection time)
```

Questo esempio mostra che il pooling delle connessioni può migliorare significativamente i tempi di risposta. Per informazioni sulla configurazione di RDS Proxy per il cluster database Aurora PostgreSQL, consulta Proxy Amazon RDS per Aurora.

In Amazon Aurora PostgreSQL, puoi utilizzare diversi parametri che controllano la quantità di memoria utilizzata per varie attività di elaborazione. Se un'attività richiede più memoria rispetto alla quantità impostata per un determinato parametro, Aurora Postgre SQL utilizza altre risorse per l'elaborazione, ad esempio la scrittura su disco. Ciò può causare il rallentamento o l'arresto del cluster Aurora Postgre SQL DB, con un errore. out-of-memory

L'impostazione predefinita per ogni parametro di memoria può in genere gestire le attività di elaborazione previste. Tuttavia, puoi anche ottimizzare il cluster Aurora Postgre SQL DB RDS per i parametri relativi alla memoria dell'istanza DB. Questa ottimizzazione garantisce che venga allocata una quantità di memoria sufficiente per l'elaborazione del carico di lavoro specifico.

Di seguito sono disponibili informazioni sui parametri che controllano la gestione della memoria. Sono incluse anche informazioni su come valutare l'utilizzo della memoria.

Controllo e impostazione dei valori dei parametri

I parametri che è possibile impostare per gestire la memoria e valutare l'utilizzo della memoria del cluster Aurora Postgre SQL DB includono quanto segue:

- `work_mem`: specifica la quantità di memoria utilizzata dal cluster Aurora SQL Postgre DB per le operazioni di ordinamento interno e le tabelle hash prima di scrivere su file temporanei su disco.
- `log_temp_files`: registra la data di creazione di file temporanei, i nomi file e le dimensioni. Quando questo parametro è attivato, una voce di registro viene memorizzata per ogni file temporaneo creato. Attivala per vedere con quale frequenza il cluster Aurora Postgre SQL DB deve scrivere su disco. Disattivalo nuovamente dopo aver raccolto informazioni sulla generazione di file temporanei del cluster Aurora Postgre SQL DB, per evitare registrazioni eccessive.
- `logical_decoding_work_mem`— Speciifica la quantità di memoria (in kilobyte) da utilizzare da ogni buffer di riordino interno prima della distribuzione su disco. Questa memoria viene utilizzata per la decodifica logica, ovvero il processo di creazione di una replica. Viene eseguita convertendo i dati dal file write-ahead log (WAL) nell'output di streaming logico richiesto dalla destinazione.

  Il valore di questo parametro crea un singolo buffer della dimensione specificata per ogni connessione di replica. Per impostazione predefinita, è 65536 KB. Dopo aver riempito questo buffer, l'eccesso viene scritto su disco come un file. Per ridurre al minimo l'attività del disco, puoi impostare il valore di questo parametro su un valore molto più alto di quello di `work_mem`.

Questi sono tutti parametri dinamici, quindi puoi modificarli per la sessione corrente. Per fare ciò, connettiti al cluster Aurora Postgre SQL DB RDS per l'istanza Postgre DB utilizzando l'istruzione, come illustrato di seguito. `SET`

```
SET parameter_name TO parameter_value;
```

Le impostazioni della sessione vengono mantenute solo per la durata della sessione. Quando la sessione scade, l'impostazione del parametro viene ripristinata nel gruppo di parametri cluster database. Prima di modificare qualsiasi parametro, controlla innanzitutto i valori correnti eseguendo una query della tabella `pg_settings`, come riportato di seguito.

```
SELECT unit, setting, max_val
   FROM pg_settings WHERE name='parameter_name';
```

Ad esempio, per trovare il valore del `work_mem` parametro, connettiti all'istanza writer del cluster Aurora Postgre SQL DB ed esegui la seguente query.

```
SELECT unit, setting, max_val, pg_size_pretty(max_val::numeric)
  FROM pg_settings WHERE name='work_mem';
unit  | setting  | max_val   | pg_size_pretty
------+----------+-----------+---------------
 kB   | 1024     | 2147483647| 2048 MB
(1 row)
```

La modifica delle impostazioni dei parametri in modo che vengano mantenuti richiede l'utilizzo di un gruppo di parametri cluster DB personalizzato. Dopo aver esercitato il con valori diversi per questi parametri utilizzando `SET` l'istruzione, è possibile creare un gruppo di parametri personalizzato e applicarlo al cluster Aurora Postgre DB. SQL Per ulteriori informazioni, consulta .

Comprendere il parametro memoria di lavoro

Il parametro della memoria di lavoro (`work_mem`) specifica la quantità massima di memoria che Aurora SQL Postgre può utilizzare per elaborare query complesse. Le query complesse includono quelle che implicano operazioni di ordinamento o raggruppamento, in altre parole, le query che utilizzano le seguenti clausole:

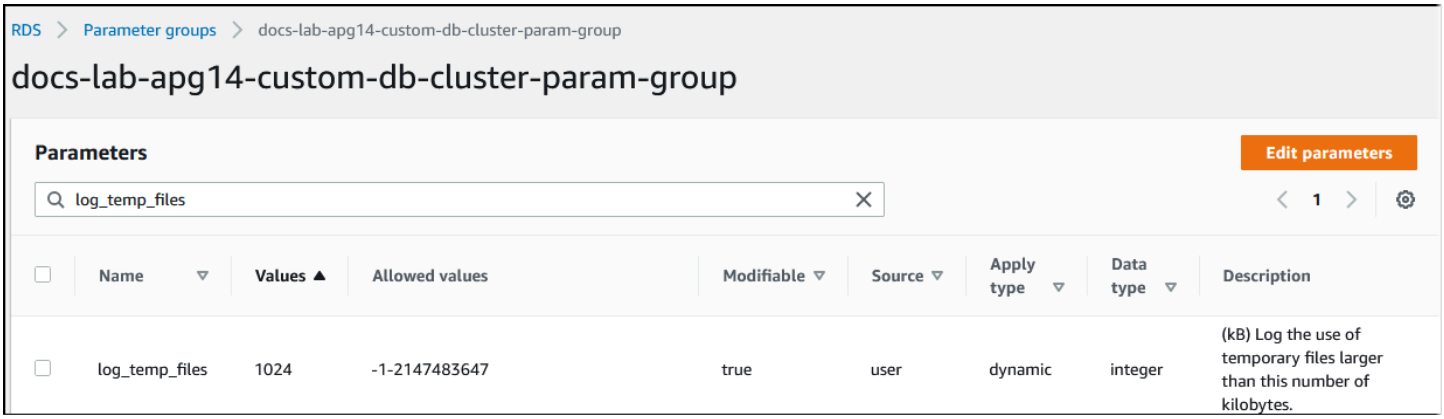
- ORDERDI
- DISTINCT
- GROUPDI
- JOIN(MERGEeHASH)

Il pianificatore di query influisce indirettamente sul modo in cui il cluster Aurora Postgre SQL DB utilizza la memoria di lavoro. Il pianificatore di query genera piani di esecuzione per l'elaborazione delle istruzioni. SQL Un piano specifico potrebbe suddividere una query complessa in più unità di lavoro che possono essere eseguite in parallelo. Quando possibile, Aurora Postgre SQL utilizza la quantità di memoria specificata nel `work_mem` parametro per ogni sessione prima di scrivere su disco per ogni processo parallelo.

Più utenti del database che eseguono più operazioni contemporaneamente e generano più unità di lavoro in parallelo possono esaurire la memoria di lavoro allocata del cluster Aurora Postgre DB. SQL Ciò può portare alla creazione eccessiva di file temporanei e all'I/O del disco o, peggio, a un errore. out-of-memory

Identificazione dell'uso di file temporanei

Ogni volta che la memoria richiesta per l'elaborazione delle query supera il valore specificato nel parametro `work_mem`, i dati di lavoro vengono scaricati su disco in un file temporaneo. Per visualizzare la frequenza di questo evento, attiva il parametro `log_temp_files`. Il parametro è disattivato (è impostato su -1) per impostazione predefinita. Per acquisire tutte le informazioni sui file temporanei, imposta questo parametro su 0. Imposta `log_temp_files` su qualsiasi altro numero intero positivo per acquisire informazioni sui file temporanei per file uguali o superiori a tale quantità di dati (in kilobyte). Nell'immagine seguente, puoi vedere un esempio di AWS Management Console.



Dopo aver configurato la registrazione dei file temporanei, puoi effettuare un test con il carico di lavoro per vedere se l'impostazione della memoria di lavoro è sufficiente. Puoi anche simulare un carico di lavoro usando pgbench, una semplice applicazione di benchmarking della community Postgre. SQL

L'esempio seguente inizializza (`-i`) `pgbench` creando le tabelle e le righe necessarie per eseguire i test. In questo esempio, il fattore di dimensionamento (`-s` 50) crea 50 righe nella tabella `pgbench_branches`, 500 righe in `pgbench_tellers` e 5.000.000 di righe nella tabella `pgbench_accounts` nel database `labdb`.

```
pgbench -U postgres -h your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -i -s 50 labdb
Password:
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
```

```
5000000 of 5000000 tuples (100%) done (elapsed 15.46 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 61.13 s (drop tables 0.08 s, create tables 0.39 s, client-side generate 54.85
 s, vacuum 2.30 s, primary keys 3.51 s)
```

Dopo aver inizializzato l'ambiente, puoi eseguire il benchmark per un periodo di tempo specifico (-T) e il numero di client (-c). Questo esempio utilizza anche l'-dopzione per generare informazioni di debug mentre le transazioni vengono elaborate dal cluster DB Aurora Postgre. SQL

```
pgbench -h -U postgres your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -d -T 60 -c 10 labdb
Password:*******
pgbench (14.3)
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 1
duration: 60 s
number of transactions actually processed: 1408
latency average = 398.467 ms
initial connection time = 4280.846 ms
tps = 25.096201 (without initial connection time)
```

Per ulteriori informazioni su pgbench, consulta pgbench nella documentazione di Postgre. SQL

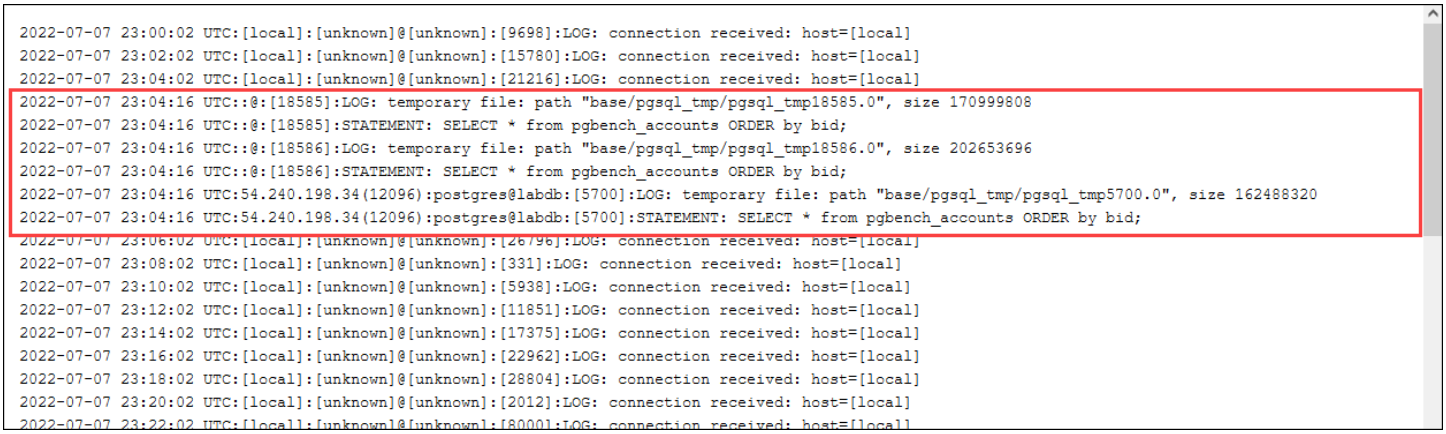
Puoi usare il metacomando psql (\d) per elencare le relazioni quali tabelle, viste e indici creati da pgbench.

```
labdb=>  \d pgbench_accounts
 Table "public.pgbench_accounts"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 aid      | integer       |           | not null |
 bid      | integer       |           |          |
 abalance | integer       |           |          |
 filler   | character(84) |           |          |
Indexes:
    "pgbench_accounts_pkey" PRIMARY KEY, btree (aid)
```

Come mostrato nell'output, la tabella `pgbench_accounts` è indicizzata sulla colonna `aid`. Per garantire che questa query successiva utilizzi la memoria di lavoro, esegui una query su qualsiasi colonna non indicizzata, come quella mostrata nell'esempio seguente.

```
postgres=> SELECT * FROM pgbench_accounts ORDER BY bid;
```

Controlla se il registro contiene file temporanei. A tale scopo, apri AWS Management Console, scegli l'istanza del cluster Aurora Postgre SQL DB, quindi scegli la scheda Logs & Events. Visualizza i registri nella console o scaricali per ulteriori analisi. Come mostrato nell'immagine seguente, la dimensione dei file temporanei necessari per elaborare la query indica che è opportuno valutare se aumentare la quantità specificata per il parametro `work_mem`.



Puoi configurare questo parametro in modo diverso per singoli e gruppi, in base alle esigenze operative. Ad esempio, puoi impostare il parametro `work_mem` su 8 GB per il ruolo denominato `dev_team`.

```
postgres=> ALTER ROLE dev_team SET work_mem='8GB';
```

Con questa impostazione per `work_mem`, a qualsiasi ruolo che è un membro del ruolo `dev_team` sono assegnati fino a 8 GB di memoria di lavoro.

Utilizzo degli indici per tempo di risposta più rapido

Se le query richiedono troppo tempo per restituire i risultati, puoi verificare che l'uso degli indici sia quello previsto. Innanzitutto, attiva `\timing`, il metacomando psql, come segue.

```
postgres=>  \timing on
```

Dopo aver attivato la sincronizzazione, usa una semplice SELECT dichiarazione.

```
postgres=> SELECT COUNT(*) FROM
  (SELECT * FROM pgbench_accounts
  ORDER BY bid)
  AS accounts;
count
-------
5000000
(1 row)
Time: 3119.049 ms (00:03.119)
```

Come mostrato nell'output, questa query ha richiesto poco più di 3 secondi per il completamento. Per migliorare il tempo di risposta, crea un indice su `pgbench_accounts`, come riportato di seguito.

```
postgres=> CREATE INDEX ON pgbench_accounts(bid);
CREATE INDEX
```

Esegui nuovamente la query e osserva che tempo di risposta è inferiore. In questo esempio, la query è stata completata all'incirca 5 volte più velocemente, in circa mezzo secondo.

```
postgres=>  SELECT COUNT(*) FROM (SELECT * FROM pgbench_accounts ORDER BY bid) AS
 accounts;
 count
-------
 5000000
(1 row)
Time: 567.095 ms
```

Regolazione della memoria di lavoro per la decodifica logica

La replica logica è disponibile in tutte le versioni di Aurora SQL RDS sin dalla sua introduzione nella versione 10 di SQL Postgre. SQL Quando si configura la replica logica, è anche possibile impostare il parametro `logical_decoding_work_mem` per specificare la quantità di memoria utilizzabile dal processo di decodifica logica per il processo di decodifica e streaming.

Durante la decodifica logica, i record write-ahead log (WAL) vengono convertiti in SQL istruzioni che vengono poi inviate a un'altra destinazione per la replica logica o un'altra attività. Quando una transazione viene scritta WAL e poi convertita, l'intera transazione deve rientrare nel valore specificato per. `logical_decoding_work_mem` Questo parametro è impostato su 65536 KB per impostazione predefinita. L'eventuale overflow viene scritto su disco. Ciò significa che deve

essere riletto dal disco prima di poter essere inviato alla destinazione, rallentando così il processo complessivo.

È possibile valutare l'entità dell'overflow di transazione nel carico di lavoro corrente in un momento specifico utilizzando la funzione `aurora_stat_file` come mostrato nell'esempio seguente.

```
SELECT split_part (filename, '/', 2)
   AS slot_name, count(1) AS num_spill_files,
   sum(used_bytes) AS slot_total_bytes,
   pg_size_pretty(sum(used_bytes)) AS slot_total_size
   FROM aurora_stat_file()
   WHERE filename like '%spill%'
   GROUP BY 1;
  slot_name | num_spill_files | slot_total_bytes | slot_total_size
------------+-----------------+------------------+-----------------
 slot_name  |       590       |      411600000   | 393 MB
(1 row)
```

Questa query restituisce il conteggio e la dimensione dei file di fuoriuscita sul cluster Aurora SQL Postgre DB quando viene richiamata la query. Per i carichi di lavoro a esecuzione più lunga, è possibile che sul disco non siano ancora presenti file spill. Per profilare i carichi di lavoro con esecuzione a lungo termine, è opportuno creare una tabella per acquisire le informazioni sui file spill durante l'esecuzione del carico di lavoro. È possibile creare la tabella come indicato di seguito.

```
CREATE TABLE spill_file_tracking AS
    SELECT now() AS spill_time,*
    FROM aurora_stat_file()
    WHERE filename LIKE '%spill%';
```

Per vedere come vengono utilizzati i file spill durante la replica logica, configura un autore e un sottoscrittore e quindi avvia una semplice replica. Per ulteriori informazioni, consulta Configurazione della replica logica per il cluster Aurora SQL Postgre DB. Con la replica in corso, puoi creare un processo che acquisisce il set di risultati dalla funzione file spill `aurora_stat_file()`, come riportato di seguito.

```
INSERT INTO spill_file_tracking
  SELECT now(),*
  FROM aurora_stat_file()
  WHERE filename LIKE '%spill%';
```

Usa il seguente comando psql per eseguire il processo una volta al secondo.

```
\watch 0.5
```

Mentre il processo è in esecuzione, esegui la connessione all'istanza di scrittura da un'altra sessione psql. Utilizzate la seguente serie di istruzioni per eseguire un carico di lavoro che supera la configurazione della memoria e fa sì che Aurora Postgre SQL crei uno spill file.

```
labdb=> CREATE TABLE my_table (a int PRIMARY KEY, b int);
CREATE TABLE
labdb=> INSERT INTO my_table SELECT x,x FROM generate_series(0,10000000) x;
INSERT 0 10000001
labdb=> UPDATE my_table SET b=b+1;
UPDATE 10000001
```

Queste istruzioni richiedono diversi minuti per il completamento. Al termine, premi contemporaneamente i tasti Ctrl e C per interrompere la funzione di monitoraggio. Quindi usa il seguente comando per creare una tabella che contenga le informazioni sull'utilizzo dello spill file del cluster Aurora Postgre SQL DB.

```
SELECT spill_time, split_part (filename, '/', 2)
    AS slot_name, count(1)
    AS spills, sum(used_bytes)
    AS slot_total_bytes, pg_size_pretty(sum(used_bytes))
    AS slot_total_size FROM spill_file_tracking
  GROUP BY 1,2 ORDER BY 1;
                  spill_time | slot_name             | spills | slot_total_bytes |
 slot_total_size
------------------------------+-----------------------+--------+-----------------
+-----------------
2022-04-15 13:42:52.528272+00 | replication_slot_name | 1      | 142352280        | 136
 MB
2022-04-15 14:11:33.962216+00 | replication_slot_name | 4      | 467637996        | 446
 MB
2022-04-15 14:12:00.997636+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:03.030245+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:05.059761+00 | replication_slot_name | 5      | 618410996        | 590
 MB
2022-04-15 14:12:07.22905+00  | replication_slot_name | 5      | 640585316        | 611
 MB
```

```
(6 rows)
```

L'output mostra che l'esecuzione dell'esempio ha creato cinque file spill che utilizzano 611 MB di memoria. Per evitare di scrivere su disco, è consigliabile impostare il parametro `logical_decoding_work_mem` per la dimensione di memoria successiva più elevata, 1024.

Utilizzo dei CloudWatch parametri di Amazon per analizzare l'utilizzo delle risorse per Aurora PostgreSQL

Aurora invia automaticamente i dati metrici a CloudWatch in periodi di 1 minuto. È possibile analizzare l'utilizzo delle risorse per Aurora PostgreSQL utilizzando le metriche. CloudWatch valutare la velocità di trasmissione effettiva e l'utilizzo della rete.

Valutazione del throughput di rete con CloudWatch

Quando l'utilizzo del sistema è prossimo ai limiti delle risorse per il tipo di istanza, l'elaborazione può subire un rallentamento. È possibile utilizzare CloudWatch Logs Insights per monitorare l'utilizzo delle risorse di archiviazione e garantire che siano disponibili risorse sufficienti. Se necessario, puoi modificare l'istanza database in una classe di istanza più grande.

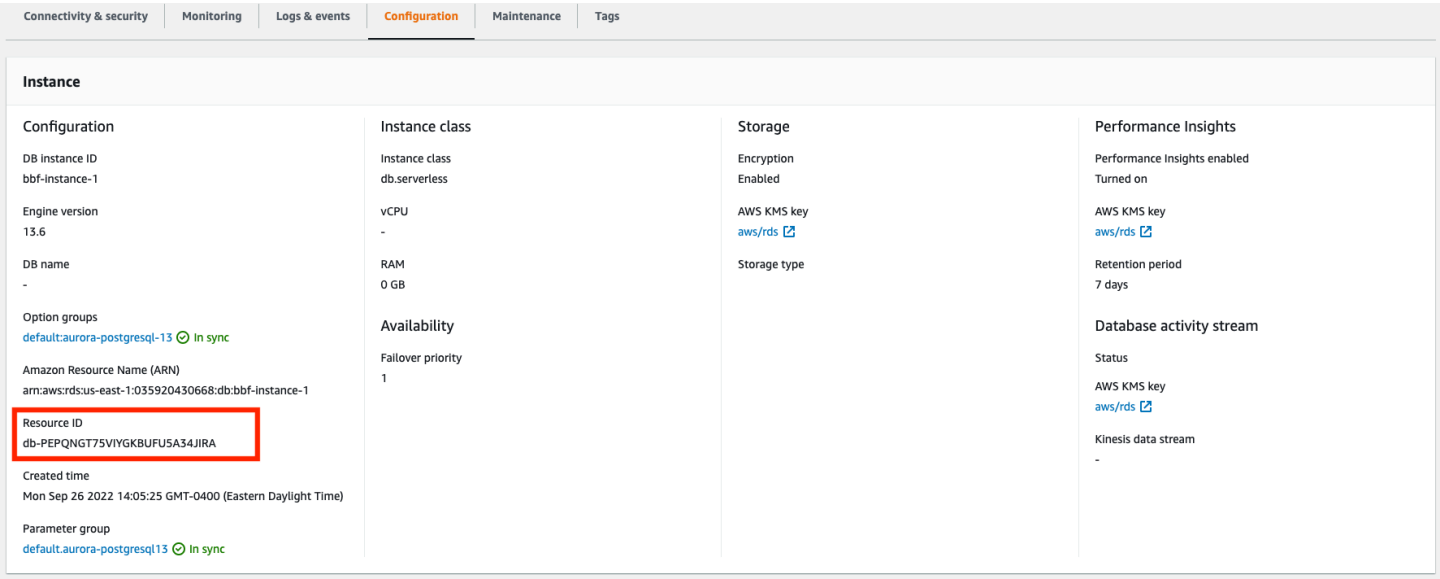
L'elaborazione dell'archiviazione di Aurora può subire un rallentamento perché:

- La larghezza di banda della rete è insufficiente tra il client e l'istanza database.
- La larghezza di banda della rete è insufficiente per il sottosistema di archiviazione.
- Un carico di lavoro è troppo elevato per il tipo di istanza in uso.

È possibile interrogare CloudWatch Logs Insights per generare un grafico dell'utilizzo delle risorse di archiviazione Aurora per monitorare le risorse. Il grafico mostra l'utilizzo della CPU e i parametri per permetterti di decidere se aumentare le dimensioni dell'istanza. Per informazioni sulla sintassi delle query per CloudWatch Logs Insights, vedere CloudWatch Sintassi delle query di Logs Insights

Per utilizzarli CloudWatch, è necessario esportare i file di registro di Aurora PostgreSQL in. CloudWatch È inoltre possibile modificare il cluster esistente in cui esportare i log. CloudWatch Per informazioni sull'esportazione dei log in, vedere CloudWatch. Attivazione dell'opzione per pubblicare i log su Amazon CloudWatch

È necessario il Resource ID dell'istanza DB per interrogare CloudWatch Logs Insights. Il valore Resource ID (ID risorsa) è disponibile nella scheda Configuration (Configurazione) della console:



Per eseguire le query sui file di log per i parametri di archiviazione delle risorse:

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.

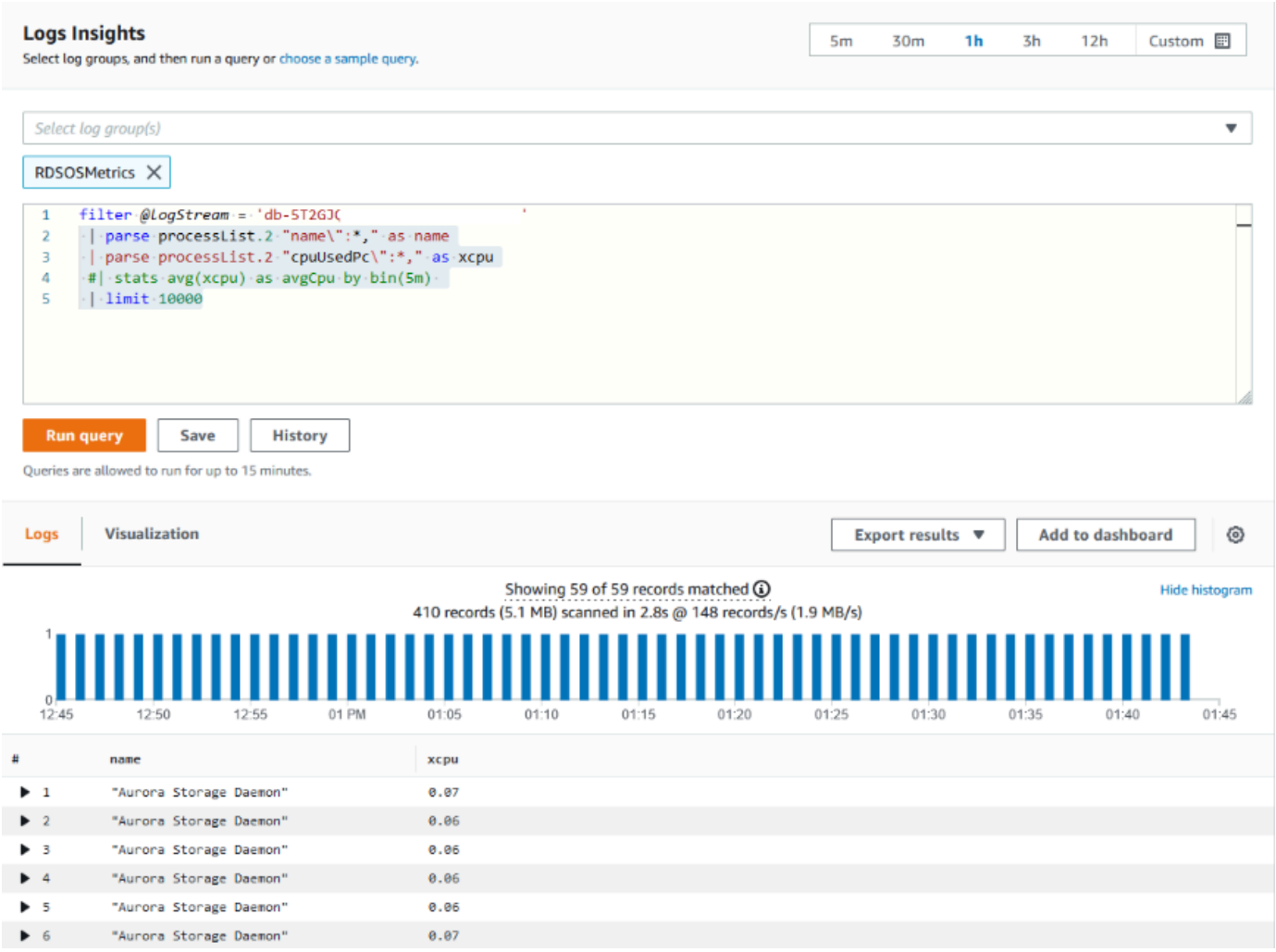
   Viene visualizzata la home page CloudWatch panoramica.
2. Se necessario, modifica Regione AWS. Nella barra di navigazione, scegli Regione AWS dove si trovano AWS le tue risorse. Per ulteriori informazioni, consulta Regioni ed endpoint.
3. Nel pannello di navigazione scegli Logs (Log), quindi Logs Insights (Approfondimenti sui file di log).

   Viene visualizzata la pagina Logs Insights (Approfondimenti sui file di log).
4. Seleziona nell'elenco a discesa i file di log da analizzare.
5. Immetti la seguente query nel campo, sostituendo `<resource ID>` con l'ID risorsa del tuo cluster database:

   ```
   filter @logStream = <resource ID> | parse @message "\"Aurora Storage Daemon\"*memoryUsedPc\":*,\"cpuUsedPc\":*," as a,memoryUsedPc,cpuUsedPc | display memoryUsedPc,cpuUsedPc #| stats avg(xcpu) as avgCpu by bin(5m) | limit 10000
   ```
6. Fai clic su Run query (Esegui query).

   Viene visualizzato il grafico dell'utilizzo dell'archiviazione.

L'immagine seguente mostra la pagina Logs Insights (Approfondimenti sui file di log) e la visualizzazione del grafico.



Valutazione dell'utilizzo delle istanze DB per Aurora CloudWatch PostgreSQL con metriche

Puoi utilizzare le CloudWatch metriche per monitorare il throughput dell'istanza e scoprire se la classe di istanza fornisce risorse sufficienti per le tue applicazioni. Per informazioni sui limiti della classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora e individua le specifiche della classe di istanza database in uso per esaminare le prestazioni della rete.

Se l'utilizzo dell'istanza database è prossimo al limite della classe di istanza, le prestazioni potrebbero iniziare a rallentare. Le CloudWatch metriche possono confermare questa situazione in modo da poter pianificare la scalabilità manuale a una classe di istanze più ampia.

Combina i seguenti valori CloudWatch delle metriche per scoprire se ti stai avvicinando al limite della classe di istanza:

- NetworkThroughput— La quantità di throughput di rete ricevuta e trasmessa dai client per ogni istanza nel cluster Aurora DB. La velocità di trasmissione effettiva non include il traffico di rete tra le istanze nel cluster database e il volume del cluster.
- StorageNetworkThroughput— La quantità di throughput di rete ricevuta e inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB.

Aggiungi NetworkThroughputa per trovare il StorageNetworkThroughputthroughput di rete ricevuto e inviato al sottosistema di archiviazione Aurora da ciascuna istanza del cluster Aurora DB. Il limite della classe dell'istanza deve essere maggiore della somma di questi due parametri combinati.

È possibile utilizzare i seguenti parametri per esaminare ulteriori dettagli del traffico di rete proveniente dalle applicazioni client durante l'invio e la ricezione:

- NetworkReceiveThroughput— La quantità di throughput di rete ricevuta dai client da ciascuna istanza nel cluster Aurora PostgreSQL DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- NetworkTransmitThroughput— La quantità di throughput di rete inviata ai client da ciascuna istanza nel cluster Aurora DB. Questo throughput non include il traffico di rete tra le istanze nel cluster di database e il volume del cluster.
- StorageNetworkReceiveThroughput— La quantità di throughput di rete ricevuta dal sottosistema di archiviazione Aurora da ciascuna istanza del cluster DB.
- StorageNetworkTransmitThroughput— La quantità di throughput di rete inviata al sottosistema di archiviazione Aurora da ciascuna istanza del cluster DB.

Somma tutti questi parametri per confrontare l'utilizzo della rete e il limite della classe di istanza. Il limite della classe di istanza deve essere maggiore della somma di questi parametri combinati.

I limiti della rete e l'utilizzo della CPU per l'archiviazione sono reciproci, pertanto quando la velocità di trasmissione effettiva della rete aumenta, cresce anche l'utilizzo della CPU. Il monitoraggio

dell'utilizzo della CPU e della rete fornisce informazioni su come e perché le risorse vengono esaurite.

Per ridurre al minimo l'utilizzo della rete, puoi prendere in considerazione:

- L'uso di una classe di istanza più grande.
- L'uso di strategie di partizionamento `pg_partman`.
- La suddivisione delle richieste di scrittura in batch per ridurre le transazioni in generale.
- L'instradamento del carico di lavoro di sola lettura a un'istanza di sola lettura.
- L'eliminazione degli indici non utilizzati.
- La verifica della presenza di oggetti di grandi dimensioni e VACUUM. In caso di dimensioni estremamente elevate, utilizza l'estensione PostgreSQL `pg_repack`. Per ulteriori informazioni su `pg_repack`, consulta Reorganize tables in PostgreSQL databases with minimal locks (Riorganizzazione delle tabelle nei database PostgreSQL con blocchi minimi).

Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL

L'utilizzo della replica logica e della clonazione rapida Aurora consente di eseguire un aggiornamento della versione principale che utilizza la versione corrente del database Aurora PostgreSQL mentre si esegue gradualmente la migrazione dei dati modificati nel nuovo database della versione principale. Questo processo di aggiornamento con tempi di inattività ridotti viene definito aggiornamento blu/verde. La versione corrente del database è denominata ambiente "blu" e la nuova versione del database è denominata ambiente "verde".

La clonazione rapida di Aurora carica completamente i dati esistenti generando uno snapshot del database di origine. La clonazione rapida utilizza un copy-on-write protocollo basato sul livello di archiviazione Aurora, che consente di creare un clone del database in breve tempo. Questo metodo è molto efficace quando si esegue l'aggiornamento a un database di grandi dimensioni.

La replica logica in PostgreSQL monitora e trasferisce le modifiche ai dati dall'istanza iniziale a una nuova istanza in esecuzione in parallelo fino al passaggio alla versione più recente di PostgreSQL. La replica logica utilizza un modello di pubblicazione e sottoscrizione. Per ulteriori informazioni sulla replica logica di Aurora PostgreSQL, consulta Replica con Amazon Aurora PostgreSQL.

> **Tip**
>
> Puoi ridurre al minimo i tempi di inattività necessari per l'aggiornamento di una versione principale utilizzando la funzionalità gestita di Amazon RDS Blue/Green Deployment. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Argomenti

- Requisiti
- Limitazioni
- Impostazione e controllo dei valori dei parametri
- Aggiornamento di Aurora PostgreSQL a una nuova versione principale
- Esecuzione di attività successive all'aggiornamento

Requisiti

È necessario soddisfare i seguenti requisiti per eseguire questo processo di aggiornamento con tempi di inattività ridotti:

- È necessario disporre delle autorizzazioni rds_superuser.
- Il cluster database Aurora PostgreSQL da aggiornare deve eseguire una versione supportata in grado di eseguire aggiornamenti di versione principali utilizzando la replica logica. Assicurarsi di applicare eventuali aggiornamenti e patch della versione secondaria al cluster database in uso. La funzione `aurora_volume_logical_start_lsn` utilizzata in questa tecnica è supportata nelle seguenti versioni di Aurora PostgreSQL:
  - 15.2 o versioni successive alla 15
  - 14.3 o versioni successive alla 14
  - 13.6 o versioni successive alla 13
  - 12.10 e versioni successive alla 12
  - 11.15 e versioni successive alla 11
  - 10.20 e versioni successive alla 10

  Per ulteriori informazioni sulla funzione `aurora_volume_logical_start_lsn` consulta aurora_volume_logical_start_lsn.

- Tutte le tabelle devono avere una chiave primaria o includere una colonna di identità PostgreSQL.
- Configura il gruppo di sicurezza per il VPC in modo da consentire l'accesso in entrata e in uscita tra i due cluster di database Aurora PostgreSQL, vecchio e nuovo. Puoi autorizzare l'accesso a un intervallo di instradamento interdominio senza classi (CIDR) specifico o a un altro gruppo di sicurezza nel tuo VPC o in un VPC in peering. Il VPC in peering richiede una connessione peering VPC.

> **Note**
>
> Per informazioni dettagliate sulle autorizzazioni necessarie per configurare e gestire uno scenario di replica logica in esecuzione, consulta la documentazione principale di PostgreSQL.

Limitazioni

Quando si esegue l'aggiornamento con tempi di inattività ridotti del cluster database Aurora PostgreSQL a una nuova versione principale, si utilizza la funzionalità di replica logica nativa di PostgreSQL. Ha le stesse capacità e limitazioni della replica logica di PostgreSQL. Per ulteriori informazioni, consulta la pagina relativa alla replica logica di PostgreSQL.

- I comandi DDL (Data Definition Language) non vengono replicati.
- La replica non supporta le modifiche dello schema in un database attivo. Lo schema viene ricreato nel suo formato originale durante il processo di clonazione. Se si modifica lo schema dopo la clonazione, ma prima di completare l'aggiornamento, la modifica non si riflette nell'istanza aggiornata.
- Gli oggetti di grandi dimensioni non vengono replicati, ma è possibile archiviare i dati in tabelle normali.
- La replica è supportata solo dalle tabelle, comprese le tabelle partizionate. La replica su altri tipi di relazioni, come viste, viste materializzate o tabelle esterne, non è supportata.
- I dati della sequenza non vengono replicati e richiedono l'aggiornamento manuale dopo il failover.

> **Note**
>
> Questo aggiornamento non supporta l'elaborazione automatica di script. È necessario eseguire tutti i passaggi manualmente.

Impostazione e controllo dei valori dei parametri

Prima dell'aggiornamento, configura l'istanza di scrittura del cluster di database Aurora PostgreSQL in modo che agisca da server di pubblicazione. L'istanza deve utilizzare un gruppo di parametri del·cluster di database personalizzato con le seguenti impostazioni:

- `rds.logical_replication`: imposta questo parametro su 1. Il parametro `rds.logical_replication` ha lo stesso scopo del parametro `wal_level` di un server PostgreSQL autonomo e di altri parametri che controllano la gestione dei file WAL (write-ahead log).
- `max_replication_slots`: imposta questo parametro sul numero totale di sottoscrizioni che intendi creare. Se la utilizzi AWS DMS, imposta questo parametro sul numero di AWS DMS attività che intendi utilizzare per l'acquisizione di dati modificati da questo cluster DB.
- `max_wal_senders`: imposta il numero di connessioni simultanee, più qualcuna extra, da rendere disponibili per le attività di gestione e le nuove sessioni. Se lo stai utilizzando AWS DMS, il numero di max_wal_senders deve essere uguale al numero di sessioni simultanee più il numero di AWS DMS attività che potrebbero funzionare in un dato momento.
- `max_logical_replication_workers`: imposta il numero di worker di replica logica e di sincronizzazione delle tabelle previsti. In genere è sicuro impostare il numero di worker di replica sullo stesso valore utilizzato per max_wal_senders. I worker vengono presi dal pool dei processi in background (max_worker_processes) allocati per il server.
- `max_worker_processes`: imposta il numero di processi in background per il server. Questo numero deve essere sufficientemente grande da assegnare i worker alla replica, ai processi di pulizia automatica e ad altri processi di manutenzione che possono avvenire contemporaneamente.

Quando si esegue l'aggiornamento a una versione più recente di Aurora PostgreSQL, è necessario duplicare tutti i parametri modificati nella versione precedente del gruppo di parametri. Questi parametri vengono applicati alla versione aggiornata. Puoi eseguire query sulla tabella

`pg_settings` per ottenere l'elenco delle impostazioni dei parametri in modo da poterle ricreare nel tuo nuovo cluster di database Aurora PostgreSQL.

Ad esempio, per ottenere le impostazioni per i parametri di replica, esegui la query riportata di seguito:

```
SELECT name, setting FROM pg_settings WHERE name in
('rds.logical_replication', 'max_replication_slots',
'max_wal_senders', 'max_logical_replication_workers',
'max_worker_processes');
```

Aggiornamento di Aurora PostgreSQL a una nuova versione principale

Per preparare l'entità di pubblicazione (blu)

1. Nell'esempio che segue, l'istanza di scrittura di origine (blu) è un cluster di database Aurora PostgreSQL che esegue PostgreSQL versione 11.15. È il nodo di pubblicazione nel nostro scenario di replica. Per questa dimostrazione, l'istanza di scrittura di origine ospita una tabella di esempio che contiene una serie di valori:

   ```
   CREATE TABLE my_table (a int PRIMARY KEY);
   INSERT INTO my_table VALUES (generate_series(1,100));
   ```

2. Per creare una pubblicazione nell'istanza di origine, connettiti al nodo di scrittura dell'istanza con psql (l'interfaccia della linea di comando per PostgreSQL) o con il client di tua scelta. Immetti il comando seguente in ogni database:

   ```
   CREATE PUBLICATION publication_name FOR ALL TABLES;
   ```

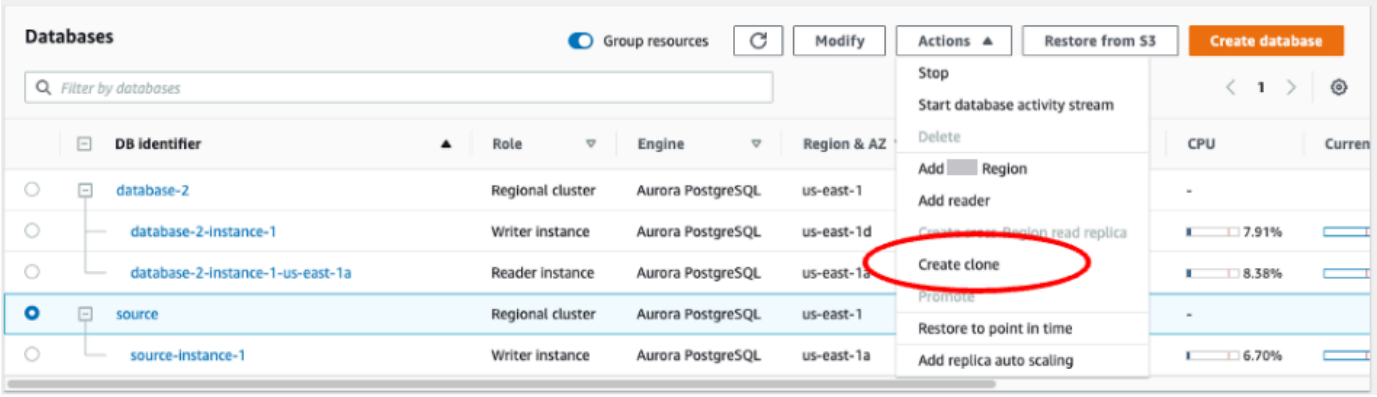
   L'indicazione publication_name specifica il nome della pubblicazione.

3. È inoltre necessario creare uno slot di replica nell'istanza. Il comando seguente crea uno slot di replica e carica il plug-in di decodifica logica `pgoutput`. Il plug-in modifica il contenuto letto da WAL (write-ahead log) al protocollo di replica logica e filtra i dati in base alle specifiche di pubblicazione.

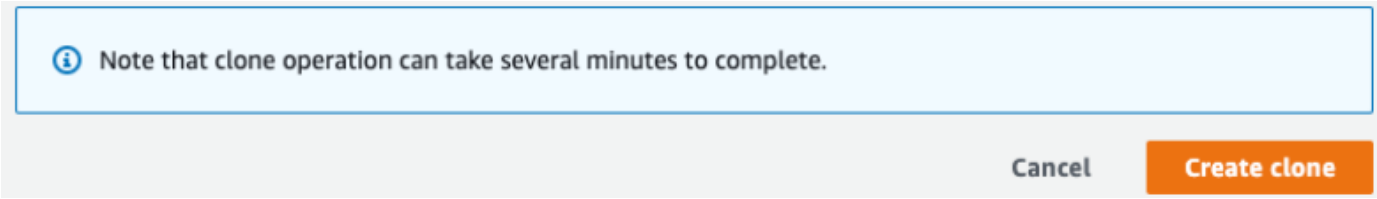
   ```
   SELECT pg_create_logical_replication_slot('replication_slot_name', 'pgoutput');
   ```

Per clonare l'entità di pubblicazione

1. Utilizza la console Amazon RDS per creare un clone dell'istanza di origine. Evidenzia il nome dell'istanza nella console Amazon RDS, quindi scegli Create clone (Crea clone) nel menu Actions (Operazioni).



2. Specifica un nome univoco per l'istanza. La maggior parte delle impostazioni sono quelle predefinite dell'istanza di origine. Dopo aver apportato le modifiche necessarie per la nuova istanza, scegli Create clone (Crea clone).



3. Durante l'avvio dell'istanza di destinazione, la colonna Status (Stato) del nodo di scrittura visualizza Creating (Creazione) nella colonna Status (Stato). Quando l'istanza è pronta, lo stato diventa Available (Disponibile).

Per preparare il clone per un aggiornamento

1. Il clone è l'istanza "verde" nel modello di implementazione. È l'host del nodo di sottoscrizione della replica. Quando il nodo diventa disponibile, connettiti con psql ed esegui una query sul nuovo nodo di scrittura per ottenere il numero di sequenza del log che identifica l'inizio di un record nel flusso WAL.

```
SELECT aurora_volume_logical_start_lsn();
```

2. Nella risposta alla query, trovi il numero di sequenza del log. Questo numero ti servirà più avanti nel processo, quindi prendine nota.

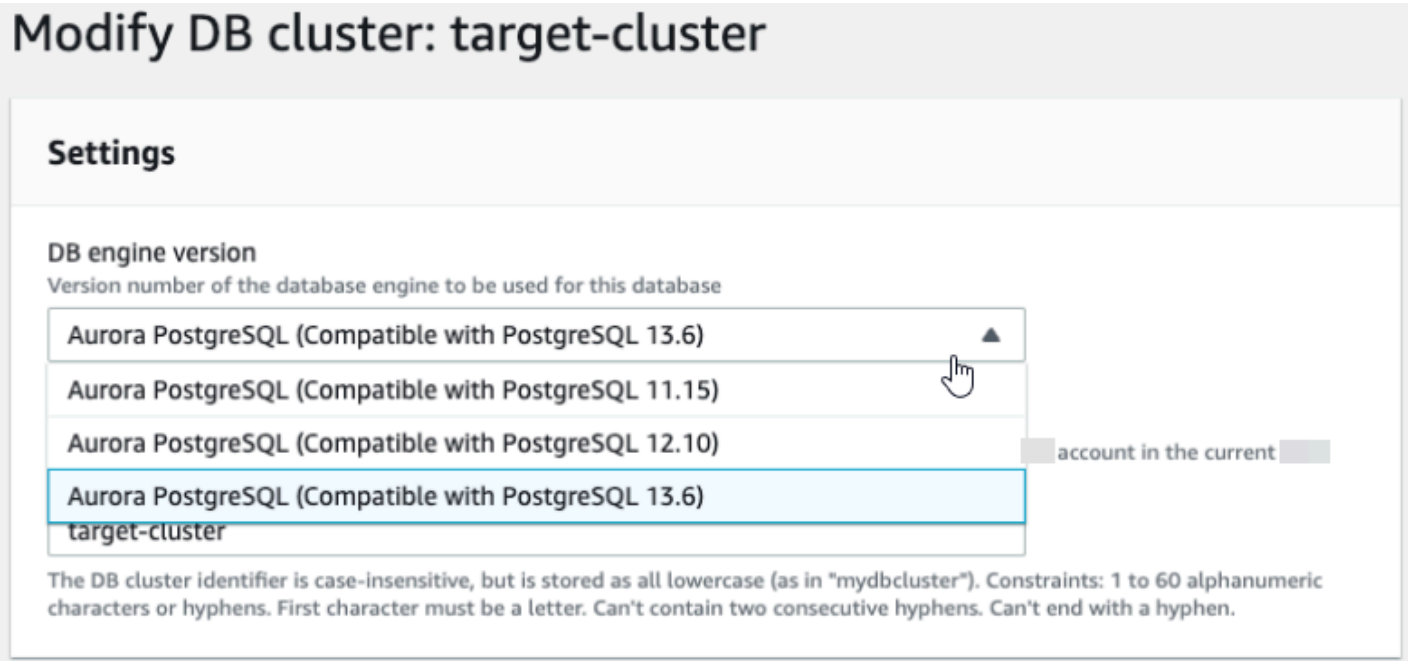
```
postgres=> SELECT aurora_volume_logical_start_lsn();
aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

3. Prima di aggiornare il clone, elimina lo slot di replica del clone.

```
SELECT pg_drop_replication_slot('replication_slot_name');
```

Per aggiornare il cluster a una nuova versione principale

- Dopo aver clonato il nodo provider, utilizza la console Amazon RDS per avviare l'aggiornamento della versione principale sul nodo di sottoscrizione. Evidenzia il nome dell'istanza nella console RDS e seleziona il pulsante Modify (Modifica). Seleziona la versione aggiornata e i gruppi di parametri aggiornati e applica immediatamente le impostazioni per aggiornare l'istanza di destinazione.



- È anche possibile utilizzare l'interfaccia della linea di comando per eseguire un aggiornamento:

```
aws rds modify-db-cluster –db-cluster-identifier $TARGET_Aurora_ID –engine-version
 13.6 –allow-major-version-upgrade –apply-immediately
```

Per preparare il sottoscrittore (verde)

1. Quando il clone diventa disponibile, connettersi a psql e definire la sottoscrizione. Per farlo, è necessario specificare le seguenti opzioni nel comando `CREATE SUBSCRIPTION`:

   - `subscription_name`: il nome della sottoscrizione.
   - `admin_user_name`: il nome di un utente amministrativo con autorizzazioni rds_superuser.
   - `admin_user_password`: la password associata all'utente amministrativo.
   - `source_instance_URL`: l'URL dell'istanza del server di pubblicazione.
   - `database`: il database a cui si connette il server di sottoscrizione.
   - `publication_name`: il nome del server di pubblicazione.
   - `replication_slot_name`: il nome dello slot di replica.

   ```
   CREATE SUBSCRIPTION subscription_name
   CONNECTION 'postgres://admin_user_name:admin_user_password@source_instance_URL/
   database' PUBLICATION publication_name
   WITH (copy_data = false, create_slot = false, enabled = false, connect = true,
    slot_name = 'replication_slot_name');
   ```

2. Dopo aver creato la sottoscrizione, esegui la query sulla vista pg_replication_origin per recuperare il valore roname, che è l'identificatore dell'origine della replica. Ogni istanza ha un solo `roname`:

   ```
   SELECT * FROM pg_replication_origin;
   ```

   Per esempio:

   ```
   postgres=>
   SELECT * FROM pg_replication_origin;

   roident | roname
   ---------+----------
   1 | pg_24586
   ```

3. Specifica il numero di sequenza del log che hai salvato dalla precedente query del nodo di pubblicazione e il `roname` restituito dall'[ISTANZA] del nodo di sottoscrizione nel comando. Questo comando utilizza la funzione `pg_replication_origin_advance` per specificare il punto iniziale nella sequenza del log per la replica.

```
SELECT pg_replication_origin_advance('roname', 'log_sequence_number');
```

`roname` è l'identificatore restituito dalla vista pg_replication_origin.

`log_sequence_number` è il valore restituito dalla precedente query della funzione `aurora_volume_logical_start_lsn`.

4. Quindi, utilizza la clausola `ALTER SUBSCRIPTION... ENABLE` per attivare la replica logica.

```
ALTER SUBSCRIPTION subscription_name ENABLE;
```

5. A questo punto, puoi verificare se la replica funziona. Aggiungi un valore all'istanza di pubblicazione, quindi verifica che il valore sia stato replicato nel nodo di sottoscrizione.

   Quindi, utilizza il comando seguente per monitorare il ritardo di replica sul nodo di pubblicazione:

```
SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';
```

   Per esempio:

```
postgres=> SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
 confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';

current_time                   | slot_name             | active | active_pid |
 diff_size | diff_bytes
-------------------------------+-----------------------+--------+------------
+-----------+------------
2022-04-13 15:11:00.243401+00  | replication_slot_name | t      | 21854      | 136
 bytes | 136
(1 row)
```

   È possibile monitorare il ritardo di replica utilizzando i valori `diff_size` e `diff_bytes`. Quando questi valori sono pari a 0, la replica ha raggiunto l'istanza database di origine.

Esecuzione di attività successive all'aggiornamento

Al termine dell'aggiornamento, lo stato dell'istanza viene visualizzato come Available (Disponibile) nella colonna Status (Stato) del dashboard della console. Nella nuova istanza, ti consigliamo di effettuare le seguenti operazioni:

- Reindirizza le tue applicazioni in modo che puntino al nodo di scrittura.
- Aggiungi i nodi di lettura per gestire il contenitore e garantire un'elevata disponibilità in caso di problemi con il nodo di scrittura.
- I cluster di database Aurora PostgreSQL richiedono occasionalmente gli aggiornamenti del sistema operativo. Questi aggiornamenti a volte includono una nuova versione della libreria glibc. Durante gli aggiornamenti, ti consigliamo di seguire le linee guida descritte in .
- Aggiorna le autorizzazioni utente sulla nuova istanza per garantire l'accesso.

Dopo aver testato l'applicazione e i dati sulla nuova istanza, ti consigliamo di eseguire un backup finale dell'istanza iniziale prima di rimuoverla. Per ulteriori informazioni sull'uso della replica logica in un host Aurora, consulta Configurazione della replica logica per il cluster Aurora SQL Postgre DB.

Risoluzione dei problemi di storage in Aurora PostgreSQL

Se la quantità di memoria di lavoro necessaria per le operazioni di ordinamento o creazione dell'indice supera la quantità allocata dal parametro `work_mem`, Aurora PostgreSQL scrive i dati in eccesso in file temporanei su disco. Quando scrive i dati, Aurora PostgreSQL usa lo stesso spazio utilizzato per l'archiviazione dei log di errori e messaggi, ossia lo spazio di archiviazione locale. Ogni istanza del cluster database Aurora PostgreSQL ha una quantità di spazio di archiviazione locale disponibile che si basa sulla relativa classe di istanza database. Per aumentare la quantità di spazio di archiviazione locale, è necessario modificare l'istanza per utilizzare una classe di istanza database più grande. Per le specifiche per la classe di istanza database, consulta Specifiche hardware per le classi di istanza database per Aurora.

Puoi monitorare lo spazio di archiviazione locale del cluster Aurora PostgreSQL DB osservando la metrica Amazon per. CloudWatch `FreeLocalStorage` Questa metrica segnala la quantità di spazio di archiviazione disponibile per ogni istanza database nel cluster database Aurora per le tabelle e i log temporanei. Per ulteriori informazioni, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch.

Le operazioni di ordinamento, indicizzazione e raggruppamento iniziano nella memoria di lavoro, ma spesso devono essere sottoposte a offload nello spazio di archiviazione locale. Se il cluster Aurora

PostgreSQL DB esaurisce la memoria locale a causa di questi tipi di operazioni, è possibile risolvere il problema eseguendo una delle seguenti azioni.

- Aumenta la quantità di memoria di lavoro. In tal modo si riduce la necessità di utilizzare lo spazio di archiviazione locale. Per impostazione predefinita, PostgreSQL alloca 4 MB per ogni operazione di ordinamento, raggruppamento e indicizzazione. Per verificare il valore corrente della memoria di lavoro per l'istanza di scrittura del cluster database Aurora PostgreSQL, esegui la connessione all'istanza utilizzando `psql` ed esegui il comando indicato di seguito.

  ```
  postgres=> SHOW work_mem;
  work_mem
  ----------
   4MB
  (1 row)
  ```

  Puoi aumentare la memoria di lavoro a livello di sessione prima di ordinare, raggruppare o eseguire altre operazioni, come indicato di seguito.

  ```
  SET work_mem TO '1 GB';
  ```

  Per ulteriori informazioni sulla memoria di lavoro, consulta Resource Consumption (Consumo delle risorse) nella documentazione di PostgreSQL.
- Modifica il periodo di conservazione dei log in modo che vengano archiviati per periodi di tempo più brevi. Per scoprire come, consulta File di log del database Aurora PostgreSQL.

Se il cluster database Aurora PostgreSQL è più grande di 40 TB, non utilizzare le classi di istanza db.t2, db.t3 or db.t4g. Consigliamo di utilizzare le classi di istanza database T solo per i server di sviluppo e test o altri server non di produzione. Per ulteriori informazioni, consulta Tipi di classi di istanza database.

Replica con Amazon Aurora PostgreSQL

Di seguito, puoi trovare informazioni sulla replica con Amazon Aurora PostgreSQL, incluso come monitorare e utilizzare la replica logica.

Argomenti

- Utilizzo delle repliche di Aurora

- Miglioramento della disponibilità di lettura delle repliche Aurora
- Monitoraggio della replica Aurora PostgreSQL.
- Panoramica della replica SQL logica di Postgree con Aurora
- Configurazione della replica logica per il cluster Aurora SQL Postgre DB
- Disattivazione della replica logica
- Monitoraggio della cache di scrittura e degli slot logici per la replica logica di Aurora Postgree SQL
- Esempio: utilizzo della replica logica con i cluster Aurora SQL Postgre DB
- Esempio: replica logica con SQL Aurora Postgre e AWS Database Migration Service

Utilizzo delle repliche di Aurora

Le repliche Aurora sono endpoint indipendenti in un cluster database Aurora, utilizzati specialmente per operazioni di dimensionamento della lettura e maggiore disponibilità. Un cluster Aurora DB può includere fino a 15 repliche Aurora situate nelle zone di disponibilità della regione del cluster Aurora DB. AWS

Il volume del cluster DB si compone di più copie dei dati per il cluster DB. Tuttavia, i dati nel volume del cluster sono rappresentati come singolo volume logico all'istanza primaria di scrittura e alle repliche di Aurora nel cluster di database. Per ulteriori informazioni sulle repliche di Aurora, consulta Repliche di Aurora.

Le repliche di Aurora funzionano bene per il dimensionamento della lettura perché sono dedicate completamente a operazioni di lettura nel volume del cluster. L'istanza database di scrittura gestisce le operazioni di scrittura. Il volume del cluster è condiviso tra tutte le istanze del cluster di database Aurora PostgreSQL. Di conseguenza, non occorre fare altro per replicare una copia dei dati per ciascuna replica Aurora.

Con Aurora PostgreSQL, quando una replica Aurora viene eliminata, l'endpoint dell'istanza viene rimosso immediatamente e la replica Aurora viene rimossa dall'endpoint di lettura. Se vi sono istruzioni in esecuzione nella replica Aurora da eliminare, si ha un periodo di tolleranza di tre minuti. Le istruzioni esistenti possono finire correttamente durante un periodo di tolleranza. Quando finisce il periodo di tolleranza, la replica Aurora viene chiusa ed eliminata.

I cluster Aurora PostgreSQL DB supportano le repliche Aurora in diverse regioni, utilizzando il database globale Aurora. AWS Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.

> **Note**
>
> Con la funzionalità di disponibilità di lettura, se si desidera riavviare le repliche Aurora nel cluster DB, è necessario eseguirla manualmente. Per i cluster database creati prima di questa funzionalità, il riavvio dell'istanza database di scrittura comporta il riavvio automatico delle repliche Aurora. Il riavvio automatico ristabilisce un punto di ingresso che garantisce la coerenza di lettura e scrittura in tutto il cluster di database.

Miglioramento della disponibilità di lettura delle repliche Aurora

Aurora PostgreSQL migliora la disponibilità di lettura nel cluster database soddisfacendo continuamente le richieste di lettura quando l'istanza database di scrittura si riavvia o quando la replica Aurora non è in grado di gestire il traffico di scrittura.

La funzionalità della disponibilità di lettura è disponibile per impostazione predefinita nelle seguenti versioni di Aurora PostgreSQL:

- 16.1 e tutte le versioni successive
- 15.2 o versioni successive alla 15
- 14.7 o versioni successive alla 14
- 13.10 o versioni successive alla 13
- 12.14 e versioni successive alla 12

La funzionalità di disponibilità in lettura è supportata dal database globale Aurora nelle seguenti versioni:

- 16.1 e tutte le versioni successive
- 15.4 e versioni successive
- 14.9 e versioni successive
- 13.12 e versioni successive 13
- 12.16 e versioni successive 12

Per utilizzare la funzionalità della disponibilità di lettura per un cluster database creato in una di queste versioni prima del lancio, riavviare l'istanza di scrittura del cluster database.

Quando vengono modificati i parametri statici del cluster database Aurora PostgreSQL, è necessario riavviare l'istanza di scrittura per rendere effettive le modifiche apportate ai parametri. Ad esempio, è necessario riavviare l'istanza di scrittura quando si imposta il valore di `shared_buffers`. Con la funzionalità di disponibilità in lettura di Aurora Replicas, il cluster DB mantiene una maggiore disponibilità, riducendo l'impatto su di essa al riavvio dell'istanza di writer. Le istanze di lettura non si riavviano e continuano a rispondere alle richieste di lettura. Per applicare modifiche statiche ai parametri, riavviare ogni singola istanza di lettura.

Una replica Aurora di un cluster database Aurora PostgreSQL è in grado di ripristinare gli errori di replica, come riavvii dell'istanza di scrittura, failover, replica lenta e problemi di rete, ripristinando rapidamente lo stato del database in memoria dopo la riconnessione all'istanza di scrittura. Questo approccio consente alle istanze delle repliche Aurora di raggiungere la coerenza con gli ultimi aggiornamenti dell'archiviazione mentre il database client è ancora disponibile.

Le transazioni in corso che sono in conflitto con il ripristino della replica potrebbero ricevere un errore, ma il client può provare a rieseguire queste transazioni, dopo che le istanze di lettura hanno raggiunto i livelli di prestazioni dell'istanza di scrittura.

Monitoraggio delle repliche Aurora

È possibile monitorare le repliche Aurora durante il ripristino dopo una disconnessione dell'istanza di scrittura. Utilizzare le metriche riportate di seguito per verificare le informazioni più recenti sull'istanza di lettura e per tenere traccia delle transazioni di sola lettura in corso.

- La `aurora_replica_status` funzione viene aggiornata per restituire la maggior parte delle up-to-date informazioni per l'istanza Reader quando è ancora connessa. Il timestamp dell'ultimo aggiornamento in `aurora_replica_status` è sempre vuoto per la riga corrispondente all'istanza database su cui viene eseguita la query. Ciò indica che l'istanza di lettura include i dati più recenti.
- Quando la replica Aurora si disconnette dall'istanza di scrittura e si riconnette, viene emesso il seguente evento del database:

  ```
  Read replica has been disconnected from the writer instance and
  reconnected.
  ```
- Quando una query di sola lettura viene annullata a causa di un conflitto di ripristino, è possibile che nel registro degli errori del database vengano visualizzati uno o più dei seguenti messaggi di errore:

  ```
  Canceling statement due to conflict with recovery.
  ```

```
User query may not have access to page data to replica disconnect.

User query might have tried to access a file that no longer exists.

When the replica reconnects, you will be able to repeat your command.
```

Limitazioni

Le seguenti limitazioni si applicano alle repliche Aurora con la funzionalità di disponibilità di lettura:

- Le repliche Aurora del cluster DB secondario possono riavviarsi se i dati non possono essere trasmessi in streaming dall'istanza del writer durante il ripristino della replica.
- Le repliche Aurora non supportano il ripristino delle repliche online se una replica è già in corso e pertanto questa verrà riavviata.
- Le repliche Aurora verranno riavviate quando l'istanza database si avvicina al wraparound dell'ID delle transazioni. Per ulteriori informazioni sul wraparound dell'ID delle transazioni, consulta l'argomento relativo alla risoluzione degli errori di wraparound dell'ID delle transazioni.
- È possibile che le repliche Aurora vengano riavviate quando il processo di replica viene bloccato in determinate circostanze.

Monitoraggio della replica Aurora PostgreSQL.

Il dimensionamento di lettura e l'alta disponibilità dipendono da un periodo di ritardo minimo. Puoi monitorare il ritardo di una replica Aurora rispetto all'istanza Writer DB del tuo cluster Aurora PostgreSQL DB monitorando la metrica Amazon. CloudWatch `ReplicaLag` Poiché le repliche di Aurora eseguono la lettura dallo stesso volume del cluster dell'istanza database di scrittura, il parametro `ReplicaLag` assume un significato diverso per un cluster di database Aurora PostgreSQL. Il parametro `ReplicaLag` per una replica Aurora indica il ritardo per la cache di pagina della replica Aurora rispetto a quello dell'istanza database di scrittura.

Per ulteriori informazioni sul monitoraggio delle istanze e dei parametri RDS, consulta. CloudWatch Monitoraggio dei parametri in un cluster di database Amazon Aurora

Panoramica della replica SQL logica di Postgree con Aurora

Utilizzando la funzionalità di replica logica SQL di Postgre con il cluster Aurora Postgre SQL DB, è possibile replicare e sincronizzare singole tabelle anziché l'intera istanza del database. La replica

logica utilizza un modello di pubblicazione e sottoscrizione per replicare le modifiche da un'origine in uno o più destinatari. Funziona utilizzando i record delle modifiche dal log write-ahead di Postgre (). SQL WAL L'origine, o editore, invia WAL i dati per le tabelle specificate a uno o più destinatari (sottoscrittore), replicando così le modifiche e mantenendo la tabella del sottoscrittore sincronizzata con la tabella dell'editore. L'insieme di modifiche apportate dall'editore vengono identificate mediante una pubblicazione. Gli abbonati ottengono le modifiche creando un abbonamento che definisce la connessione al database dell'autore e alle sue pubblicazioni. Uno slot di replica è il meccanismo utilizzato in questo schema per tenere traccia dell'avanzamento di una sottoscrizione.

Per i cluster Aurora Postgre SQL DB, i record WAL vengono salvati nell'archivio Aurora. Il cluster Aurora Postgre SQL DB che funge da editore in uno scenario di replica logica legge i WAL dati dallo storage Aurora, li decodifica e li invia al sottoscrittore in modo che le modifiche possano essere applicate alla tabella su quell'istanza. L'autore utilizza un decodificatore logico per decodificare i dati per l'uso da parte degli abbonati. Per impostazione predefinita, i cluster Aurora Postgre SQL DB utilizzano il plug-in Postgre nativo per l'invio SQL `pgoutput` di dati. Sono disponibili altri decodificatori logici. Ad esempio, Aurora Postgre supporta SQL anche il `wal2json` plugin che converte i dati in. WAL JSON

A partire dalle SQL versioni 14.5, 13.8, 12.12 e 11.17 di Aurora Postgre, Aurora Postgre amplia il processo di replica logica di SQL Postgre con una cache di scrittura per migliorare le prestazioni. SQL I log WAL delle transazioni vengono memorizzati nella cache locale, in un buffer, per ridurre la quantità di I/O del disco, ovvero la lettura dallo storage Aurora durante la decodifica logica. La cache write-through viene utilizzata per impostazione predefinita ogni volta che si utilizza la replica logica per il cluster Aurora Postgre DB. SQL Aurora offre diverse funzioni che puoi utilizzare per gestire la cache. Per ulteriori informazioni, consulta Monitoraggio della cache write-through della replica logica di Aurora Postgree SQL.

La replica logica è supportata da tutte le versioni di SQL Aurora Postgre attualmente disponibili. Per ulteriori informazioni, gli SQLaggiornamenti di Amazon Aurora Postgre nelle Note di rilascio per Aurora Postgre. SQL

La replica logica è supportata da Babelfish per Aurora SQL Postgre dalle seguenti versioni:

- 15.7 e versioni successive
- 16.3 e versioni successive

> **Note**
>
> Oltre alla funzionalità di replica SQL logica nativa di Postgre introdotta in Postgre SQL 10, SQL Aurora Postgre supporta anche l'estensione. `pglogical` Per ulteriori informazioni, consulta Utilizzo di pglogical per sincronizzare i dati tra le istanze.

Per ulteriori informazioni sulla replica SQL logica di Postgre, consulta Replica logica e Concetti di decodifica logica nella documentazione di Postgre. SQL

Configurazione della replica logica per il cluster Aurora SQL Postgre DB

La configurazione della replica logica richiede privilegi `rds_superuser`. Il cluster Aurora Postgre SQL DB deve essere configurato per utilizzare un gruppo di parametri del cluster DB personalizzato in modo da poter impostare i parametri necessari come descritto nella procedura seguente. Per ulteriori informazioni, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

Per configurare la replica SQL logica Postgre per un cluster Aurora Postgre DB SQL

1. Accedi al AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli il tuo cluster Aurora SQL Postgre DB.
3. Apri la scheda Configurazione. Nei dettagli dell'istanza, cercare il collegamento Gruppo di parametri con l'opzione Tipo impostata su Gruppo di parametri del cluster DB.
4. Scegli il link per aprire i parametri personalizzati associati al tuo cluster Aurora SQL Postgre DB.
5. Nel campo di ricerca Parametri, digita `rds` per trovare il parametro `rds.logical_replication`. Il valore predefinito per questo parametro è `0`, per indicare che è disattivato per impostazione predefinita.
6. Scegli Modifica parametri per accedere ai valori delle proprietà, quindi seleziona 1 dal selettore per attivare la funzione. A secondo dell'utilizzo previsto, potrebbe anche essere necessario modificare le impostazioni per i seguenti parametri. Tuttavia, in molti casi, i valori predefiniti sono sufficienti.

   - `max_replication_slots`: imposta questo parametro su un valore almeno uguale al numero totale pianificato di pubblicazioni e sottoscrizioni della replica logica. Se stai usando AWS DMS, questo parametro deve corrispondere almeno alle attività di acquisizione dei dati di modifica pianificate dal cluster, più le pubblicazioni e gli abbonamenti di replica logica.

- `max_wal_senderse max_logical_replication_workers` — Imposta questi parametri su un valore almeno uguale al numero di slot di replica logica che intendi rendere attivi o al numero di slot di replica logica che intendi siano attivi AWS DMS attività per l'acquisizione dei dati di modifica. Lasciando inattivo uno slot di replica logica si impedisce al vacuum di rimuovere le tuple obsolete dalle tabelle, pertanto ti consigliamo di monitorare gli slot di replica e rimuovere gli slot inattivi in base alle esigenze.
- `max_worker_processes`: imposta questo parametro su un valore che sia almeno uguale al totale dei valori `max_logical_replication_workers`, `autovacuum_max_workers` e `max_parallel_workers`. Su classi di istanza database di piccole dimensioni, i processi dell'operatore in background potrebbero influire sui carichi di lavoro delle applicazioni, pertanto monitorare le prestazioni del database se si imposta `max_worker_processes` su un valore più elevato di quello predefinito. (Il valore predefinito è il risultato di`GREATEST(${DBInstanceVCPU*2},8}`, il che significa che, per impostazione predefinita, è 8 o due volte l'CPUequivalente della classe di istanza DB, a seconda di quale sia il maggiore).

> **Note**
>
> È possibile modificare i valori dei parametri in un gruppo di parametri database creato dal cliente, ma non i valori dei parametri in un gruppo di parametri database predefinito.

7. Scegli Save changes (Salva modifiche).
8. Riavvia l'istanza writer del cluster Aurora SQL Postgre DB in modo che le modifiche abbiano effetto. Nella RDS console Amazon, scegli l'istanza DB principale del cluster e scegli Reboot dal menu Azioni.
9. Quando l'istanza è disponibile, puoi verificare che la replica logica sia attivata, come riportato di seguito.

   a. Utilizzalo `psql` per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

   ```
   psql --host=your-db-cluster-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=postgres --password --dbname=labdb
   ```

   b. Verifica che la replica logica sia stata abilitata utilizzando il seguente comando.

   ```
   labdb=> SHOW rds.logical_replication;
    rds.logical_replication
   ```

```
-------------------------
 on
(1 row)
```

c. Verifica che `wal_level` sia impostato su `logical`.

```
labdb=> SHOW wal_level;
  wal_level
-----------
 logical
(1 row)
```

Per un esempio di utilizzo della replica logica per mantenere una tabella di database sincronizzata con le modifiche da un cluster Aurora SQL Postgre DB di origine, vedere. Esempio: utilizzo della replica logica con i cluster Aurora SQL Postgre DB

Disattivazione della replica logica

Dopo aver completato le attività di replica, è necessario interrompere il processo di replica, eliminare gli slot di replica e disattivare la replica logica. Prima di eliminare gli slot, assicurati che non siano più necessari. Gli slot di replica attivi non possono essere eliminati.

Disattivazione della replica logica

1. Abbandonare tutti gli slot di replica.

   Per eliminare tutti gli slot di replica, connettiti all'editore ed esegui il comando seguente. SQL

   ```
   SELECT pg_drop_replication_slot(slot_name)
     FROM pg_replication_slots
    WHERE slot_name IN (SELECT slot_name FROM pg_replication_slots);
   ```

   Gli slot di replica non possono essere attivi quando si esegue questo comando.

2. Modifica il gruppo di parametri cluster database personalizzato associato all'editore come descritto in Configurazione della replica logica per il cluster Aurora SQL Postgre DB, ma imposta il parametro `rds.logical_replication` su 0

   Per ulteriori informazioni sui gruppi di parametri, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.

3. Riavviare il cluster SQL DB dell'editore Aurora Postgre per rendere effettiva la modifica al `rds.logical_replication` parametro.

Monitoraggio della cache di scrittura e degli slot logici per la replica logica di Aurora Postgree SQL

Monitora la cache write-through della replica logica e gestisci gli slot logici per migliorare le prestazioni del tuo cluster Aurora Postgree DB. SQL Di seguito, sono disponibili ulteriori informazioni sulla cache di scrittura e sugli slot logici.

Argomenti

- Monitoraggio della cache write-through della replica logica di Aurora Postgree SQL
- Gestione degli slot logici per Aurora Postgre SQL

Monitoraggio della cache write-through della replica logica di Aurora Postgree SQL

Per impostazione predefinita, SQL le versioni 14.5, 13.8, 12.12 e 11.17 di Aurora Postgre e successive utilizzano una cache di scrittura per migliorare le prestazioni per la replica logica. Senza la cache di scrittura, Aurora Postgre SQL utilizza il livello di archiviazione Aurora per implementare il processo di replica logica di Postgre nativo. SQL A tale scopo, scrive WAL i dati nello storage e quindi li rilegge per decodificarli e inviarli (replicarli) ai destinatari (abbonati). Ciò può causare colli di bottiglia durante la replica logica per i cluster Aurora Postgre DB. SQL

La cache di scrittura riduce al minimo la dipendenza dal livello di archiviazione Aurora. Invece di scrivere e leggere costantemente da questo livello, Aurora Postgre SQL utilizza un buffer per memorizzare nella cache il WAL flusso logico da utilizzare durante il processo di replica, riducendo la necessità di accedere al disco. Questo buffer è la SQL cache Postgre nativa utilizzata nella replica logica ed è identificato nei parametri del cluster Aurora Postgre DB come. SQL `rds.logical_wal_cache`

Quando utilizzi la replica logica con il tuo cluster Aurora SQL Postgre DB (per le versioni che supportano la cache write-through), puoi monitorare il rapporto di accesso alla cache per vedere se funziona bene per il tuo caso d'uso. A tale scopo, connettiti all'istanza di scrittura del cluster Aurora Postgre SQL DB utilizzando `psql` e quindi utilizza la funzione Auroraaurora_stat_logical_wal_cache, come illustrato nell'esempio seguente.

```
SELECT * FROM aurora_stat_logical_wal_cache();
```

La funzione restituisce un output come il seguente:

```
name       | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
-----------+------------+-----------+------------+-----------+----------+--------------
test_slot1 | 79183      | 24        | 0          | 24        | 100.00%  | 2022-08-05
 17:39...
test_slot2 |            | 1         | 0          |  1        | 100.00%  | 2022-08-05
 17:34...
(2 rows)
```

I valori `last_reset_timestamp` sono stati abbreviati per garantire la leggibilità. Per ulteriori informazioni su questa funzione, consulta aurora_stat_logical_wal_cache.

Aurora Postgre SQL offre le seguenti due funzioni per il monitoraggio della cache di scrittura.

- La funzione `aurora_stat_logical_wal_cache`: per la documentazione di riferimento, consulta aurora_stat_logical_wal_cache.
- La funzione `aurora_stat_reset_wal_cache`: per la documentazione di riferimento, consulta aurora_stat_reset_wal_cache.

Se ritieni che la dimensione della WAL cache regolata automaticamente non sia sufficiente per i tuoi carichi di lavoro, puoi modificarne il valore manualmente. `rds.logical_wal_cache` Considera i seguenti aspetti:

- Quando il `rds.logical_replication` parametro è disabilitato, `rds.logical_wal_cache` è impostato su zero (0).
- Quando il `rds.logical_replication` parametro è abilitato, `rds.logical_wal_cache` ha un valore predefinito di 16 MB.
- Il `rds.logical_wal_cache` parametro è statico e richiede il riavvio dell'istanza del database per rendere effettive le modifiche. Questo parametro è definito in termini di blocchi da 8 Kb. Si noti che qualsiasi valore positivo inferiore a 32 Kb viene considerato come 32 Kb. Per ulteriori informazioni su, `wal_buffers` consulta Write Ahead Log nella documentazione di Postgre. SQL

Gestione degli slot logici per Aurora Postgre SQL

L'attività di streaming viene acquisita nella vista `pg_replication_origin_status`. Per visualizzare il contenuto di questa vista, è possibile utilizzare la funzione `pg_show_replication_origin_status()`, come illustrato di seguito:

```
SELECT * FROM pg_show_replication_origin_status();
```

È possibile ottenere un elenco degli slot logici utilizzando la seguente query. SQL

```
SELECT * FROM pg_replication_slots;
```

Per eliminare uno slot logico, utilizza `pg_drop_replication_slot` con il nome dello slot, come illustrato nel seguente comando.

```
SELECT pg_drop_replication_slot('test_slot');
```

Esempio: utilizzo della replica logica con i cluster Aurora SQL Postgre DB

La procedura seguente mostra come avviare la replica logica tra due cluster Aurora SQL Postgre DB. Sia l'editore che il sottoscrittore devono essere configurati per la replica logica, come descritto in Configurazione della replica logica per il cluster Aurora SQL Postgre DB.

Il cluster Aurora Postgre SQL DB che è l'editore designato deve inoltre consentire l'accesso allo slot di replica. A tale scopo, modifica il gruppo di sicurezza associato al cloud pubblico virtuale del cluster Aurora Postgre SQL DB (VPC) basato sul servizio Amazon. VPC Consenti l'accesso in entrata aggiungendo il gruppo di sicurezza associato all'abbonato al gruppo di sicurezza dell'VPCeditore. Per ulteriori informazioni, consulta Controllare il traffico verso le risorse utilizzando i gruppi di sicurezza nella Amazon VPC User Guide.

Una volta completati questi passaggi preliminari, puoi utilizzare SQL i comandi Postgre `CREATE PUBLICATION` sull'editore e `CREATE SUBSCRIPTION` sull'abbonato, come descritto nella procedura seguente.

Per avviare il processo di replica logica tra due cluster Aurora SQL Postgre DB

Questi passaggi presuppongono che i cluster Aurora Postgre SQL DB abbiano un'istanza writer con un database in cui creare le tabelle di esempio.

1. Sul cluster DB dell'editore Aurora SQL Postgre

a. Crea una tabella utilizzando la seguente istruzione. SQL

```
CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
```

b. Inserite i dati nel database dell'editore utilizzando la seguente SQL istruzione.

```
INSERT INTO LogicalReplicationTest VALUES (generate_series(1,10000));
```

c. Verificate che i dati esistano nella tabella utilizzando la seguente SQL istruzione.

```
SELECT count(*) FROM LogicalReplicationTest;
```

d. Crea una pubblicazione per questa tabella utilizzando l'istruzione `CREATE PUBLICATION`, come descritto di seguito.

```
CREATE PUBLICATION testpub FOR TABLE LogicalReplicationTest;
```

2. Sul cluster Aurora SQL Postgre DB per abbonati

a. Nel sottoscrittore, crea la stessa tabella `LogicalReplicationTest` che hai creato nell'editore, come descritto di seguito.

```
CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
```

b. Verifica che questa tabella sia vuota.

```
SELECT count(*) FROM LogicalReplicationTest;
```

c. Crea una sottoscrizione per ottenere le modifiche dall'editore. È necessario utilizzare i seguenti dettagli sul cluster DB dell'editore Aurora SQL Postgre.

- host — L'istanza DB writer del cluster di Publisher Aurora Postgre SQL DB.
- porta: la porta di ascolto dell'istanza database di scrittura. L'impostazione predefinita per SQL Postgre è 5432.
- dbname: il nome del database.

```
CREATE SUBSCRIPTION testsub CONNECTION
   'host=publisher-cluster-writer-endpoint port=5432 dbname=db-name user=user
 password=password'
```

```
  PUBLICATION testpub;
```

> **Note**
>
> Specifica una password diversa dal prompt mostrato qui come best practice per la sicurezza.

Successivamente alla creazione della sottoscrizione, dal lato del publisher viene creato uno slot di replica logica.

d. Per verificare, per questo esempio, che i dati iniziali vengano replicati sul sottoscrittore, utilizzate la seguente SQL istruzione sul database degli abbonati.

```
SELECT count(*) FROM LogicalReplicationTest;
```

Ogni successiva modifica al publisher viene replicata nel sottoscrittore.

La replica logica influisce sulle prestazioni. Ti consigliamo di disattivare la replica logica al termine delle attività di replica.

Esempio: replica logica con SQL Aurora Postgre e AWS Database Migration Service

Puoi utilizzare il plugin AWS Database Migration Service (AWS DMS) per replicare un database o una parte di un database. Utilizzo AWS DMS per migrare i dati da un database Aurora SQL Postgre a un altro database open source o commerciale. Per ulteriori informazioni sull' AWS DMS, vedi il AWS Database Migration Service Guida per l'utente.

L'esempio seguente mostra come impostare la replica logica da un database Aurora SQL Postgre come editore e quindi utilizzare AWS DMS per la migrazione. L’esempio utilizza lo stesso publisher e lo stesso sottoscrittore creati in Esempio: utilizzo della replica logica con i cluster Aurora SQL Postgre DB.

Per configurare la replica logica con AWS DMS, hai bisogno di informazioni sul tuo editore e sul tuo abbonato da AmazonRDS. In particolare, occorrono i dettagli sull’istanza database di scrittura del publisher e sull’istanza database del sottoscrittore.

Ottenere le seguenti informazioni per l’istanza database di scrittura del publisher:

- L'identificatore del cloud privato virtuale (VPC)
- Il gruppo di sottoreti
- La zona di disponibilità (Availability Zone, AZ)
- Il gruppo VPC di sicurezza
- L'ID dell'istanza database

Ottenere le seguenti informazioni per l'istanza database del sottoscrittore:

- L'ID dell'istanza database
- Il motore di origine

Per utilizzare AWS DMS per la replica logica con Aurora Postgre SQL

1. Preparate il database dell'editore con cui lavorare AWS DMS.

   A tale scopo, i database Postgre SQL 10.x e versioni successive richiedono l'applicazione AWS DMS funzioni wrapper nel database dell'editore. Per i dettagli su questo passaggio e su quelli successivi, consulta le istruzioni in Uso di Postgre SQL versione 10.x e successive come fonte per AWS DMS nella AWS Database Migration Service Guida per l'utente.
2. Accedi al AWS Management Console e apri il AWS DMS console pressohttps://console.aws.amazon.com/dms/v2. In alto a destra, scegli lo stesso AWS Regione in cui si trovano l'editore e l'abbonato.
3. Crea un AWS DMS istanza di replica.

   Selezionare valori che siano identici a quelli dell'istanza database di scrittura del publisher. Tali valori includono le seguenti impostazioni:

   - Per VPC, scegli la stessa che per VPC l'istanza Writer DB.
   - Per Replication Subnet Group (Gruppo di sottoreti di replica), selezionare lo stesso gruppo di sottoreti con gli stessi valori dell'istanza database di scrittura. Creane uno nuovo se necessario.
   - Per Availability zone (Zona di disponibilità), selezionare la stessa zona dell'istanza database di scrittura.
   - Per VPCSecurity Group, scegli lo stesso gruppo utilizzato per l'istanza Writer DB.
4. Crea un AWS DMS endpoint per la fonte.

Specificare il publisher come endpoint di origine utilizzando le seguenti impostazioni:

- Per Endpoint type (Tipo di endpoint), selezionare Source endpoint (Endpoint di origine).
- Scegli Seleziona istanza RDS DB.
- Ad RDSesempio, scegli l'identificatore DB dell'istanza DB Writer del publisher.
- In Source engine (Motore di origine), scegliere postgres (postgres).

5. Crea un AWS DMS endpoint per il bersaglio.

   Specificare il publisher come endpoint di destinazione utilizzando le seguenti impostazioni:

   - Per Endpoint type (Tipo di endpoint), scegliere Target endpoint (Endpoint di destinazione).
   - Scegli Seleziona istanza RDS DB.
   - Ad RDSesempio, scegli l'identificatore DB dell'istanza DB del sottoscrittore.
   - Selezionare un valore per Source engine (Motore di origine). Ad esempio, se il sottoscrittore è un database Postgres, RDS scegli postgres. SQL Se il sottoscrittore è un database Aurora SQL Postgre, scegli aurora-postgresql.

6. Crea un AWS DMS attività di migrazione del database.

   Un'attività di migrazione del database serve a specificare le tabelle del database di cui effettuare la migrazione, a mappare i dati utilizzando lo schema di destinazione e a creare nuove tabelle nel database di destinazione. Utilizzare almeno le seguenti impostazioni per Task configuration (Configurazione attività):

   - Per Replication instance (Istanza di replica), scegliere l'istanza di replica creata in precedenza.
   - Per Source database endpoint (Endpoint del database di origine), selezionare l'origine del publisher creata in precedenza.
   - Per Target database endpoint (Endpoint del database di destinazione), selezionare la destinazione del sottoscrittore creata in precedenza.

   I rimanenti dettagli dell'attività dipendono dal progetto di migrazione. Per ulteriori informazioni sulla specificazione di tutti i dettagli per le DMS attività, vedere Lavorare con AWS DMSattività in AWS Database Migration Service Guida per l'utente.

Dopo AWS DMS crea l'attività, inizia la migrazione dei dati dall'editore al sottoscrittore.

Inoltro di scrittura locale in Aurora PostgreSQL

L'inoltro di scrittura locale (all'interno del cluster) consente alle applicazioni di eseguire transazioni di lettura/scrittura direttamente su una replica di Aurora. I comandi di scrittura vengono quindi inoltrati all'istanza Writer DB per il commit. È possibile utilizzare l'inoltro di scrittura locale per le applicazioni che richiedono operazioni di scrittura occasionali e richiedono read-after-write coerenza, ovvero la capacità di leggere l'ultima scrittura in una transazione.

Senza l'inoltro di scrittura, le applicazioni devono suddividere completamente tutto il traffico di lettura e scrittura, mantenendo due set di connessioni al database per inviare il traffico all'endpoint appropriato. Le repliche di lettura ricevono gli aggiornamenti in modo asincrono dall'istanza writer. Inoltre, poiché il ritardo di replica può variare tra le repliche di lettura, è difficile ottenere una coerenza di lettura globale su tutte le repliche. È necessario effettuare transazioni di lettura che richiedono read-after-write coerenza sull'istanza del database Writer. In alternativa, è necessario sviluppare una logica applicativa personalizzata complessa per sfruttare le repliche di lettura multiple per garantire la scalabilità e al contempo garantire la coerenza.

Con l'inoltro delle scritture, si evita la necessità di suddividere tali transazioni o inviarle esclusivamente all'istanza di Writer. Inoltre, non è necessario sviluppare una logica applicativa complessa per ottenere read-after-write coerenza.

L'inoltro di scrittura locale è disponibile in tutte le regioni in cui è disponibile Aurora PostgreSQL. È supportato nelle seguenti versioni di Aurora PostgreSQL:

- 16.4 e versioni successive (16)
- 15.8 e versioni successive 15
- 14.13 e versioni successive 14

L'inoltro locale delle scritture viene utilizzato per inoltrare le scritture da repliche interne alla regione. Per inoltrare le scritture da una replica globale, vedere. Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora

Argomenti

- Limitazioni e considerazioni sull'inoltro di scrittura locale in Aurora PostgreSQL
- Configurazione di Aurora SQL Postgre per l'inoltro di scrittura locale
- Utilizzo dell'inoltro di scrittura locale per Aurora PostgreSQL
- Monitoraggio dell'inoltro di scrittura locale in Aurora Postgre SQL

Limitazioni e considerazioni sull'inoltro di scrittura locale in Aurora PostgreSQL

Le seguenti limitazioni si applicano attualmente all'inoltro di scrittura locale in Aurora PostgreSQL:

- L'inoltro locale della scrittura non è supportato con RDS Proxy.
- Alcune istruzioni non sono consentite o possono produrre risultati obsoleti quando vengono utilizzate in Aurora PostgreSQL con inoltro di scrittura. Inoltre, le funzioni e le procedure definite dall'utente non sono supportate. Pertanto, l'`EnableLocalWriteForwarding`impostazione è disattivata per impostazione predefinita per i cluster DB. Prima di attivarlo, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.
- I seguenti tipi di istruzioni SQL non sono supportati con l'inoltro di scrittura:

> **Note**
>
> Queste istruzioni possono essere utilizzate implicitamente dall'utente nella propria applicazione o dedotte dal protocollo PostgreSQL. Ad esempio, la gestione delle eccezioni PL/SQL può comportare l'uso di SAVEPOINT, che non è un'istruzione supportata.

  - `ANALYZE`
  - `CLUSTER`
  - `COPY`
  - Cursori: i cursori non sono supportati, quindi assicurati di chiuderli prima di utilizzare l'inoltro di scrittura locale.
  - Istruzioni DDL (Data Definition Language)
  - `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
  - `LISTEN / NOTIFY`
  - `LOCK`
  - `SAVEPOINT`
  - `SELECT INTO`
  - `SET CONSTRAINTS`
  - Aggiornamenti delle sequenze:, `nextval()` `setval()`
  - `TRUNCATE`

- Comandi di commit in due fasi:`PREPARE TRANSACTION`,, `COMMIT PREPARED` `ROLLBACK PREPARED`
- Funzioni e procedure definite dall'utente.
- `VACUUM`

Puoi prendere in considerazione l'utilizzo delle seguenti istruzioni SQL con l'inoltro della scrittura:

- Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o un'istruzione `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata all'istanza database di scrittura ed eseguita lì.
- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`.
- `EXPLAIN`istruzioni con le istruzioni in questo elenco.
- Istruzioni `PREPARE` e `EXECUTE`.
- Istruzioni `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`.

Configurazione di Aurora SQL Postgre per l'inoltro di scrittura locale

Utilizzando le seguenti sezioni, puoi abilitare l'inoltro di scrittura locale per il tuo cluster Amazon Aurora SQL Postgre DB, configurare i livelli di coerenza e gestire le transazioni con l'inoltro di scrittura.

Abilitazione dell'inoltro di scrittura locale

Per impostazione predefinita, l'inoltro di scrittura locale non è abilitato per i cluster DB Aurora Postgre. SQL Abilitare l'inoltro di scrittura locale a livello di cluster, non a livello di istanza.

Console

Utilizzando AWS Management Console, seleziona la casella di controllo Attiva l'inoltro di scrittura locale in Leggi l'inoltro di scrittura della replica quando crei o modifichi un cluster DB.

AWS CLI

Per abilitare l'inoltro di scrittura locale con, utilizzate l'opzione. AWS CLI`--enable-local-write-forwarding` Questa opzione funziona quando si crea un nuovo cluster database secondario tramite il comando `create-db-cluster`. Inoltre, funziona quando si modifica un cluster database

esistente tramite il comando `modify-db-cluster`. È possibile disabilitare l'inoltro di scrittura locale utilizzando l'`--no-enable-local-write-forwarding`opzione con questi stessi comandi. CLI

L'esempio seguente crea un cluster Aurora Postgre SQL DB con l'inoltro di scrittura locale abilitato.

```
aws rds create-db-cluster \
--db-cluster-identifier write-forwarding-test-cluster \
--enable-local-write-forwarding \
--engine aurora-postgresql \
--engine-version 16.4 \
--master-username myuser \
--master-user-password mypassword \
--backup-retention 1
```

Vengono quindi create istanze database di scrittura e lettura in modo da poter utilizzare l'inoltro di scrittura. Per ulteriori informazioni, consulta Creazione di un cluster database Amazon Aurora.

RDS API

Per abilitare l'inoltro di scrittura locale tramite Amazon RDSAPI, imposta il `EnableLocalWriteForwarding` parametro su. `true` Questo parametro funziona quando si crea un nuovo cluster database secondario utilizzando l'operazione `CreateDBCluster`. Funziona anche quando si modifica un cluster database esistente utilizzando l'operazione `ModifyDBCluster`. Puoi disabilitare l'inoltro di scrittura locale impostando il parametro su. `EnableLocalWriteForwarding` `false`

Abilitazione dell'inoltro locale di scrittura per le sessioni del database

Il parametro `apg_write_forward.consistency_mode` è un parametro database e un parametro del cluster database che abilita l'inoltro di scrittura. È possibile specificare`SESSION`, `EVENTUALGLOBAL`, o `OFF` per il livello di coerenza di lettura. Per ulteriori informazioni sui livelli di coerenza, consulta Coerenza e isolamento per l'inoltro di scrittura locale in Aurora Postgre SQL.

A questo parametro si applicano le seguenti regole:

- Il valore predefinito è `SESSION`.
- L'inoltro locale della scrittura è disponibile solo se è `apg_write_forward.consistency_mode` impostato su `EVENTUALSESSION`, o. `GLOBAL` Questo parametro è rilevante solo nelle istanze di lettura di cluster DB con inoltro di scrittura locale abilitato.

- L'impostazione del valore su `OFF` disabilita l'inoltro locale della scrittura nella sessione.

Coerenza e isolamento per l'inoltro di scrittura locale in Aurora Postgre SQL

È possibile controllare il grado di coerenza di lettura su una replica di lettura. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nella replica di lettura prima di qualsiasi interrogazione successiva. È inoltre possibile utilizzare questa impostazione per garantire che le query sulla replica di lettura visualizzino sempre gli aggiornamenti più recenti dall'istanza DB di Writer. Ciò si verifica anche per quelli inviati da altre sessioni o altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegli il valore appropriato per il parametro a livello di sessione `apg_write_forward.consistency_mode`. Il `apg_write_forward.consistency_mode` parametro ha effetto solo sulle repliche di lettura che hanno l'inoltro di scrittura locale abilitato.

> **Note**
>
> Per il parametro `apg_write_forward.consistency_mode`, è possibile specificare i valori `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. Per impostazione predefinita, il valore è impostato su `SESSION`. L'impostazione del valore su `OFF` disabilita l'inoltro della scrittura.

All'aumentare del livello di coerenza, l'applicazione impiega più tempo ad aspettare che le modifiche vengano propagate alle repliche di lettura. È possibile scegliere l'equilibrio tra una latenza inferiore e garantire che le modifiche apportate in altre posizioni siano completamente disponibili prima dell'esecuzione delle query.

Ogni impostazione della modalità di coerenza disponibile, produce un effetto come descritto di seguito:

- `SESSION`— Una sessione su una replica di lettura che utilizza l'inoltro di scrittura locale consente di visualizzare i risultati di tutte le modifiche apportate in quella sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati nell'istanza DB del lettore corrente. Non attende i risultati aggiornati delle operazioni di scrittura eseguite in altre sessioni all'interno del cluster DB corrente.
- `EVENTUAL`— Una sessione su una replica di lettura che utilizza l'inoltro di scrittura locale potrebbe visualizzare dati leggermente obsoleti a causa del ritardo di replica. I risultati delle operazioni di scrittura nella stessa sessione non sono visibili finché l'operazione di scrittura non viene eseguita

sull'istanza DB di Writer e replicata nella replica di lettura. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati meno recenti o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica.

- `GLOBAL`— Una sessione su una replica di lettura visualizza le modifiche apportate da quella sessione. Inoltre, visualizza tutte le modifiche confermate sia dall'istanza Writer DB che da altre repliche di lettura. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando la replica di lettura contiene tutti up-to-date i dati salvati dall'istanza DB di scrittura, a partire dal momento in cui è iniziata la query.

  > **Note**
  >
  > La modalità di coerenza globale influisce sulla latenza delle query eseguite all'interno di una sessione. Eseguirà un'attesa anche quando la sessione non ha inviato alcuna query di scrittura.

- `OFF`— L'inoltro locale delle scritture è disabilitato.

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo i livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`. Tuttavia, il livello di isolamento `SERIALIZABLE` non è supportato.

Per ulteriori informazioni su tutti i parametri coinvolti nell'inoltro di scrittura, consulta Impostazioni predefinite dei parametri per l'inoltro della scrittura.

Modalità di accesso alle transazioni con inoltro di scrittura

Se la modalità di accesso alle transazioni è impostata su sola lettura, l'inoltro di scrittura locale non viene utilizzato. È possibile impostare la modalità di accesso in modalità di sola lettura e scrittura mentre si è connessi a un cluster DB e a una sessione con l'inoltro locale della scrittura abilitato.

Per ulteriori informazioni sulle modalità di accesso alle transazioni, vedere. SETTRANSACTION

Utilizzo dell'inoltro di scrittura locale per Aurora PostgreSQL

Utilizzando le seguenti sezioni, è possibile verificare se in un cluster di database è abilitato l'inoltro di scrittura locale, visualizzare le considerazioni sulla compatibilità e vedere i parametri configurabili e la configurazione dell'autenticazione. Queste informazioni forniscono i dettagli per utilizzare in modo efficace la funzionalità di inoltro locale della scrittura in Aurora PostgreSQL.

> **Note**
>
> Quando un'istanza di scrittura in un cluster che utilizza l'inoltro di scrittura locale viene riavviata, le eventuali transazioni e query attive, inoltrate sulle istanze di lettura che utilizzano l'inoltro di scrittura locale vengono chiuse automaticamente. Dopo che l'istanza di scrittura sarà nuovamente disponibile, è possibile riprovare a eseguire queste transazioni.

Verifica se in un cluster DB è abilitato l'inoltro di scrittura locale

Per determinare se è possibile utilizzare l'inoltro di scrittura locale in un cluster DB, verificate che il cluster abbia l'attributo impostato su. `LocalWriteForwardingStatus` `enabled`

Nella AWS Management Console scheda Configurazione della pagina dei dettagli del cluster, viene visualizzato lo stato Abilitato per l'inoltro locale di lettura, replica e scrittura.

Per visualizzare lo stato dell'impostazione di inoltro locale della scrittura per tutti i cluster, esegui il comando seguente. AWS CLI

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
{
"LocalWriteForwardingStatus": "enabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-1"
},
{
"LocalWriteForwardingStatus": "disabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-2"
},
{
"LocalWriteForwardingStatus": "requested",
"DBClusterIdentifier": "test-global-cluster-2"
},
{
"LocalWriteForwardingStatus": "null",
"DBClusterIdentifier": "aurora-postgresql-v2-cluster"
}
```

```
]
```

Un cluster database può avere i seguenti valori per `LocalWriteForwardingStatus`:

- `disabled`— L'inoltro locale delle scritture è disabilitato.
- `disabling`— L'inoltro locale delle scritture è in fase di disabilitazione.
- `enabled`— L'inoltro locale delle scritture è abilitato.
- `enabling`— L'inoltro locale delle scritture è in fase di attivazione.
- `null`— L'inoltro di scrittura locale non è disponibile per questo cluster DB.
- `requested`— L'inoltro locale di scrittura è stato richiesto, ma non è ancora attivo.

Impostazioni predefinite dei parametri per l'inoltro della scrittura

I gruppi di parametri del cluster Aurora includono le impostazioni per la funzionalità di inoltro di scrittura locale. Poiché si tratta di parametri cluster, tutte le istanze database in ogni cluster hanno gli stessi valori per queste variabili. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Parametro	Ambito	Tipo	Valore predefinito	Valori validi
`apg_write_forward.connect_timeout`	Sessione	secondi	30	0–2147483647
`apg_write_forward.consistency_mode`	Sessione	enum	Sessione	`SESSION`, `EVENTUAL`, `GLOBAL`, e `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Sessione	millisecondi	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Sessione	millisecondi	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Globale	int	25	1-100

Il parametro `apg_write_forward.max_forwarding_connections_percent` rappresenta il limite superiore degli slot di connessione al database che possono essere utilizzati per gestire le query inoltrate dalle istanze di lettura. Viene espresso come percentuale dell'`max_connections`impostazione per l'istanza Writer DB. Ad esempio, se `max_connections` è `800` e `apg_write_forward.max_forwarding_connections_percent` è `10`, allora l'istanza di scrittura consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione `max_connections`. Questa impostazione si applica solo all'istanza Writer DB quando nel cluster è abilitato l'inoltro di scrittura locale.

Utilizzate le seguenti impostazioni per controllare le richieste locali di inoltro di scrittura:

- `apg_write_forward.consistency_mode`— Un parametro a livello di sessione che controlla il grado di coerenza di lettura su una replica di lettura. I valori validi sono `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. Per impostazione predefinita, il valore è impostato su `SESSION`. L'impostazione del valore su `OFF` disabilita l'inoltro locale della scrittura nella sessione. Per ulteriori informazioni sui livelli di coerenza, consulta Coerenza e isolamento per l'inoltro di scrittura locale in Aurora Postgre SQL. Questo parametro è rilevante solo nelle istanze di lettura che hanno l'inoltro di scrittura locale abilitato.
- `apg_write_forward.connect_timeout`— Il numero massimo di secondi di attesa della replica di lettura quando stabilisce una connessione all'istanza DB di Writer prima di rinunciare. Il valore `0` indica un tempo di attesa infinito.
- `apg_write_forward.idle_in_transaction_session_timeout`— Il numero di millisecondi in cui l'istanza Writer DB attende l'attività su una connessione inoltrata da una replica di lettura con una transazione aperta prima di chiuderla. Se la sessione continua ad avere una transazione inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.
- `apg_write_forward.idle_session_timeout`— Il numero di millisecondi in cui l'istanza Writer DB attende l'attività su una connessione inoltrata da una replica di lettura prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.

rdswriteforward_user

`rdswriteforwarduser`È un utente che useremo per stabilire una connessione tra la replica di lettura e l'istanza DB di scrittura.

> **Note**
>
> `rdswriteforwarduser`eredita i privilegi CONNECT nei database dei clienti tramite il ruolo PUBLIC. Se i privilegi per il ruolo PUBLIC vengono revocati, dovrai CONCEDERE i privilegi CONNECT per i database a cui devi inoltrare le scritture.

Monitoraggio dell'inoltro di scrittura locale in Aurora Postgre SQL

Utilizzando le seguenti sezioni è possibile monitorare l'inoltro di scrittura locale nei SQL cluster Aurora Postgre, incluse le CloudWatch metriche pertinenti e gli eventi di attesa per tenere traccia delle prestazioni e identificare potenziali problemi.

CloudWatch Parametri Amazon e variabili di stato di Aurora SQL Postgre per l'inoltro della scrittura

I seguenti CloudWatch parametri di Amazon si applicano alle istanze Writer DB quando utilizzi l'inoltro di scrittura su una o più repliche di lettura.

CloudWatch Parametro	Unità e descrizione
`AuroraLocal ForwardingWriterDML Throughput`	Conteggio (al secondo) Numero di DML istruzioni inoltrate elaborate ogni secondo da questa istanza Writer DB.
`AuroraLocalForwardingWriter OpenSessions`	Conteggio Numero di sessioni aperte su questa istanza database di scrittura che elabora le query inoltrate.
`AuroraLocal ForwardingWriterTo talSessions`	Conteggio Numero totale di sessioni inoltrate sull'istanza database di scrittura.

Le seguenti CloudWatch metriche si applicano a ciascuna replica letta. Queste metriche vengono misurate su ogni istanza Reader DB nel cluster DB con l'inoltro di scrittura locale abilitato.

CloudWatch Parametro	Unità e descrizione
`AuroraForwardingReplicaCommitThroughput`	Conteggio (al secondo) Numero di commit in sessioni inoltrate da questa replica ogni secondo.
`AuroraForwardingReplicaDMLLatency`	Millisecondi Tempo di risposta medio in millisecondi di inoltro sulla replica. DMLs
`AuroraForwardingReplicaDMLThroughput`	Conteggio (al secondo) Numero di dichiarazioni DML inoltrate elaborate su questa replica ogni secondo.
`AuroraForwardingReplicaErrorSessionsLimit`	Conteggio Numero di sessioni rifiutate dall'istanza Writer DB perché è stato raggiunto il limite per il numero massimo di connessioni o il numero massimo di connessioni di scrittura in avanti.
`AuroraForwardingReplicaOpenSessions`	Conteggio Il numero di sessioni che utilizzano l'inoltro di scrittura locale su un'istanza di replica.
`AuroraForwardingReplicaReadWaitLatency`	Millisecondi Il tempo di attesa medio in millisecondi impiegato dalla replica sia coerente con l'istanza DB di Writer. LSN Il grado di attesa dell'istanza database in lettura dipende dall'impostazione `apg_write_forward.consistency_mode` . Per ulteriori informazioni su questa impostazione, consulta the section called “Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL”.

Attendi gli eventi per l'inoltro di scrittura locale in Aurora Postgre SQL

Amazon Aurora genera i seguenti eventi di attesa quando utilizzi l'inoltro di scrittura con Aurora Postgre. SQL

Argomenti

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

L'`IPC:AuroraWriteForwardConnect`evento si verifica quando un processo di backend sulla replica di lettura è in attesa dell'apertura di una connessione all'istanza DB di Writer.

Probabili cause di aumento delle attese

Questo evento aumenta all'aumentare del numero di tentativi di connessione da una replica di lettura al nodo di scrittura.

Azioni

Riduce il numero di connessioni simultanee da una replica di lettura al nodo di scrittura.

IPC:AuroraWriteForwardConsistencyPoint

L'`IPC:AuroraWriteForwardConsistencyPoint`evento descrive per quanto tempo una query proveniente da un nodo sulla replica di lettura attenderà prima che i risultati delle operazioni di scrittura inoltrate vengano replicati nella regione corrente. Questo evento viene generato solo se il parametro `apg_write_forward.consistency_mode` a livello di sessione è impostato su uno dei seguenti:

- `SESSION`— le interrogazioni su una replica di lettura attendono i risultati di tutte le modifiche apportate in quella sessione.
- `GLOBAL`— le interrogazioni su una replica di lettura attendono i risultati delle modifiche apportate da quella sessione, oltre a tutte le modifiche confermate sia dall'istanza DB di scrittura che dalla replica di lettura.

Per ulteriori informazioni sull'impostazione del parametro `apg_write_forward.consistency_mode`, consulta the section called “Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL”.

Probabili cause di aumento delle attese

Alcune cause comuni dei tempi di attesa più lunghi sono:

- Aumento del ritardo di replica, misurato dalla metrica di Amazon CloudWatch `ReplicaLag`. Per ulteriori informazioni su questa metrica, consulta Monitoraggio della replica Aurora PostgreSQL..
- Aumento del carico sull'istanza DB di scrittura o sulla replica di lettura.

Azioni

Modifica la modalità di coerenza in base ai requisiti dell'applicazione.

IPC:AuroraWriteForwardExecute

L'`IPC:AuroraWriteForwardExecute`evento si verifica quando un processo di backend sulla replica di lettura è in attesa del completamento di una query inoltrata e dell'ottenimento dei risultati dal nodo writer del cluster DB.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Recupero di un gran numero di righe dal nodo writer.
- L'aumento della latenza di rete tra il nodo writer e la replica di lettura aumenta il tempo impiegato dalla replica di lettura per ricevere i dati dal nodo writer.
- L'aumento del carico sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- L'aumento del carico sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Ottimizza le query per recuperare solo i dati necessari.

- Ottimizza le operazioni del linguaggio di manipolazione dei dati (DML) per modificare solo i dati necessari.
- Se il nodo di lettura, replica o scrittura è limitato dalla nostra larghezza di banda di rete, valuta la possibilità di cambiarlo in un tipo di istanza con maggiore CPU capacità CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

L'`IPC:AuroraWriteForwardGetGlobalConsistencyPoint`evento si verifica quando un processo di backend sulla replica di lettura che utilizza la modalità di coerenza è in attesa di ottenere il punto di GLOBAL coerenza globale dal nodo di scrittura prima di eseguire una query.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo impiegato dalla replica di lettura per ricevere i dati dal nodo di scrittura.
- L'aumento del carico sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- L'aumento del carico sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Modifica la modalità di coerenza in base ai requisiti dell'applicazione.
- Se il nodo di lettura, replica o scrittura è limitato dalla nostra larghezza di banda di rete, valuta la possibilità di cambiarlo in un tipo di istanza con maggiore CPU capacità CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardXactAbort

L'`IPC:AuroraWriteForwardXactAbort`evento si verifica quando un processo di backend sulla replica di lettura è in attesa del risultato di una query di pulizia remota. Le query di pulizia vengono emesse per riportare il processo allo stato ottimale dopo l'interruzione di una transazione di scrittura

inoltrata. Amazon Aurora le esegue perché è stato rilevato un errore o perché un utente ha chiamato un comando ABORT esplicito o ha annullato una query in esecuzione.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo impiegato dalla replica di lettura per ricevere i dati dal nodo di scrittura.
- L'aumento del carico sulla replica di lettura può ritardare la trasmissione della richiesta di query di pulizia dalla replica di lettura al nodo di scrittura.
- L'aumento del carico sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Indaga la causa della transazione interrotta.
- Se la replica di lettura o l'istanza DB di scrittura è limitata dalla nostra larghezza di banda di rete, valuta la possibilità di cambiarla con un tipo di istanza con maggiore CPU capacità CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardXactCommit

L'`IPC:AuroraWriteForwardXactCommit`evento si verifica quando un processo di backend sulla replica di lettura è in attesa del risultato di un comando di transazione di commit inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo impiegato dalla replica di lettura per ricevere i dati dal nodo di scrittura.
- L'aumento del carico sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- L'aumento del carico sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Se il nodo di lettura, replica o scrittura è limitato dalla nostra larghezza di banda di rete, valuta la possibilità di cambiarlo in un tipo di istanza con maggiore CPU capacità CPU o maggiore larghezza di banda di rete.

IPC:AuroraWriteForwardXactStart

L'`IPC:AuroraWriteForwardXactStart`evento si verifica quando un processo di backend sulla replica di lettura è in attesa del risultato di un comando di avvio della transazione inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra la replica di lettura e il nodo di scrittura aumenta il tempo impiegato dalla replica di lettura per ricevere i dati dal nodo di scrittura.
- L'aumento del carico sulla replica di lettura può ritardare la trasmissione della richiesta di query dalla replica di lettura al nodo di scrittura.
- L'aumento del carico sul nodo di scrittura può ritardare la trasmissione dei dati dal nodo di scrittura alla replica di lettura.

Azioni

Se il nodo di lettura, replica o scrittura è limitato dalla nostra larghezza di banda di rete, valuta la possibilità di cambiarlo in un tipo di istanza con maggiore CPU capacità CPU o maggiore larghezza di banda di rete.

Utilizzo di Aurora PostgreSQL come Knowledge Base per Amazon Bedrock

Puoi utilizzare un cluster Aurora PostgreSQL DB come Knowledge Base per Amazon Bedrock. Per ulteriori informazioni, consulta Creazione di un archivio vettoriale in Amazon Aurora. Una Knowledge Base acquisisce automaticamente dati di testo non strutturati archiviati in un bucket Amazon S3, li converte in blocchi di testo e vettori e li archivia in un database PostgreSQL. Con le applicazioni di intelligenza artificiale generativa, puoi utilizzare Agents for Amazon Bedrock per interrogare i dati archiviati nella Knowledge Base e utilizzare i risultati di tali query per aumentare le risposte fornite

dai modelli di base. Questo flusso di lavoro si chiama Retrieval Augmented Generation (RAG). Per ulteriori informazioni su RAG, vedere Retrieval Augmented Generation (RAG).

Per informazioni dettagliate sull'utilizzo di Aurora PostgreSQL per creare applicazioni AI generative utilizzando RAG, consulta questo post del blog.

Argomenti

- Prerequisiti
- Preparazione di Aurora PostgreSQL per l'utilizzo come Knowledge Base per Amazon Bedrock
- Creazione di una Knowledge Base nella console Bedrock
- Crea rapidamente una SQL Knowledge Base Aurora Postgre per Amazon Bedrock

Prerequisiti

Acquisisci familiarità con i seguenti prerequisiti per utilizzare il cluster Aurora PostgreSQL come Knowledge Base per Amazon Bedrock. A un livello elevato, è necessario configurare i seguenti servizi da utilizzare con Bedrock:

- Cluster DB Amazon Aurora PostgreSQL creato in una delle seguenti versioni:
  - 16.1 e tutte le versioni successive
  - 15.4 e versioni successive
  - 14.9 e versioni successive
  - 13.12 e versioni successive
  - 12.16 e versioni successive

> **Note**
>
> È necessario abilitare l'`pgvector`estensione nel database di destinazione e utilizzare la versione 0.5.0 o successiva. Per ulteriori informazioni, consulta pgvector v0.5.0 con indicizzazione HNSW.

- API dati RDS
- Un utente gestito in AWS Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager.

Preparazione di Aurora PostgreSQL per l'utilizzo come Knowledge Base per Amazon Bedrock

Segui i passaggi spiegati nelle sezioni seguenti per preparare Aurora PostgreSQL da utilizzare come Knowledge Base per Amazon Bedrock.

Creazione e configurazione di Aurora PostgreSQL

Per configurare Amazon Bedrock con un cluster Aurora PostgreSQL DB, devi prima creare un cluster Aurora PostgreSQL DB e prendere nota dei campi importanti per configurarlo con Amazon Bedrock. Per ulteriori informazioni sulla creazione del cluster Aurora PostgreSQL DB, consulta. Creazione e connessione di un cluster di database Aurora PostgreSQL

- Abilita Data API durante la creazione del cluster Aurora PostgreSQL DB. Per ulteriori informazioni sulle versioni supportate, consulta. Utilizzo dell'API dati Amazon RDS
- Assicurati di annotare gli Amazon Resource Names (ARN) del tuo cluster Aurora PostgreSQL DB. Ti servirà per configurare il cluster DB da utilizzare con Amazon Bedrock. Per ulteriori informazioni, consulta Amazon Resource Names (ARNs).

Connessione a un database e installazione di pgvector

È possibile connettersi ad Aurora PostgreSQL utilizzando una qualsiasi delle utilità di connessione. Per informazioni più dettagliate su queste utilità, consulta. Connessione a un cluster di database Amazon Aurora PostgreSQL In alternativa, è possibile utilizzare l'editor di query della console RDS per eseguire le query. È necessario un cluster Aurora DB con l'API RDS Data abilitata per utilizzare l'editor di query.

1. Accedi al database con il tuo utente principale e configura pgvector. Usa il seguente comando se l'estensione non è installata:

   ```
   CREATE EXTENSION IF NOT EXISTS vector;
   ```

   Usa la versione `pgvector` 0.5.0 e successive che supportano l'indicizzazione HNSW. Per ulteriori informazioni, vedere pgvector v0.5.0 con indicizzazione HNSW.
2. Utilizzate il seguente comando per verificare la versione del file installato: `pg_vector`

   ```
   SELECT extversion FROM pg_extension WHERE extname='vector';
   ```

Impostazione degli oggetti e dei privilegi del database

1. Crea uno schema specifico che Bedrock può utilizzare per interrogare i dati. Utilizzate il seguente comando per creare uno schema:

```
CREATE SCHEMA bedrock_integration;
```

2. Crea un nuovo ruolo che Bedrock può utilizzare per interrogare il database. Usa il seguente comando per creare un nuovo ruolo:

```
CREATE ROLE bedrock_user WITH PASSWORD 'password' LOGIN;
```

> **Note**
>
> Prendi nota di questa password poiché ti servirà in seguito per creare una password di Secrets Manager.

Se utilizzi `psql` client, utilizza i seguenti comandi per creare un nuovo ruolo:

```
CREATE ROLE bedrock_user LOGIN;
\PASSWORD password;
```

3. Concedi le `bedrock_user` autorizzazioni per gestire lo `bedrock_integration` schema. Ciò fornirà la possibilità di creare tabelle o indici all'interno dello schema.

```
GRANT ALL ON SCHEMA bedrock_integration to bedrock_user;
```

4. Effettua il login come `bedrock_user` e crea una tabella in. `bedrock_integration schema`

```
CREATE TABLE bedrock_integration.bedrock_kb (id uuid PRIMARY KEY, embedding
 vector(n), chunks text, metadata json, custom_metadata jsonb);
```

Questo comando creerà la `bedrock_kb` tabella nello `bedrock_integration` schema con gli incorporamenti di Titan.

Sostituisci n nel tipo di `vector(n)` dati con la dimensione appropriata per il modello di incorporamento che stai utilizzando. Utilizzate i consigli riportati di seguito per aiutarvi a selezionare le dimensioni:

- Per il modello Titan v2, usa `vector(1024)``vector(512)`, o o. `vector (256)` Per ulteriori informazioni, consulta Amazon Titan Embeddings Text.
- Per il modello Titan v1.2, usa. `vector(1536)` Per ulteriori informazioni, consulta Amazon Titan Multimodal Embeddings G1.
- Per il modello Cohere Embed, usa. `vector(1024)` Per saperne di più, consulta Modelli Cohere Embed.
- Per Cohere Embed Multilingual v3, usa. `vector(1024)`

Le prime quattro colonne sono obbligatorie. Per la gestione dei metadati, Bedrock scrive i dati dai file di metadati nella colonna. `custom_metadata` Ti consigliamo di creare questa colonna se prevedi di utilizzare metadati e filtri. Se non crei una `custom_metadata` colonna, aggiungi singole colonne per ogni attributo di metadati nella tabella prima di iniziare l'inserimento. Per ulteriori informazioni, consulta Configurare e personalizzare le query e la generazione di risposte.

5. Segui questi passaggi per creare gli indici richiesti che Bedrock utilizza per interrogare i tuoi dati:

   - Crea un indice con l'operatore coseno che Bedrock può utilizzare per interrogare i dati.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops);
     ```

   - Ti consigliamo di impostare il valore su 256 `ef_construction` per la versione `pgvector` 0.6.0 e successive che utilizzano la creazione di indici paralleli.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops) WITH (ef_construction=256);
     ```

   - Crea un indice che Bedrock può utilizzare per interrogare i dati di testo.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (to_tsvector('simple',
      chunks));
     ```

   - Se hai creato una colonna per i metadati personalizzati, crea un indice che Bedrock può utilizzare per interrogare i metadati.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (custom_metadata);
     ```

Crea un segreto in Secrets Manager

Secrets Manager consente di archiviare le credenziali Aurora in modo che possano essere trasmesse in modo sicuro alle applicazioni. Se non hai scelto l'opzione di gestione AWS dei segreti durante la creazione del cluster Aurora PostgreSQL DB, ora puoi creare un segreto. Per ulteriori informazioni sulla creazione del segreto AWS Secrets Manager del database, vedere AWS Secrets Manager database secret.

Creazione di una Knowledge Base nella console Bedrock

Durante la preparazione di Aurora PostgreSQL per l'utilizzo come archivio vettoriale per una Knowledge Base, è necessario raccogliere i seguenti dettagli da fornire alla console Amazon Bedrock.

- ARN del cluster Amazon Aurora DB: l'ARN del tuo cluster DB.
- ARN segreto: l'ARN della chiave Secrets Manager AWS per il cluster DB.
- Nome del database: il nome del database. Ad esempio, è possibile utilizzare il database predefinito*postgres*.
- Nome della tabella: si consiglia di fornire un nome qualificato dello schema durante la creazione della tabella utilizzando il comando simile al seguente:

  ```
  CREATE TABLE bedrock_integration.bedrock_kb;
  ```

  Questo comando creerà la `bedrock_kb` tabella nello `bedrock_integration` schema.
- Quando crei la tabella, assicurati di configurarla con le colonne e i tipi di dati specificati. Puoi utilizzare i nomi di colonna preferiti anziché quelli elencati nella tabella. Ricordati di prendere nota dei nomi che hai scelto come riferimento durante la configurazione della Knowledge Base.

Nome colonna	Tipo di dati	Descrizione
id	UUID (chiave primaria)	Contiene identificatori univoci per ogni record.
pezzi	Testo	Contiene i blocchi di testo non elaborato provenienti dalle origini dati.

Nome colonna	Tipo di dati	Descrizione
incorporamento	Vettore	Contiene gli incorporamenti vettoriali delle origini dati.
metadata	JSON	Contiene i metadati necessari per eseguire l'attribuzione della fonte e per consentir e l'inserimento e l'interro gazione dei dati.
metadati personalizzati	JSONB	(Facoltativo) Definisce la colonna di destinazione in cui Amazon Bedrock scrive i dettagli dei metadati dalle tue fonti di dati.

Con questi dettagli, ora puoi creare una Knowledge Base nella console Bedrock. Per informazioni più dettagliate sulla configurazione di un indice vettoriale e sulla creazione di informazioni della Knowledge Base, consulta Creare un archivio vettoriale in Amazon Aurora e Creare un archivio vettoriale in Amazon Aurora.

Dopo aver aggiunto Aurora come Knowledge Base, ora puoi importare le tue fonti di dati per ricerche e interrogazioni. Per ulteriori informazioni, consulta Inserire le fonti di dati nella Knowledge Base.

Crea rapidamente una SQL Knowledge Base Aurora Postgre per Amazon Bedrock

Il flusso di lavoro di recupero di generazione aumentata (RAG) di Amazon Bedrock si basa su dati vettoriali archiviati in un database Aurora Postgre per consentire il recupero dei contenuti. SQL In precedenza, la configurazione di Aurora Postgre SQL come archivio dati vettoriale per Bedrock Knowledge Bases era un processo in più fasi, che richiedeva numerose azioni manuali su diverse interfacce utente. Ciò ha reso difficile per i data scientist e gli sviluppatori sfruttare Aurora per i loro progetti Bedrock.

Per migliorare l'esperienza utente, AWS ha creato una nuova opzione CloudFormation basata sulla creazione rapida che semplifica il processo di configurazione. Con Aurora quick create, ora puoi

effettuare il provisioning di un cluster Aurora SQL Postgre DB preconfigurato come archivio vettoriale per le tue Knowledge Base Amazon Bedrock con un solo clic.

Argomenti

- Regioni supportate e versioni di Aurora Postgre SQL
- Comprendere il processo di creazione rapida
- Vantaggi dell'utilizzo di Aurora quick create
- Limitazioni del processo di creazione rapida di Aurora

Regioni supportate e versioni di Aurora Postgre SQL

L'opzione di creazione rapida Aurora è disponibile in tutte le AWS regioni che supportano Amazon Bedrock Knowledge Base. Per impostazione predefinita, crea un cluster Aurora Postgre SQL DB con la versione 15.7. Per ulteriori informazioni sulle regioni supportate, consulta Modelli e regioni supportati per Amazon Bedrock Knowledge Base.

Comprendere il processo di creazione rapida

Il processo di creazione rapida fornisce automaticamente le seguenti risorse per configurare un SQL database Amazon Aurora Postgre come archivio dati vettoriale per la tua Bedrock Knowledge Base:

Un cluster Aurora Postgre SQL DB nel tuo account, configurato con impostazioni predefinite.

- ACUs(Unità di capacità Aurora) sono impostate da 0 a 16. Ciò consente al tuo archivio vettoriale di ridimensionarsi fino a zero quando non è in uso, risparmiando sui costi di elaborazione. ACUsPuò essere regolato in un secondo momento nella RDS console Amazon.
- HNSWIndice (Hierarchical Navigable Small World) che utilizza la distanza euclidea come misura di somiglianza per gli incorporamenti vettoriali Bedrock memorizzati in Aurora.
- L'istanza DB è un'istanza v2 senza server.
- Il cluster è associato all'impostazione predefinita VPC e alle sottoreti e ha i dati abilitati. RDS API
- Le credenziali di amministratore del cluster sono gestite da AWS Secrets Manager.

Oltre alle impostazioni predefinite, vengono configurate automaticamente le seguenti impostazioni. Durante il processo, vedrai delle schermate che spiegano il flusso di lavoro.

- Eseguire il seeding del cluster Aurora con gli oggetti di database necessari:

  - Crea l'estensione pgvector, lo schema, il ruolo e le tabelle necessari per la Bedrock Knowledge Base.
  - Registra un utente del database con privilegi limitati per consentire a Bedrock di interagire con il cluster.
- Durante tutto il processo di approvvigionamento delle risorse verrà visualizzato un banner sullo stato di avanzamento, che consente di tenere traccia dello stato dei seguenti eventi secondari:
  - Creazione di cluster Aurora
  - Seminare l'ammasso Aurora
  - Creazione di una base di conoscenze

  Il banner rimane visibile fino alla creazione completa della knowledge base, anche se si esce dalla pagina e si torna indietro.
- Puoi fare clic `View details` sul banner di avanzamento per visualizzare lo stato di ogni passaggio. Per ulteriori informazioni sugli eventi durante la creazione della knowledge base, scegli il CloudFormation link nella schermata di visualizzazione dei dettagli. Una volta completato il processo, la tua nuova Bedrock Knowledge Base sarà pronta per l'uso.
- Lo stack di tutte le risorse IDs per la creazione rapida è disponibile nei tag della Bedrock Knowledge Base, nel caso in cui fosse necessario farvi riferimento.

Una Bedrock Knowledge Base, con la configurazione del cluster Aurora appena fornito al momento della creazione del vector store.

Vantaggi dell'utilizzo di Aurora quick create

- Il processo di creazione rapida CloudFormation basato su Aurora riduce significativamente il tempo e la complessità necessari per utilizzare Aurora come archivio vettoriale.
- Aurora offre prestazioni eccellenti, scalabilità vettoriale e vantaggi in termini di costi con la possibilità di scalare fino a zero costi di elaborazione quando non viene utilizzata.
- Il processo di creazione rapida semplifica l' end-to-endesperienza, consentendoti di creare e configurare facilmente le tue Bedrock Knowledge Base utilizzando Aurora.
- I clienti possono basarsi su CloudFormation un modello per personalizzare il provisioning con le proprie configurazioni.

Limitazioni del processo di creazione rapida di Aurora

- Con l'opzione di creazione rapida Aurora, il cluster DB viene fornito con configurazioni predefinite. Tuttavia, queste impostazioni predefinite potrebbero non soddisfare i requisiti specifici o il caso d'uso previsto. La creazione rapida non offre opzioni per modificare le configurazioni durante il processo di provisioning. Le configurazioni vengono impostate automaticamente per semplificare l'esperienza di distribuzione. Se devi personalizzare la configurazione del cluster Aurora DB, puoi farlo dopo la distribuzione iniziale tramite creazione rapida nella console AmazonRDS.
- Sebbene il flusso di creazione rapido semplifica il processo di configurazione, il tempo necessario per creare il cluster Aurora DB è ancora di circa 10 minuti, lo stesso di una distribuzione manuale. Ciò è dovuto al tempo necessario per effettuare il provisioning dell'infrastruttura Aurora.
- L'opzione di creazione rapida è progettata per la sperimentazione e la configurazione rapida. Le risorse create tramite la creazione rapida potrebbero non essere adatte all'uso in produzione e non sarà possibile migrarle direttamente in un ambiente di produzione del proprio. VPC

Integrazione di Amazon Aurora SQL Postgre con altri servizi AWS

Amazon Aurora si integra con altri AWS servizi in modo da poter estendere il cluster Aurora Postgre SQL DB per utilizzare funzionalità aggiuntive nel cloud. AWS Il cluster Aurora Postgre SQL DB può utilizzare i AWS servizi per eseguire le seguenti operazioni:

- Raccogli, visualizza e valuta rapidamente le prestazioni delle tue SQL istanze DB Aurora Postgre con Amazon Performance Insights. RDS Performance Insights amplia le funzionalità di RDS monitoraggio di Amazon esistenti per illustrare le prestazioni del database e aiutarti ad analizzare eventuali problemi che lo riguardano. Con la dashboard Performance Insights, puoi visualizzare il carico del database e filtrarlo per attese, SQL istruzioni, host o utenti. Per ulteriori informazioni su Performance Insights, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.
- Configura il tuo cluster Aurora Postgre SQL DB per pubblicare i dati di log su Amazon Logs. CloudWatch CloudWatch I log forniscono uno storage estremamente durevole per i tuoi record di log. Con CloudWatch Logs, è possibile eseguire analisi in tempo reale dei dati di registro e utilizzarli CloudWatch per creare allarmi e visualizzare metriche. Per ulteriori informazioni, consulta Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs CloudWatch .
- Importa dati da un bucket Amazon S3 a un cluster Aurora Postgre SQL DB o esporta dati da un cluster Aurora Postgre DB a un bucket Amazon S3. SQL Per ulteriori informazioni, consulta

Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL e Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.

- Aggiungi previsioni basate sull'apprendimento automatico alle applicazioni di database utilizzando il linguaggio. SQL L'apprendimento automatico Aurora utilizza un'integrazione altamente ottimizzata tra il database Aurora e i servizi di AWS machine learning (ML) AI e SageMaker Amazon Comprehend. Per ulteriori informazioni, consulta Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora Postgre SQL.
- Richiama AWS Lambda funzioni da un cluster Aurora SQL Postgre DB. Per fare ciò, usa l'SQLestensione `aws_lambda` Postgre fornita con Aurora Postgre. SQL Per ulteriori informazioni, consulta .
- Integra le query di Amazon Redshift e Aurora Postgre. SQL Per ulteriori informazioni, consulta la sezione Guida introduttiva all'utilizzo di query federate su Postgre nella SQL Amazon Redshift Database Developer Guide.

Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL

Puoi importare i dati che sono stati archiviati utilizzando Servizio di archiviazione semplice Amazon in una tabella su un'istanza cluster database Aurora PostgreSQL. A questo scopo, installa innanzitutto l'estensione Aurora PostgreSQL `aws_s3`. Questa estensione fornisce le funzioni utilizzate per importare i dati da un bucket Amazon S3. Un bucket è un container Amazon S3 per oggetti e file. I dati possono trovarsi in un file con valori separati da virgole (CSV), un file di testo o un file compresso (gzip). Di seguito, sono fornite informazioni su come installare l'estensione e come importare dati da Amazon S3 in una tabella.

Per eseguire l'importazione da Amazon S3 a , il database deve eseguire PostgreSQL versione 10.7 o successive. Aurora PostgreSQL.

Se Amazon S3 non contiene dati, occorre innanzitutto creare un bucket e archiviare i dati.
Per ulteriori informazioni, consulta i seguenti argomenti nella Guida per l'utente di Servizio di archiviazione semplice Amazon.

- Creazione di un bucket
- Aggiunta di un oggetto a un bucket.

È supportata l'importazione multiaccount da Amazon S3. Per ulteriori informazioni, consulta Concessione di autorizzazioni multiaccount nella Guida per l'utente di Amazon Simple Storage Service.

Puoi utilizzare la chiave gestita dal cliente per la crittografia durante l'importazione dei dati da S3. Per ulteriori informazioni, consulta Chiavi KMS archiviate in AWS KMS nella Guida per l'utente di Amazon Simple Storage Service.

> **Note**
>
> L'importazione di dati da Amazon S3 non è supportata per Aurora Serverless v1. È supportata per Aurora Serverless v2.

Argomenti

- Installazione dell'estensione aws_s3
- Panoramica dell'importazione di dati dai dati di Amazon S3
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)
- Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL
- Informazioni di riferimento sulle funzioni

Installazione dell'estensione aws_s3

Prima di poter utilizzare Amazon S3 con il cluster Aurora Postgre SQL DB, , devi installare l'estensione. `aws_s3` Questa estensione fornisce funzioni per l'importazione dei dati da Amazon S3. Fornisce inoltre funzioni per esportare dati da un'istanza di un cluster Aurora Postgree DB e per un'istanza SQL Postgree DB . Per ulteriori informazioni, consulta Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3. L'estensione `aws_s3` dipende da alcune delle funzioni helper nell'estensione `aws_commons`, che vengono installate automaticamente quando necessario.

Per installare l'estensione **`aws_s3`**

1. Usa psql (orpgAdmin) per connetterti all'istanza writer del tuo RDS cluster Aurora Postgre DB per l'istanza SQL Postgre DB . `rds_superuser` Se hai mantenuto il nome predefinito durante il processo di configurazione, esegui la connessione come `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Per installare l'estensione, esegui il comando seguente.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. Per verificare che l'estensione sia installata, puoi usare il metacomando psql \dx.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Le funzioni per importare dati da Amazon S3 ed esportare dati in Amazon S3 sono ora disponibili per l'uso.

Panoramica dell'importazione di dati dai dati di Amazon S3

Per importare i dati S3 in Aurora PostgreSQL

Raccogli innanzitutto i dettagli che devi fornire alla funzione. Questi includono il nome della tabella sull'istanza del cluster Aurora PostgreSQL DB, l'istanza DB RDS per PostgreSQL e il nome del bucket, il i dati di Amazon S3. Regione AWS Per ulteriori informazioni, consulta Visualizzazione di un oggetto nella Guida per l'utente di Servizio di archiviazione semplice Amazon.

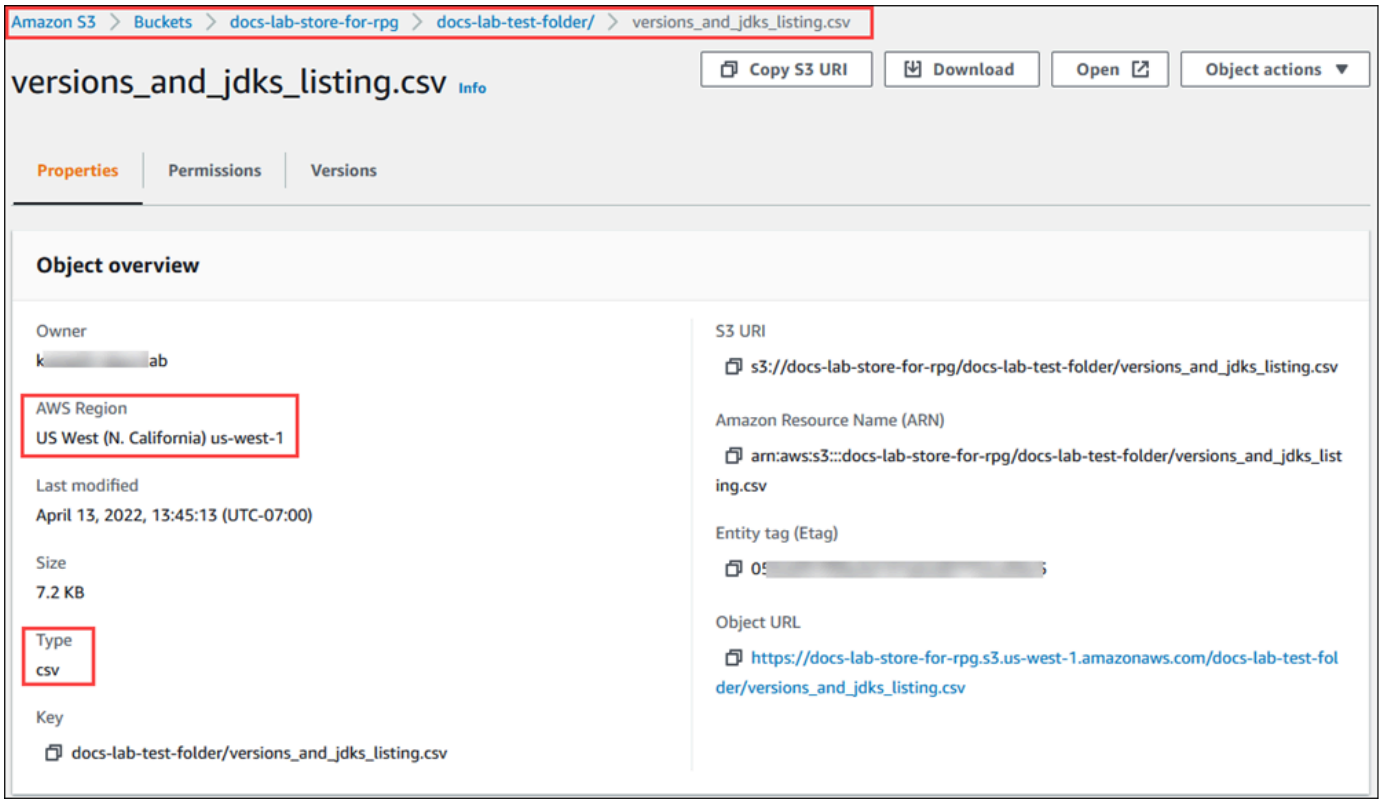
> **Note**
>
> L'importazione in più parti da Amazon S3 non è attualmente supportata.

1. Otteni il nome della tabella in cui la funzione `aws_s3.table_import_from_s3` deve importare dati. Il seguente comando, ad esempio, crea una tabella `t1` che può essere utilizzata in fasi successive.

```
postgres=> CREATE TABLE t1
    (col1 varchar(80),
    col2 varchar(80),
    col3 varchar(80));
```

2. Ottieni i dettagli relativi al bucket Amazon S3 e i dati da importare. A tale scopo, apri la console Amazon S3 all'indirizzo e scegli https://console.aws.amazon.com/s3/Bucket. Individua il bucket contenente i dati nell'elenco. Scegli il bucket, apri la pagina Object overview (Panoramica degli oggetti) e quindi scegli Properties (Proprietà).

   Prendi nota del nome, del percorso, del e del tipo di file del Regione AWS bucket. Il nome della risorsa Amazon (ARN) è richiesto in un secondo momento per configurare l'accesso ad Amazon S3 tramite un ruolo IAM. Per ulteriori informazioni, consulta Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3). Un esempio è illustrato nell'immagine seguente.



3. Puoi verificare il percorso dei dati nel bucket Amazon S3 utilizzando il comando. AWS CLI `aws s3 cp` Se le informazioni sono corrette, questo comando scarica una copia del file Amazon S3.

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. Configura le autorizzazioni sul cluster database Aurora PostgreSQL per consentire l'accesso al file sul bucket Amazon S3. A tale scopo, utilizzi un ruolo AWS Identity and Access Management (IAM) o credenziali di sicurezza. Per ulteriori informazioni, consulta Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3).
5. Fornisci alla funzione `create_s3_uri` il percorso e gli altri dettagli dell'oggetto Amazon S3 raccolti (vedi passaggio 2) per costruire un oggetto URI Amazon S3. Per ulteriori informazioni su questa funzione, consulta aws_commons.create_s3_uri. Di seguito è riportato un esempio di costruzione dell'oggetto durante una sessione psql.

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
   'us-west-1'
) AS s3_uri \gset
```

   Nella fase seguente, si passa questo oggetto (`aws_commons._s3_uri_1`) alla funzione `aws_s3.table_import_from_s3` per importare i dati nella tabella.
6. Invoca la funzione `aws_s3.table_import_from_s3` per importare dati da Amazon S3 nella tabella. Per informazioni di riferimento, consulta aws_s3.table_import_from_s3. Per alcuni esempi, consulta Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL.

Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3)

Per importare i dati da un file Amazon S3, concedere al cluster database Aurora PostgreSQL a l'autorizzazione ad accedere al bucket Amazon S3 che contiene il file. Puoi concedere l'accesso a un bucket Amazon S3 in uno dei due modi descritti negli argomenti seguenti.

Argomenti

- Utilizzo di un ruolo IAM per accedere a un bucket Amazon S3.
- Utilizzo delle credenziali di sicurezza per accedere a un bucket Amazon S3
- Risoluzione dei problemi di accesso a Amazon S3

Utilizzo di un ruolo IAM per accedere a un bucket Amazon S3.

Prima di caricare i dati da un file Amazon S3, è necessario concedere al cluster di database Aurora PostgreSQL l'autorizzazione per accedere al bucket Amazon S3 che contiene il file. In questo modo non dovrai gestire ulteriori informazioni sulle credenziali né fornirle nella chiamata della funzione aws_s3.table_import_from_s3.

Per svolgere questa operazione, creare una policy IAM che fornisca accesso al bucket Amazon S3. Creare un ruolo IAM e collegarvi la policy. Quindi, assegnare il ruolo IAM al cluster .

> **Note**
>
> Non è possibile associare un ruolo IAM a un cluster di database Aurora Serverless v1, quindi i seguenti passaggi non sono attinenti.

Per consentire a un cluster database Aurora PostgreSQL l'accesso ad Amazon S3 tramite un ruolo IAM

1. Creare una policy IAM

   Questa policy fornisce le autorizzazioni bucket e di oggetto che consentono al cluster di database Aurora PostgreSQL di accedere a Amazon S3.

   Includere nella policy le seguenti operazioni necessarie per consentire il trasferimento dei file da un bucket Amazon S3 a Aurora PostgreSQL:

   - `s3:GetObject`
   - `s3:ListBucket`

   Includere nella policy le seguenti risorse per identificare il bucket Amazon S3 e gli oggetti nel bucket. Questo mostra il formato Amazon Resource Name (ARN) per accedere a Amazon S3.

   - arn:aws:s3::: *`amzn-s3-demo-bucket`*
   - arn:aws:s3::: /* *`amzn-s3-demo-bucket`*

   Per ulteriori informazioni sulla creazione di una policy IAM per Aurora PostgreSQL, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. Consulta anche il Tutorial: Creare e collegare la prima policy gestita dal cliente nella Guida per l'utente di IAM.

Il AWS CLI comando seguente crea una policy IAM denominata con queste opzioni. `rds-s3-import-policy` Concede l'accesso a un bucket denominato *amzn-s3-demo-bucket*.

> **Note**
>
> Prendi nota del nome della risorsa Amazon (ARN) della policy restituita mediante questo comando. L'ARN sarà richiesto in una fase successiva quando si associa la policy a un ruolo IAM.

Example

Per LinuxmacOS, oUnix:

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

Per Windows:

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
```

```
    "Statement": [
      {
        "Sid": "s3import",
        "Action": [
          "s3:GetObject",
          "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
          "arn:aws:s3:::amzn-s3-demo-bucket",
          "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
      }
    ]
  }'
```

2. Crea un ruolo IAM.

   In questo modo, Aurora PostgreSQL può assumere questo ruolo IAM per tuo conto, per accedere ai bucket Amazon S3. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente IAM.

   Si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy basate sulle risorse per limitare le autorizzazioni del servizio a una risorsa specifica. Questo è il modo più efficace per proteggersi dal problema di deputy confused.

   Se si utilizzano entrambe le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID account, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account quando viene utilizzato nella stessa dichiarazione di policy.

   - Utilizzare `aws:SourceArn` se si desidera un accesso cross-service per una singola risorsa.
   - Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso cross-service.

   Nella policy, assicurarsi di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. L'esempio seguente mostra come eseguire questa operazione utilizzando il AWS CLI comando per creare un ruolo denominato`rds-s3-import-role`.

Example

Per LinuxmacOS, oUnix:

```
aws iam create-role \
   --role-name rds-s3-import-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
             }
       }
     ]
   }'
```

Per Windows:

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
```

```
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
            }
        }
      ]
   }'
```

3. Collegare la policy IAM al ruolo IAM creato.

   Il AWS CLI comando seguente collega la policy creata nel passaggio precedente al ruolo denominato `rds-s3-import-role` Replace *`your-policy-arn`* con l'ARN della policy annotato in un passaggio precedente.

   Example

   PerLinux, omacOS: Unix

   ```
   aws iam attach-role-policy \
      --policy-arn your-policy-arn \
      --role-name rds-s3-import-role
   ```

   Per Windows:

   ```
   aws iam attach-role-policy ^
      --policy-arn your-policy-arn ^
      --role-name rds-s3-import-role
   ```

4. Aggiungere il ruolo IAM al cluster.

   A tale scopo, utilizzare AWS Management Console o AWS CLI, come descritto di seguito.

Console

Per aggiungere un ruolo IAM al cluster di database PostgreSQL tramite la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere il nome del cluster di database PostgreSQL per visualizzarne i dettagli.

3. Nella scheda Connettività e sicurezza, nella sezione Gestisci ruoli IAM, scegli il ruolo da aggiungere in Aggiungi ruoli IAM a questa istanza del .
4. In Feature (Caratteristica), scegliere s3Import.
5. Scegliere Add role (Aggiungi ruolo).

AWS CLI

Per aggiungere un ruolo IAM a un cluster di database PostgreSQL tramite CLI

- Utilizzare il seguente comando per aggiungere il ruolo al cluster di database PostgreSQL denominato `my-db-cluster`. Sostituire *`your-role-arn`* con l'ARN del ruolo annotato in precedenza. Utilizzare `s3Import` come valore dell'opzione `--feature-name`.

  Example

  Per LinuxmacOS, oUnix:

  ```
  aws rds add-role-to-db-cluster \
     --db-cluster-identifier my-db-cluster \
     --feature-name s3Import \
     --role-arn your-role-arn   \
     --region your-region
  ```

  Per Windows:

  ```
  aws rds add-role-to-db-cluster ^
     --db-cluster-identifier my-db-cluster ^
     --feature-name s3Import ^
     --role-arn your-role-arn ^
     --region your-region
  ```

API RDS

Per aggiungere un ruolo IAM per un' del cluster DB PostgreSQL utilizzando l'API Amazon RDS, chiama l'operazione. AddRoleToDBCluster

Utilizzo delle credenziali di sicurezza per accedere a un bucket Amazon S3

Se preferisci, puoi utilizzare le credenziali di sicurezza per fornire accesso a un bucket Amazon S3 invece di fornire accesso con un ruolo IAM. A tale scopo, specifica il parametro `credentials` nella chiamata di funzione aws_s3.table_import_from_s3.

Il `credentials` parametro è una struttura di tipo `aws_commons._aws_credentials_1` che contiene le credenziali. AWS Utilizzare la funzione aws_commons.create_aws_credentials per impostare la chiave di accesso e la chiave segreta in una struttura `aws_commons._aws_credentials_1`, come illustrato di seguito.

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

Dopo aver creato la struttura `aws_commons._aws_credentials_1` , utilizzare la funzione aws_s3.table_import_from_s3 con il parametro `credentials` per importare i dati, come illustrato di seguito.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

Oppure si può includere la chiamata inline di funzione aws_commons.create_aws_credentials all'interno della chiamata di funzione `aws_s3.table_import_from_s3`.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Risoluzione dei problemi di accesso a Amazon S3

Se riscontri problemi di connessione quanto tenti di importare i dati da Amazon S3, segui questi suggerimenti:

- Risoluzione dei problemi di identità e accesso in Amazon Aurora

- Risoluzione dei problemi di Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service
- Risoluzione dei problemi di Amazon S3 e IAM nella Guida per l'utente di IAM.

Importazione di dati da Amazon S3 nel cluster database Aurora PostgreSQL

Importa i dati dal bucket Amazon S3 utilizzando la funzione `table_import_from_s3` dell'estensione aws_s3. Per informazioni di riferimento, consulta aws_s3.table_import_from_s3.

> **Note**
>
> Gli esempi seguenti utilizzano il metodo del ruolo IAM per consentire l'accesso al bucket Amazon S3. Pertanto, le chiamate della funzione `aws_s3.table_import_from_s3` non includono parametri di credenziali.

Di seguito viene illustrato un tipico esempio.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

I parametri sono i seguenti:

- `t1` – Il nome della tabella nel cluster di database PostgreSQL in cui copiare i dati.
- `''` – Un elenco opzionale di colonne nella tabella di database. Questo parametro può essere utilizzato per indicare quali colonne di dati S3 vanno in quali colonne della tabella. Se non viene specificata alcuna colonna, tutte le colonne vengono copiate nella tabella. Per un esempio di utilizzo di un elenco di colonne, consulta Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato.
- `(format csv)` – Argomenti COPY di PostgreSQL. La procedura di copia utilizza gli argomenti e il formato del comando COPY di PostgreSQL per importare i dati. Le scelte di formato includono valori separati da virgole (CSV), come mostrato in questo esempio, testo e file binario. Il valore predefinito è testo.

- `s3_uri` – Una struttura contenente le informazioni che identificano il file Amazon S3. Per un esempio di utilizzo della funzione aws_commons.create_s3_uri per creare una struttura `s3_uri`, consulta Panoramica dell'importazione di dati dai dati di Amazon S3.

Per ulteriori informazioni su questa funzione, consulta aws_s3.table_import_from_s3.

La funzione restituisce `aws_s3.table_import_from_s3`. Per specificare altri tipi di file da importare da un bucket Amazon S3, consulta uno dei seguenti esempi.

> **Note**
>
> L'importazione di un file da 0 byte genererà un errore.

Argomenti

- Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato
- Importazione di un file compresso (gzip) Amazon S3
- Importazione di un file Amazon S3 codificato

Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato

Il seguente esempio mostra come importare un file che utilizza un delimitatore personalizzato. Mostra anche come controllare dove inserire i dati nella tabella di database utilizzando il parametro `column_list` della funzione aws_s3.table_import_from_s3.

In questo esempio si presuppone che le seguenti informazioni siano organizzate in colonne delimitate da pipe nel file Amazon S3.

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
4|foo4|bar4|elephant4
...
```

Per importare un file che utilizza un delimitatore personalizzato

1. Creare una tabella nel database per i dati importati.

```
postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
```

2. Utilizzare il seguente formato della funzione aws_s3.table_import_from_s3 per importare i dati dal file Amazon S3.

   Si può includere la chiamata inline di funzione aws_commons.create_s3_uri all'interno della chiamata di funzione `aws_s3.table_import_from_s3` per specificare il file.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   'test',
   'a,b,d,e',
   'DELIMITER ''|''',
   aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
east-2')
);
```

I dati sono ora nella tabella nelle seguenti colonne.

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Importazione di un file compresso (gzip) Amazon S3

Il seguente esempio mostra come importare da Amazon S3 un file compresso con gzip. Il file importato deve avere i seguenti metadati Amazon S3:

- Chiave: `Content-Encoding`
- Valore: `gzip`

Se carichi il file utilizzando il AWS Management Console, i metadati vengono in genere applicati dal sistema. Per informazioni sul caricamento di file su Amazon S3 utilizzando AWS Management Console l', la o l'API, AWS CLIconsulta Uploading objects nella Amazon Simple Storage Service User Guide.

Per ulteriori informazioni sui metadati di Amazon S3 e i dettagli sui metadati forniti dal sistema, consulta la sezione Modifica dei metadati degli oggetti nella console Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.

Importare il file gzip nel cluster di database Aurora PostgreSQL, come illustrato di seguito.

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

Importazione di un file Amazon S3 codificato

Il seguente esempio mostra come importare da Amazon S3 un file codificato con Windows-1252

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

Informazioni di riferimento sulle funzioni

Funzioni

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri
- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

L'estensione `aws_s3` fornisce la funzione `aws_s3.table_import_from_s3`. Il valore restituito è testo.

Sintassi

I parametri richiesti sono `table_name`, `column_list` e `options`. Identificano la tabella di database e specificano il modo in cui i dati vengono copiati nella tabella

Puoi inoltre utilizzare i seguenti parametri:

- Il parametro `s3_info` specifica il file Amazon S3 da importare.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1
)
```

- Il parametro `credentials` specifica le credenziali per accedere a Amazon S3. Quando si utilizza questo parametro, non si utilizza alcun ruolo. IAM

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

Parametri

table_name

Una stringa di testo obbligatoria contenente il nome della tabella del SQL database Postgre in cui importare i dati.

column_list

Una stringa di testo obbligatoria contenente un elenco opzionale delle colonne della tabella del SQL database Postgre in cui copiare i dati. Se la stringa è vuota, vengono utilizzate tutte le colonne della tabella. Per un esempio, consulta Importazione di un file Amazon S3 che utilizza un delimitatore personalizzato.

options

Una stringa di testo obbligatoria contenente argomenti per il comando SQL `COPY` Postgre. Questi argomenti specificano come i dati devono essere copiati nella tabella Postgre. SQL Per maggiori dettagli, consulta la documentazione di Postgre. SQL COPY

s3_info

Un tipo composito `aws_commons._s3_uri_1` contenente le seguenti informazioni sull'oggetto S3:

- `bucket` – Il nome del bucket Amazon S3 contenente il file.
- `file_path` – Il nome file di Amazon S3, incluso il percorso.
- `region`— Il AWS Regione in cui si trova il file. Per un elenco di AWS Nomi delle regioni e valori associati, vedereRegioni e zone di disponibilità.

credenziali

Un tipo composito `aws_commons._aws_credentials_1` contenente le seguenti credenziali da utilizzare per l'operazione di importazione:

- Chiave di accesso
- Chiave segreta
- Token di sessione

Per informazioni sulla creazione di una struttura composita `aws_commons._aws_credentials_1`, consulta aws_commons.create_aws_credentials.

Sintassi alternativa

Per un aiuto nei test, si può utilizzare un set più ampio di parametri al posto dei parametri `s3_info` e `credentials`. Di seguito vengono riportate le variazioni di sintassi aggiuntive per la funzione `aws_s3.table_import_from_s3`.

- Invece di utilizzare il parametro `s3_info` per identificare un file Amazon S3, utilizzare la combinazione dei parametri `bucket`, `file_path` e `region`. Con questa forma di funzione, l'accesso ad Amazon S3 è fornito da un IAM ruolo sull'istanza DB di SQL Postgre.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text
)
```

- Invece di utilizzare il parametro `credentials` per specificare l'accesso a Amazon S3, utilizzare la combinazione dei parametri `access_key`, `session_key` e `session_token`.

```
aws_s3.table_import_from_s3 (
```

```
    table_name text,
    column_list text,
    options text,
    bucket text,
    file_path text,
    region text,
    access_key text,
    secret_key text,
    session_token text
)
```

Parametri alternativi

bucket

Una stringa di testo contenente il nome del bucket Amazon S3 che contiene il file

file_path

Una stringa di testo contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo che identifica il Regione AWS posizione del file. Per un elenco di Regione AWS nomi e valori associati, vedereRegioni e zone di disponibilità.

chiave_accesso

Una stringa di testo contenente la chiave di accesso da utilizzare per l'operazione di importazione. L'impostazione predefinita èNULL.

secret_key

Una stringa di testo contenente la chiave segreta da utilizzare per l'operazione di importazione. L'impostazione predefinita èNULL.

session_token

(Opzionale) Una stringa di testo contenente la chiave di sessione da utilizzare per l'operazione di importazione. L'impostazione predefinita èNULL.

aws_commons.create_s3_uri

Crea una struttura `aws_commons._s3_uri_1` per conservare le informazioni relative al file Amazon S3. Si utilizzano i risultati della funzione `aws_commons.create_s3_uri` nel parametro `s3_info` della funzione aws_s3.table_import_from_s3.

Sintassi

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parametri

bucket

Una stringa di testo obbligatoria contenente il nome del bucket Amazon S3 del file.

file_path

Una stringa di testo obbligatoria contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo obbligatoria contenente Regione AWS in cui si trova il file. Per un elenco di Regione AWS nomi e valori associati, vedereRegioni e zone di disponibilità.

aws_commons.create_aws_credentials

Imposta una chiave di accesso e una chiave segreta in una struttura `aws_commons._aws_credentials_1`. Si utilizzano i risultati della funzione `aws_commons.create_aws_credentials` nel parametro `credentials` della funzione aws_s3.table_import_from_s3.

Sintassi

```
aws_commons.create_aws_credentials(
   access_key text,
   secret_key text,
   session_token text
)
```

Parametri

chiave_accesso

Una stringa di testo obbligatoria contenente la chiave di accesso da utilizzare per l'importazione di un file Amazon S3. L'impostazione predefinita èNULL.

secret_key

Una stringa di testo obbligatoria contenente la chiave segreta da utilizzare per l'importazione di un file Amazon S3. L'impostazione predefinita èNULL.

session_token

Una stringa di testo opzionale contenente il token di sessione da utilizzare per l'importazione di un file Amazon S3. L'impostazione predefinita èNULL. Se si fornisce un `session_token` opzionale, è possibile utilizzare credenziali temporanee.

Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3

È possibile eseguire query sui dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL ed esportarli direttamente in file memorizzati in un bucket Amazon S3. A questo scopo, installa innanzitutto l'estensione Aurora PostgreSQL `aws_s3`. Questa estensione fornisce le funzioni utilizzate per esportare i risultati delle quey in Amazon S3. Di seguito, sono disponibili informazioni su come installare l'estensione ed esportare i dati in Amazon S3.

È possibile esportare da un sistema predisposto o un Aurora Serverless v2 Istanza DB. Questi passaggi non sono supportati per Aurora Serverless v1.

> **Note**
>
> L'esportazione tra account in Amazon S3 non è supportata.

Tutte le versioni attualmente disponibili di Aurora PostgreSQL supportano l'esportazione dei dati in Servizio di archiviazione semplice Amazon. Per informazioni dettagliate sulla versione, consulta gli aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

Se non disponi di un bucket configurato per l'esportazione, consulta i seguenti argomenti nella Guida per l'utente di Servizio di archiviazione semplice Amazon.

- Configurazione di Amazon S3
- Creare un bucket

Per impostazione predefinita, i dati esportati da Aurora PostgreSQL ad Amazon S3 utilizzano la crittografia lato server con. Chiave gestita da AWS In alternativa, puoi utilizzare la chiave gestita dal cliente che hai già creato. Se utilizzi la crittografia a bucket, il bucket Amazon S3 deve essere crittografato AWS Key Management Service con la chiave AWS KMS() (SSE-KMS). Attualmente, i bucket crittografati con chiavi gestite di Amazon S3 (SSE-S3) non sono supportati.

> **Note**
>
> Puoi salvare i dati degli snapshot del database e del cluster DB su Amazon S3 utilizzando AWS CLI, o AWS Management Console l'API Amazon RDS. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot del cluster di database in Amazon S3.

Argomenti

- Installazione dell'estensione aws_s3
- Panoramica dell'esportazione di dati in Amazon S3
- Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione
- Configurazione dell'accesso a un bucket Amazon S3
- Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3
- Informazioni di riferimento sulle funzioni
- Risoluzione dei problemi di accesso a Amazon S3

Installazione dell'estensione aws_s3

Prima di poter utilizzare Servizio di archiviazione semplice Amazon con il cluster database Aurora PostgreSQL, è necessario installare l'estensione `aws_s3`. Questa estensione fornisce funzioni per l'esportazione di dati dall'istanza di scrittura di un cluster database Aurora PostgreSQL in un bucket Amazon S3. Fornisce inoltre funzioni per l'importazione dei dati da Amazon S3. Per ulteriori informazioni, consulta Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL. L'estensione `aws_s3` dipende da alcune delle funzioni helper nell'estensione `aws_commons`, che vengono installate automaticamente quando necessario.

Per installare l'estensione **aws_s3**

1. Usa psql (o pgAdmin) per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL come un utente che dispone di privilegi `rds_superuser`. Se hai mantenuto il nome predefinito durante il processo di configurazione, esegui la connessione come `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Per installare l'estensione, esegui il comando seguente.

```
postgres=> CREATE EXTENSION aws_s3 CASCADE;
NOTICE: installing required extension "aws_commons"
CREATE EXTENSION
```

3. Per verificare che l'estensione sia installata, puoi usare il metacomando psql `\dx`.

```
postgres=> \dx
       List of installed extensions
    Name     | Version |   Schema   |                 Description
-------------+---------+------------+---------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Le funzioni per importare dati da Amazon S3 ed esportare dati in Amazon S3 sono ora disponibili per l'uso.

Verifica che la versione di Aurora PostgreSQL supporti le esportazioni in Amazon S3

Per verificare che la versione di Aurora PostgreSQL supporti l'esportazione in Amazon S3, puoi utilizzare il comando `describe-db-engine-versions`. L'esempio seguente verifica se la versione 10.14 può eseguire l'esportazione in Amazon S3.

```
aws rds describe-db-engine-versions --region us-east-1 \
--engine aurora-postgresql --engine-version 10.14 | grep s3Export
```

Se l'output include la stringa `"s3Export"`, allora il motore supporta le esportazioni Amazon S3. In caso contrario, il motore non le supporta.

Panoramica dell'esportazione di dati in Amazon S3

Per esportare i dati archiviati in un database Aurora PostgreSQL verso un bucket Amazon S3, attenersi alla procedura descritta di seguito.

Per esportare i dati Aurora PostgreSQL in S3.

1. Identifica un percorso del file Amazon S3 da utilizzare per l'esportazione dei dati. Per informazioni dettagliate su questo processo, consulta Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione.
2. Fornisci l'autorizzazione ad accedere al bucket Amazon S3.

   Per esportare i dati in un file Amazon S3, concedi al cluster di database Aurora PostgreSQL l'autorizzazione per accedere al bucket Amazon S3 che verrà utilizzato dall'esportazione per lo storage. Questa operazione include le seguenti fasi:

   1. Crea una policy IAM che fornisce l'accesso a un bucket Amazon S3 in cui desideri eseguire l'esportazione.
   2. Creare un ruolo IAM.
   3. Collega la policy creata al ruolo creato.
   4. Aggiungi questo ruolo IAM al cluster di database .

   Per informazioni dettagliate su questo processo, consulta Configurazione dell'accesso a un bucket Amazon S3.
3. Identifica una query del database per ottenere i dati. Esporta i dati della query chiamando la funzione `aws_s3.query_export_to_s3`.

   Dopo aver completato le attività di preparazione precedenti, utilizza la funzione aws_s3.query_export_to_s3 per esportare i risultati della query in Amazon S3. Per informazioni dettagliate su questo processo, consulta Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione

Specifica le seguenti informazioni per identificare la posizione in Amazon S3 in cui desideri esportare i dati:

- Nome del bucket – Un bucket è un container di oggetti o file Amazon S3.

  Per ulteriori informazioni sull'archiviazione dei dati con Amazon S3, consulta Creating a bucket e Working with objects nella Amazon Simple Storage Service User Guide.

- Percorso del file – Il percorso del file identifica la posizione di archiviazione dell'esportazione nel bucket Amazon S3. Il percorso del file comprende:
  - Un prefisso del percorso facoltativo che identifica un percorso di cartella virtuale.
  - Un prefisso del file che identifica uno o più file da archiviare. Esportazioni di dimensioni maggiori vengono archiviate in più file, ciascuno con una dimensione massima di circa 6 GB. I nomi di file aggiuntivi hanno lo stesso prefisso di file ma con l'aggiunta di `_part`*XX*. *XX* rappresenta 2, poi 3 e così via.

  Ad esempio, un percorso del file con una cartella `exports` e un prefisso del file `query-1-export` è `/exports/query-1-export`.

- AWS Regione (opzionale): la AWS regione in cui si trova il bucket Amazon S3.

  > **Note**

  Per un elenco dei nomi delle AWS regioni e dei valori associati, vedereRegioni e zone di disponibilità.

Per conservare le informazioni sul file Amazon S3 relative alla posizione di archiviazione dell'esportazione, puoi utilizzare la funzione aws_commons.create_s3_uri per creare una struttura composita `aws_commons._s3_uri_1` come descritto di seguito.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

In seguito fornisci questo valore `s3_uri_1` come un parametro nella chiamata alla funzione aws_s3.query_export_to_s3. Per alcuni esempi, consulta Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

Configurazione dell'accesso a un bucket Amazon S3

Per esportare i dati in Amazon S3, concedi all' del cluster di database PostgreSQL l'autorizzazione per accedere al bucket Amazon S3 di destinazione dei file.

A tale scopo, procedi come indicato di seguito.

Per concedere a un cluster di database PostgreSQL l'accesso ad Amazon S3 tramite un ruolo IAM

1. Creare una policy IAM

   Questa policy fornisce le autorizzazioni bucket e di oggetto che consentono all' del cluster di database PostgreSQL di accedere a Amazon S3.

   Come parte della creazione di questa policy, attenersi alla seguente procedura:

   a. Includere nella policy le seguenti operazioni obbligatorie per consentire il trasferimento dei file dall' del cluster del database PostgreSQL a un bucket Amazon S3:

      - `s3:PutObject`
      - `s3:AbortMultipartUpload`

   b. Includere l'Amazon Resource Name (ARN) che identifica il bucket Amazon S3 e gli oggetti nel bucket. Il formato ARN per l'accesso a Amazon S3 è: `arn:aws:s3:::`*`amzn-s3-demo-bucket`*`/*`

   Per ulteriori informazioni sulla creazione di una policy IAM per Aurora PostgreSQL, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. Consulta anche il Tutorial: Creare e collegare la prima policy gestita dal cliente nella Guida per l'utente di IAM.

   Il AWS CLI comando seguente crea una policy IAM denominata `rds-s3-export-policy` con queste opzioni. Concede l'accesso a un bucket denominato *`amzn-s3-demo-bucket`*.

   > **Warning**
   >
   > Si consiglia di impostare il database all'interno di un VPC privato con policy di endpoint configurate per accedere a bucket specifici. Per ulteriori informazioni, consulta Utilizzo delle policy dell'endpoint per Amazon S3 nella Guida per l'utente di Amazon VPC.
   > Si consiglia di non creare una policy con accesso a tutte le risorse. Questo accesso può rappresentare una minaccia per la sicurezza dei dati. Se si crea una policy che consente

a `S3:PutObject` di accedere a tutte le risorse utilizzando `"Resource":"*"`, un utente con privilegi di esportazione può esportare i dati in tutti i bucket dell'account. Inoltre, l'utente può esportare i dati in qualsiasi bucket pubblicamente scrivibile all'interno della regione AWS .

Dopo aver creato la policy, annotarne l'Amazon Resource Name (ARN). Per la fase successiva, in cui si associa la policy a un ruolo IAM, è necessario l'ARN.

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
         "Action": [
           "s3:PutObject*",
           "s3:ListBucket",
           "s3:GetObject*",
           "s3:DeleteObject*",
           "s3:GetBucketLocation",
           "s3:AbortMultipartUpload"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

2. Creare un ruolo IAM.

   In questo modo, Aurora PostgreSQL può assumere questo ruolo IAM per tuo conto, per accedere ai bucket Amazon S3. Per ulteriori informazioni, consulta la pagina relativa alla creazione di un ruolo per delegare le autorizzazioni a un utente IAM nella Guida per l'utente IAM.

   Si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy basate sulle risorse per limitare le autorizzazioni del servizio a una risorsa specifica. Questo è il modo più efficace per proteggersi dal problema di deputy confused.

Se si utilizzano entrambe le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID account, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account quando viene utilizzato nella stessa dichiarazione di policy.

- Utilizzare `aws:SourceArn` se si desidera un accesso cross-service per una singola risorsa.
- Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso cross-service.

Nella policy, assicurarsi di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. L'esempio seguente mostra come eseguire questa operazione utilizzando il AWS CLI comando per creare un ruolo denominato`rds-s3-export-role`.

Example

In Linux, macOS, oppure Unix:

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

In Windows:

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

3. Collegare la policy IAM al ruolo IAM creato.

   Il AWS CLI comando seguente associa la policy creata in precedenza al ruolo denominato `rds-s3-export-role`. Replace *`your-policy-arn`* con l'ARN della policy annotato in un passaggio precedente.

   ```
   aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
   export-role
   ```

4. Aggiungere il ruolo IAM al cluster. A tale scopo, utilizzare AWS Management Console o AWS CLI, come descritto di seguito.

Console

Per aggiungere un ruolo IAM al cluster di database PostgreSQL tramite la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere il nome del cluster di database PostgreSQL per visualizzarne i dettagli.
3. Nella scheda Connectivity & security (Connettività e sicurezza), nella sezione Manage IAM roles (Gestisci ruoli IAM), selezionare il ruolo da aggiungere sotto Add IAM roles to this instance (Aggiungi ruoli IAM a questa istanza).
4. In Feature (Caratteristica), scegliere s3Export.
5. Scegliere Add role (Aggiungi ruolo).

AWS CLI

Per aggiungere un ruolo IAM a un cluster di database PostgreSQL tramite CLI

- Utilizzare il seguente comando per aggiungere il ruolo al cluster di database PostgreSQL denominato `my-db-cluster`. Sostituire *`your-role-arn`* con l'ARN del ruolo annotato in precedenza. Utilizzare `s3Export` come valore dell'opzione `--feature-name`.

  Example

  In Linux, macOS, oppure Unix:

```
aws rds add-role-to-db-cluster \
   --db-cluster-identifier my-db-cluster \
   --feature-name s3Export \
   --role-arn your-role-arn   \
   --region your-region
```

  In Windows:

```
aws rds add-role-to-db-cluster ^
   --db-cluster-identifier my-db-cluster ^
   --feature-name s3Export ^
   --role-arn your-role-arn ^
   --region your-region
```

Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3

Esporta i dati PostgreSQL in Amazon S3 chiamando la funzione aws_s3.query_export_to_s3.

Argomenti

- Prerequisiti
- Chiamare aws_s3.query_export_to_s3
- Esportazione in un file CSV che utilizza un delimitatore personalizzato
- Esportazione in un file binario con codifica

Prerequisiti

Prima di utilizzare la funzione `aws_s3.query_export_to_s3`, assicurati di completare i seguenti prerequisiti:

- Installa le estensioni PostgreSQL richieste come descritto in Panoramica dell'esportazione di dati in Amazon S3.
- Determina dove esportare i dati in Amazon S3 come descritto in Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione.
- Assicurati che il cluster di database abbia accesso di esportazione a Amazon S3 come descritto in Configurazione dell'accesso a un bucket Amazon S3.

Gli esempi seguenti utilizzano una tabella del database denominata `sample_table`. Questi esempi esportano i dati in un bucket denominato *`amzn-s3-demo-bucket`*. La tabella e i dati di esempio vengono creati con le seguenti istruzioni SQL in psql.

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

Chiamare aws_s3.query_export_to_s3

Di seguito vengono illustrati le modalità di base per chiamare la funzione aws_s3.query_export_to_s3.

In questi esempi viene utilizzata la variabile `s3_uri_1` per identificare una struttura contenente le informazioni che identificano il file Amazon S3. Utilizzare la funzione aws_commons.create_s3_uri per creare la struttura.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Anche se i parametri variano per le due chiamate di funzione `aws_s3.query_export_to_s3` seguenti, i risultati sono gli stessi per questi esempi. Tutte le righe della tabella `sample_table` vengono esportate in un bucket denominato *amzn-s3-demo-bucket*.

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');

psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

I parametri sono descritti come segue:

- `'SELECT * FROM sample_table'` – Il primo parametro è una stringa di testo obbligatoria contenente una query SQL. Il motore PostgreSQL esegue questa query. I risultati della query vengono copiati nel bucket S3 identificato in altri parametri.
- `:'s3_uri_1'` – Questo parametro è una struttura che identifica il file Amazon S3. In questo esempio viene utilizzata una variabile per identificare la struttura creata in precedenza. È invece possibile creare la struttura includendo la chiamata di funzione `aws_commons.create_s3_uri` in linea all'interno della chiamata di funzione `aws_s3.query_export_to_s3` come segue.

  ```
  SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
     aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
  );
  ```

- `options :='format text'` – Il parametro `options` è una stringa di testo opzionale contenente argomenti `COPY` PostgreSQL. La procedura di copia utilizza gli argomenti e il formato del comando COPY di PostgreSQL.

Se il file specificato non esiste nel bucket Amazon S3, viene creato. Se il file esiste già, viene sovrascritto. La sintassi per accedere ai dati esportati Amazon S3 è la seguente.

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

Esportazioni di dimensioni maggiori vengono archiviate in più file, ciascuno con una dimensione massima di circa 6 GB. I nomi di file aggiuntivi hanno lo stesso prefisso di file ma con l'aggiunta di `_part`*XX*. *XX* rappresenta 2, poi 3 e così via. Ad esempio, supponi di specificare il percorso in cui archivi i file di dati come segue.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

Se l'esportazione deve creare tre file di dati, il bucket Amazon S3 contiene i seguenti file di dati.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

Per il riferimento completo per questa funzione e altri modi per chiamarla, consulta aws_s3.query_export_to_s3. Per ulteriori informazioni sull'accesso ai file in Amazon S3, consulta Visualizzazione di un oggetto nella Guida per l'utente di Amazon Simple Storage Service.

Esportazione in un file CSV che utilizza un delimitatore personalizzato

Nell'esempio seguente viene illustrato come chiamare la funzione aws_s3.query_export_to_s3 per esportare i dati in un file che utilizza un delimitatore personalizzato. Nell'esempio vengono utilizzati gli argomenti del comando PostgreSQL COPY per specificare il formato CSV (valori delimitati da virgole) e un delimitatore : (due punti).

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

Esportazione in un file binario con codifica

Nell'esempio seguente viene illustrato come chiamare la funzione aws_s3.query_export_to_s3 per esportare i dati in un file binario con codifica Windows-1253.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

Informazioni di riferimento sulle funzioni

Funzioni

- aws_s3.query_export_to_s3

- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

Esporta un risultato della query PostgreSQL in un bucket Amazon S3. L'estensione `aws_s3` fornisce la funzione `aws_s3.query_export_to_s3`.

I due parametri richiesti sono `query` e `s3_info`. Questi definiscono la query da esportare e identificano il bucket Amazon S3 in cui eseguire l'esportazione. Un parametro opzionale chiamato `options` fornisce la definizione di vari parametri di esportazione. Per esempi di utilizzo della funzione `aws_s3.query_export_to_s3`, consulta Esportazione dei dati della query utilizzando la funzione aws_s3.query_export_to_s3.

Sintassi

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
)
```

Parametri di input

query

Una stringa di testo obbligatoria contenente una query SQL eseguita dal motore PostgreSQL. I risultati di questa query vengono copiati in un bucket S3 identificato nel parametro `s3_info`.

s3_info

Un tipo composito `aws_commons._s3_uri_1` contenente le seguenti informazioni sull'oggetto S3:

- `bucket` – Il nome del bucket Amazon S3 per contenere il file.
- `file_path` – Il nome e il percorso del file Amazon S3.
- `region`— La AWS regione in cui si trova il bucket. Per un elenco dei nomi delle AWS regioni e dei valori associati, vedereRegioni e zone di disponibilità.

  Attualmente, questo valore deve essere la stessa AWS regione dell' che esporta. L'impostazione predefinita è la AWS regione dell'istanza DB del cluster che esporta.

Per creare una struttura composita `aws_commons._s3_uri_1`, consulta la funzione aws_commons.create_s3_uri.

options

Una stringa di testo opzionale contenente gli argomenti del comando `COPY` di PostgreSQL. Questi argomenti specificano come i dati devono essere copiati quando vengono esportati. Per maggiori dettagli, consulta la documentazione di COPY PostgreSQL.

testo kms_key

Una stringa di testo opzionale contenente la chiave KMS gestita dal cliente del bucket S3 in cui esportare i dati.

Parametri di input alternativi

Per facilitare il testing, puoi utilizzare un set esteso di parametri al posto del parametro `s3_info`. Di seguito vengono riportate le variazioni di sintassi aggiuntive per la funzione `aws_s3.query_export_to_s3`.

Invece di utilizzare il parametro `s3_info` per identificare un file Amazon S3, utilizzare la combinazione dei parametri `bucket`, `file_path` e `region`.

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

query

Una stringa di testo obbligatoria contenente una query SQL eseguita dal motore PostgreSQL. I risultati di questa query vengono copiati in un bucket S3 identificato nel parametro `s3_info`.

bucket

Una stringa di testo obbligatoria contenente il nome del bucket Amazon S3 che contiene il file

file_path

Una stringa di testo obbligatoria contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo opzionale contenente la AWS regione in cui si trova il bucket. Per un elenco dei nomi delle AWS regioni e dei valori associati, vedereRegioni e zone di disponibilità.

Attualmente, questo valore deve essere la stessa AWS regione dell' che esporta. L'impostazione predefinita è la AWS regione dell'istanza DB del cluster che esporta.

options

Una stringa di testo opzionale contenente gli argomenti del comando `COPY` di PostgreSQL. Questi argomenti specificano come i dati devono essere copiati quando vengono esportati. Per maggiori dettagli, consulta la documentazione di COPY PostgreSQL.

testo kms_key

Una stringa di testo opzionale contenente la chiave KMS gestita dal cliente del bucket S3 in cui esportare i dati.

Parametri di output

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

Il numero di righe della tabella che sono state caricate correttamente in Amazon S3 per la query specificata.

files_uploaded

Il numero di file caricati in Amazon S3. I file vengono creati in dimensioni di circa 6 GB. A ogni file aggiuntivo creato è stato aggiunto `_part`*XX* al nome. *XX* rappresenta 2, poi 3 e così via, se necessario.

bytes_uploaded

Il numero totale di byte caricati in Amazon S3.

Esempi

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Crea una struttura `aws_commons._s3_uri_1` per conservare le informazioni relative al file Amazon S3. I risultati della funzione `aws_commons.create_s3_uri` vengono utilizzati nel parametro `s3_info` della funzione aws_s3.query_export_to_s3. Per un esempio di utilizzo della funzione `aws_commons.create_s3_uri`, consulta Specifica del percorso del file Amazon S3 in cui eseguire l'esportazione.

Sintassi

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Parametri di input

bucket

Una stringa di testo obbligatoria contenente il nome del bucket Amazon S3 del file.

file_path

Una stringa di testo obbligatoria contenente il nome file di Amazon S3, incluso il percorso.

Regione

Una stringa di testo obbligatoria contenente la AWS regione in cui si trova il file. Per un elenco dei nomi delle AWS regioni e dei valori associati, vedereRegioni e zone di disponibilità.

Risoluzione dei problemi di accesso a Amazon S3

Se si verificano problemi di connessione durante il tentativo di esportare i dati in Amazon S3, conferma innanzi tutto che le regole di accesso in uscita per il gruppo di sicurezza VPC associato all'istanza DB consentano la connettività di rete. In particolare, il gruppo di sicurezza deve disporre di una regola che consenta all'istanza DB di inviare traffico TCP alla porta 443 e a qualsiasi IPv4 indirizzo (0.0.0.0/0). Per ulteriori informazioni, consulta Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza.

Consulta anche quanto segue per i suggerimenti:

- Risoluzione dei problemi di identità e accesso in Amazon Aurora
- Risoluzione dei problemi di Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service
- Risoluzione dei problemi di Amazon S3 e IAM nella Guida per l'utente di IAM.

AWS Lambda è un servizio di elaborazione basato sugli eventi che consente di eseguire codice senza effettuare il provisioning o la gestione di server. Ad esempio, è possibile utilizzare le funzioni Lambda per elaborare le notifiche di eventi da un database o per caricare dati da file ogni volta che un nuovo file viene caricato su Amazon S3. Per ulteriori informazioni su Lambda, vedi Cos'è? AWS Lambda nella Guida per gli AWS Lambda sviluppatori.

> **Note**
>
> Le AWS Lambda funzioni di richiamo sono supportate in Aurora PostgreSQL 11.9 e versioni successive (tra cui Aurora Serverless v2).

Di seguito sono riportati i riepiloghi dei passaggi necessari.

Per ulteriori informazioni sulle funzioni Lambda, consulta Nozioni di base su Lambda e Fondamenti di AWS Lambda nella Guida per gli sviluppatori di AWS Lambda .

Argomenti

- Fase 1: configurare l'istanza DB del cluster Aurora PostgreSQL DB per PostgreSQL per le connessioni in uscita a AWS Lambda
- AWS Lambda

- Fase 3: installazione dell'estensione aws_lambda per un cluster database Aurora PostgreSQL
- Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo)
- Fase 5: richiamo di una funzione Lambda dal cluster database Aurora PostgreSQL
- Fase 6: concessione delle autorizzazioni ad altri utenti per richiamare le funzioni Lambda
-
- Messaggi di errore della funzione Lambda
- AWS Lambdariferimento a funzioni e parametri

Fase 1: configurare l'istanza DB del cluster Aurora PostgreSQL DB per PostgreSQL per le connessioni in uscita a AWS Lambda

Le funzioni Lambda vengono sempre eseguite all'interno di un Amazon VPC di proprietà del servizio. AWS Lambda Lambda applica le regole di accesso alla rete e di sicurezza a questo VPC e mantiene e monitora il VPC automaticamente. Il cluster database Aurora PostgreSQL deve inviare traffico di rete al VPC del servizio Lambda. La modalità di configurazione dipende dal fatto che l'istanza database principale del cluster database Aurora sia pubblica o privata.

- Cluster database pubblico Aurora PostgreSQL RDS per PostgreSQL: l'istanza DB è pubblica se si trova in una sottorete pubblica sul VPC e se la proprietà "" dell' è. PubliclyAccessible `true` Per trovare il valore di questa proprietà, puoi usare il comando. describe-db-instances AWS CLI In alternativa, puoi utilizzare la AWS Management Console per aprire la scheda Connectivity & security (Connettività e sicurezza) e controllare che l'opzione Publicly accessible (Accessibile pubblicamente) sia impostata su Yes (Sì). Per verificare se l'istanza si trova nella sottorete pubblica del VPC, puoi utilizzare la AWS Management Console o la AWS CLI.

  Per configurare l'accesso a Lambda, usi AWS Management Console o AWS CLI per creare una regola in uscita sul gruppo di sicurezza del tuo VPC. La regola in uscita specifica che TCP può utilizzare la porta 443 per inviare pacchetti a qualsiasi indirizzo (0.0.0.0/0). IPv4
- Istanza privata del cluster Aurora PostgreSQL DB RDS per PostgreSQL si trova in una sottorete privata. PubliclyAccessible `false` Per consentire all'istanza di funzionare con Lambda, è possibile utilizzare un gateway Network Address Translation (NAT). Per ulteriori informazioni, consulta Gateway NAT. In alternativa, puoi configurare il VPC con un endpoint VPC per Lamda. Per ulteriori informazioni, consultare Endpoint VPC nella Guida per l'utente di Amazon VPC. L'endpoint risponde alle chiamate effettuate dal cluster database Aurora PostgreSQL alle funzioni Lambda.

Il tuo VPC può ora interagire con il AWS Lambda VPC a livello di rete. Configura quindi le autorizzazioni utilizzando IAM.

AWS Lambda

Il richiamo di funzioni Lambda dal cluster database Aurora PostgreSQL richiede determinati privilegi. Per configurare i privilegi necessari, si consiglia di creare una policy IAM che consenta di richiamare le funzioni Lambda, assegnare tale policy a un ruolo e quindi applicare il ruolo al cluster database. Questo approccio fornisce al cluster database privilegi per richiamare la funzione Lambda specificata per tuo conto. La procedura seguente mostra come eseguire questa operazione in AWS CLI.

Configurare le autorizzazioni IAM per l'utilizzo del cluster con Lambda

1. Usa il AWS CLI comando create-policy per creare una policy IAM che consenta all'istanza database Aurora PostgreSQL DB del cluster RDS per di richiamare la funzione Lambda specificata. (L'ID dichiarazione (Sid) è una descrizione facoltativa per la dichiarazione di policy e non ha alcun effetto sull'utilizzo). Questa policy fornisce al cluster database Aurora le autorizzazioni minime necessarie per richiamare la funzione Lambda specificata.

   ```
   aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
       "Version": "2012-10-17",
       "Statement": [
           {
           "Sid": "AllowAccessToExampleFunction",
           "Effect": "Allow",
           "Action": "lambda:InvokeFunction",
           "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
           }
       ]
   }'
   ```

   In alternativa, puoi utilizzare la policy `AWSLambdaRole` predefinita che ti consente di richiamare una qualsiasi delle tue funzioni Lambda. Per ulteriori informazioni, consulta Policy IAM basate sull'identità per Lambda

2. Utilizza il comando create-role per creare un ruolo IAM che la policy può assumere in fase di esecuzione AWS CLI .

   ```
   aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
       "Version": "2012-10-17",
       "Statement": [
   ```

```
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Applica la policy al ruolo utilizzando il attach-role-policy AWS CLI comando.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. https://awscli.amazonaws.com/v2/documentation/api/latest/reference/rds/add-role-to-db-cluster.html AWS CLI Quest'ultimo passaggio consente agli utenti del cluster database di richiamare le funzioni Lambda.

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Con le configurazioni di VPC e IAM completate, puoi ora installare l'estensione `aws_lambda`. (Puoi installare l'estensione in qualsiasi momento, ma fino a quando non configuri il supporto VPC e i privilegi IAM corretti, l'estensione `aws_lambda`non aggiunge nulla alle funzionalità del cluster database Aurora PostgreSQL).

Fase 3: installazione dell'estensione **`aws_lambda`** per un cluster database Aurora PostgreSQL

Questa estensione fornisce al cluster database Aurora PostgreSQL la possibilità di chiamare le funzioni Lambda da PostgreSQL.

Installare l'estensione **`aws_lambda`**nel cluster database Aurora PostgreSQL

Utilizza la riga di comando PostgreSQL `psql` o lo strumento pgAdmin per connetterti al cluster database Aurora PostgreSQL.

1. Connettiti al cluster database Aurora PostgreSQL come utente con privilegi `rds_superuser`. L'utente `postgres` predefinito viene visualizzato nell'esempio.

```
psql -h cluster-instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p
 5432
```

2. Installa l'estensione `aws_lambda`. Anche l'estensione `aws_commons` è obbligatoria. Fornisce funzioni di supporto ad `aws_lambda` e molte altre estensioni Aurora per PostgreSQL. Se non si trova già nel tuo cluster database Aurora PostgreSQL , viene installata con `aws_lambda` come mostrato di seguito.

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

L'estensione `aws_lambda` è installata nell'istanza database principale del cluster database Aurora PostgreSQL. Ora puoi creare strutture utili per richiamare le tue funzioni Lambda.

Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo)

Puoi utilizzare le funzioni di supporto nell'estensione `aws_commons` per preparare entità che è possibile richiamare più facilmente da PostgreSQL. Per farlo, devi disporre delle seguenti informazioni sulle funzioni Lambda:

- Nome funzione: il nome, l'Amazon Resource Name (ARN), la versione o l'alias della funzione Lambda. La policy IAM creata in Fase 2: configurazione di IAM per il cluster e Lambda richiede l'ARN, quindi ti consigliamo di utilizzare l'ARN della tua funzione.
- AWS Regione

Per mantenere le informazioni sul nome della funzione Lambda, puoi utilizzare la funzione aws_commons.create_lambda_function_arn. Questa funzione di supporto crea una struttura composita `aws_commons._lambda_function_arn_1` con i dettagli necessari alla funzione di richiamo. Di seguito, puoi trovare tre approcci alternativi per l'impostazione di questa struttura composita.

```
SELECT aws_commons.create_lambda_function_arn(
```

```
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

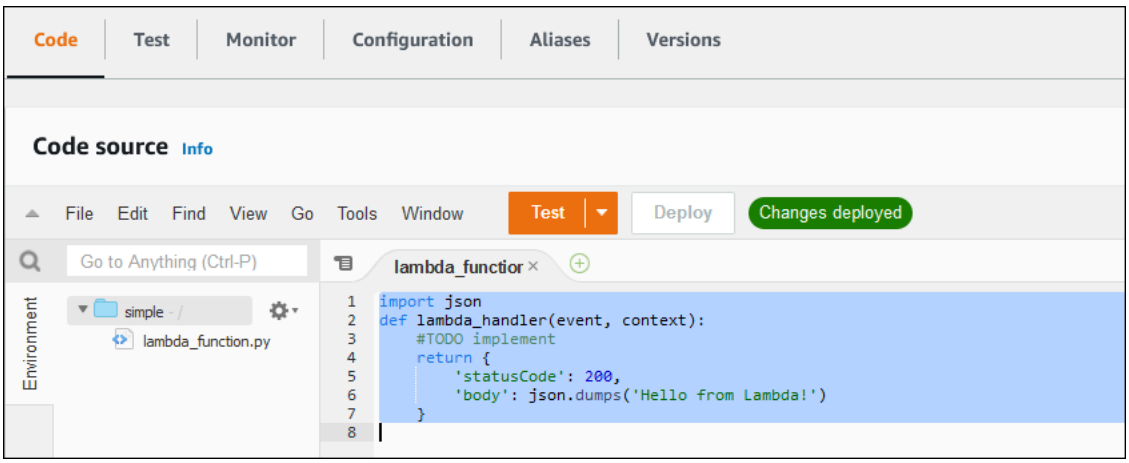
Ognuno di questi valori può essere utilizzato nelle chiamate alla funzione aws_lambda.invoke. Per alcuni esempi, consulta Fase 5: richiamo di una funzione Lambda dal cluster database Aurora PostgreSQL.

Fase 5: richiamo di una funzione Lambda dal cluster database Aurora PostgreSQL

La funzione `aws_lambda.invoke` si comporta in modo sincrono o asincrono, a seconda del `invocation_type`. Le due alternative per questo parametro sono `RequestResponse` (il valore predefinito) e `Event`, come di seguito riportato.

- **`RequestResponse`**: questo tipo di richiamo è sincrono. Questo è il comportamento predefinito quando viene effettuata la chiamata senza specificare un tipo di chiamata. Il payload di risposta include i risultati della funzione `aws_lambda.invoke`. Utilizza questo tipo di chiamata quando il flusso di lavoro richiede la ricezione dei risultati della funzione Lambda prima di procedere.
- **`Event`**: questo tipo di richiamo è asincrono. La risposta non include un payload contenente i risultati. Utilizza questo tipo di richiamo quando il flusso di lavoro non ha bisogno di un risultato della funzione Lambda per continuare l'elaborazione.

Come semplice test della configurazione, puoi connetterti all'istanza database utilizzando `psql` e richiamare una funzione di esempio dalla riga di comando. Supponiamo di avere una delle funzioni di base impostate sul tuo servizio Lambda, come la semplice funzione Python mostrata nello screenshot di seguito.



Per richiamare una funzione di esempio

1. Connettiti all'istanza database principale utilizzando `psql` o pgAdmin.

   ```
   psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. Richiama la funzione utilizzando il relativo ARN.

   ```
   SELECT * from
    aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
   region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
    Postgres!"}'::json );
   ```

   La risposta è la seguente.

   ```
   status_code |                         payload                         |
    executed_version | log_result
   -------------+---------------------------------------------------------
   +------------------+------------
            200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
        |
   (1 row)
   ```

Se il tuo tentativo di richiamo non ha esito positivo, vedi Messaggi di errore della funzione Lambda .

Fase 6: concessione delle autorizzazioni ad altri utenti per richiamare le funzioni Lambda

A questo punto della procedura, solo tu in qualità di `rds_superuser` puoi richiamare le funzioni Lambda. Per consentire ad altri utenti di richiamare le funzioni che crei, è necessario concedere loro l'autorizzazione.

Per concedere ad altri utenti l'autorizzazione per richiamare le funzioni Lambda

1. Connettiti all'istanza database principale utilizzando `psql` o pgAdmin.

```
psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. Esegui i seguenti comandi SQL:

```
postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
```

Di seguito, è possibile trovare diversi esempi di chiamate alla funzione aws_lambda.invoke. La maggior parte degli esempi utilizza la struttura composita creata per semplificare il `aws_lambda_arn_1` passaggio dei dettagli della funzione. Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo) Per un esempio di chiamata asincrona, consulta Esempio: richiamo (di eventi) asincroni di funzioni Lambda. Tutti gli altri esempi elencati utilizzano il richiamo sincrono.

Per ulteriori informazioni sui tipi di invocazione Lambda, consulta Richiamo delle funzioni Lambda nella AWS Lambda Guida per gli sviluppatori. Per ulteriori informazioni su `aws_lambda_arn_1`, consulta aws_commons.create_lambda_function_arn.

Elenco di esempi

- Esempio: invocazione sincrona (RequestResponse) di funzioni Lambda
- Esempio: richiamo (di eventi) asincroni di funzioni Lambda
- Esempio: acquisizione del registro di esecuzione Lambda in una risposta di funzione
- Esempio: inclusione del contesto client in una funzione Lambda
- Esempio: richiamo di una versione specifica di una funzione Lambda

Esempio: invocazione sincrona (RequestResponse) di funzioni Lambda

Di seguito sono riportati due esempi di una chiamata di funzione Lambda sincrona. I risultati di queste chiamate di funzione `aws_lambda.invoke` sono gli stessi.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

I parametri sono descritti come segue:

- `:'aws_lambda_arn_1'`: questo parametro identifica la struttura composita creata in Fase 4: utilizzo delle funzioni di supporto Lambda con il cluster database Aurora PostgreSQL (facoltativo) con la funzione di supporto di `aws_commons.create_lambda_function_arn`. Puoi anche creare questa struttura in linea all'interno della chiamata `aws_lambda.invoke` come segue:

  ```
  SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
   'aws-region'),
  '{"body": "Hello from Postgres!"}'::json
  );
  ```

- `'{"body": "Hello from PostgreSQL!"}'::json`— Il JSON payload da passare alla funzione Lambda.
- `'RequestResponse'`: il tipo di richiamo Lambda.

Esempio: richiamo (di eventi) asincroni di funzioni Lambda

Di seguito è riportato un esempio di una chiamata di funzione Lambda asincrona. Il tipo di richiamo `Event` pianifica il richiamo della funzione Lambda con il payload di input specificato e restituisce immediatamente un risultato. Utilizza il tipo di chiamata `Event` in determinati flussi di lavoro che non dipendono dai risultati della funzione Lambda.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

Esempio: acquisizione del registro di esecuzione Lambda in una risposta di funzione

È possibile includere gli ultimi 4 KB del registro di esecuzione nella risposta della funzione utilizzando il parametro `log_type` nella chiamata di funzione `aws_lambda.invoke`. Per impostazione predefinita, questo parametro è impostato su `None`, ma puoi specificare `Tail` per acquisire i risultati del registro di esecuzione Lambda nella risposta, come illustrato di seguito.

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Impostare il parametro aws_lambda.invoke della funzione `log_type` su `Tail` per includere il log di esecuzione nella risposta. Il valore predefinito per il parametro `log_type` è `None`.

Il `log_result` che viene restituito è una stringa `base64` codificata. È possibile decodificare il contenuto utilizzando una combinazione delle funzioni e `decode` Postgre`convert_from`. SQL

Per ulteriori informazioni su `log_type`, consulta aws_lambda.invoke.

Esempio: inclusione del contesto client in una funzione Lambda

La funzione `aws_lambda.invoke` ha un parametro `context` che puoi utilizzare per passare le informazioni separate dal payload, come illustrato di seguito.

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Per includere il contesto del client, utilizzate un JSON oggetto per il parametro della aws_lambda.invoke funzione. `context`

Per ulteriori informazioni sul parametro `context`, consulta la documentazione di riferimento di aws_lambda.invoke.

Esempio: richiamo di una versione specifica di una funzione Lambda

Puoi specificare una determinata versione di una funzione Lambda includendo il parametro `qualifier` con la chiamata `aws_lambda.invoke`. Di seguito puoi trovare un esempio in cui viene utilizzato '*`custom_version`*' come alias per la versione.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

Puoi inoltre fornire un qualificatore di funzione Lambda con i dettagli relativi al nome della funzione, come mostrato di seguito.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

Per ulteriori informazioni su `qualifier` e altri parametri, consulta documentazione di riferimento di aws_lambda.invoke.

Messaggi di errore della funzione Lambda

Nell'elenco seguente sono disponibili informazioni sui messaggi di errore, con le possibili cause e soluzioni.

- VPCproblemi di configurazione

  VPCi problemi di configurazione possono generare i seguenti messaggi di errore durante il tentativo di connessione:

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  CONTEXT:  SQL function "invoke" statement 1
  ```

  Una causa comune di questo errore è il gruppo di VPC sicurezza configurato in modo errato. Assicurati di avere una regola in uscita per l'TCPapertura sulla porta 443 del tuo gruppo di VPC sicurezza in modo da poterti connettere alla Lambda. VPC VPC

- Mancanza delle autorizzazioni necessarie per richiamare le funzioni Lambda

  Se viene visualizzato uno dei seguenti messaggi di errore, l'utente (ruolo) che richiama la funzione non dispone delle autorizzazioni appropriate.

  ```
  ERROR:  permission denied for schema aws_lambda
  ```

  ```
  ERROR:  permission denied for function invoke
  ```

A un utente (ruolo) devono essere concesse autorizzazioni specifiche per richiamare le funzioni Lambda. Per ulteriori informazioni, consulta Fase 6: concessione delle autorizzazioni ad altri utenti per richiamare le funzioni Lambda.

- Gestione impropria degli errori nelle funzioni Lambda

  Se una funzione Lambda genera un'eccezione durante l'elaborazione della richiesta, `aws_lambda.invoke` fallisce con un SQL errore Postgre come il seguente.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
ERROR:  lambda invocation failed
DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
 "Unhandled", details: "<Error details string>".
```

  Assicurati di gestire gli errori nelle funzioni Lambda o nell'applicazione SQL Postgre.

AWS Lambdariferimento a funzioni e parametri

Di seguito è riportato il riferimento per le funzioni e i parametri da utilizzare per richiamare Lambda con Aurora PostgreSQL RDS per .

Funzioni e parametri

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- parametri aws_lambda

aws_lambda.invoke

Esegue una funzione Lambda per un cluster DB Aurora PostgreSQL .

Per ulteriori dettagli sul richiamo delle funzioni Lambda, consulta anche Invoke nella Guida per gli sviluppatori di AWS Lambda.

Sintassi

JSON

```
aws_lambda.invoke(
```

```
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
```

```
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

Parametri di input

function_name

Nome identificativo della funzione Lambda. Il valore può essere il nome della funzione, un ARN o un ARN parziale. Per un elenco dei formati possibili, consulta Formati dei nomi delle funzioni Lambda nella Guida per gli sviluppatori di AWS Lambda.

payload

L'input per la funzione Lambda. Il formato può essere JSON o JSONB. Per ulteriori informazioni, consulta Tipi di JSON nella documentazione di PostgreSQL.

region

(Facoltativo) La regione Lambda per la funzione. Per impostazione predefinita, Aurora risolve la regione AWS dall'ARN completo nella `function_name` oppure utilizza la regione dell'istanza database Aurora PostgreSQL . Se il valore di questa Regione è in conflitto con quello fornito nell'ARN `function_name`, viene generato un errore.

invocation_type

Il tipo di chiamata della funzione Lambda. Il valore prevede la distinzione tra lettere maiuscole e minuscole. I valori possibili sono:

- `RequestResponse` – Il valore predefinito. Questo tipo di chiamata per una funzione Lambda è sincrono e restituisce un payload di risposta nel risultato. Utilizzare il tipo di chiamata `RequestResponse` quando il flusso di lavoro dipende dalla ricezione immediata del risultato della funzione Lambda.
- `Event` – Questo tipo di chiamata per una funzione Lambda è asincrono e restituisce immediatamente una risposta senza un payload restituito. Utilizzare il tipo di chiamata `Event`

quando non sono necessari i risultati della funzione Lambda prima che il flusso di lavoro proceda.

- `DryRun` – Questo tipo di chiamata verifica l'accesso senza eseguire la funzione Lambda.

log_type

Il tipo di log Lambda da restituire nel parametro di output `log_result`. Il valore prevede la distinzione tra lettere maiuscole e minuscole. I valori possibili sono:

- Tail – Il parametro di output `log_result` restituito includerà gli ultimi 4 KB del registro di esecuzione.
- None – Non viene restituita nessuna informazione di log Lambda.

context

Contesto client in formato JSON o JSONB. I campi da utilizzare includono `custom` e `env`.

qualifier

Un qualificatore che identifica la versione di una funzione Lambda da richiamare. Se questo valore è in conflitto con quello fornito nell' `function_name` ARN, viene generato un errore.

Parametri di output

status_code

Un codice di risposta allo stato HTTP. Per ulteriori informazioni, consulta Elementi di risposta del richiamo di Lambda nella Guida per gli sviluppatori di AWS Lambda.

payload

Le informazioni restituite dalla funzione Lambda eseguita. Il formato è in JSON o JSONB.

executed_version

La versione della funzione Lambda eseguita.

log_result

Le informazioni del registro di esecuzione restituite se il valore `log_type` è `Tail` quando è stata richiamata la funzione Lambda. Il risultato contiene gli ultimi 4 KB del registro di esecuzione codificato in Base64.

aws_commons.create_lambda_function_arn

Crea una struttura `aws_commons._lambda_function_arn_1` per contenere le informazioni sul nome della funzione Lambda. È possibile utilizzare i risultati della `aws_commons.create_lambda_function_arn` funzione nel parametro `function_name` della funzione aws_lambda.invoke aws_lambda.invoke.

Sintassi

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
    RETURNS aws_commons._lambda_function_arn_1
```

Parametri di input

function_name

Stringa di testo obbligatoria contenente il nome della funzione Lambda. Il valore può essere un nome di funzione, un ARN parziale o un ARN completo.

region

Una stringa di testo facoltativa contenente la regione AWS in cui si trova la funzione Lambda. Per un elenco di nomi di regione e dei valori associati, consulta Regioni e zone di disponibilità.

parametri aws_lambda

In questa tabella, puoi trovare i parametri associati alla funzione. `aws_lambda`

Parametro	Descrizione
`aws_lambda.connect_timeout_ms`	Si tratta di un parametro dinamico che imposta il tempo di attesa massimo durante la connessione a AWS Lambda. I valori predefiniti sono`1000`. I valori consentiti per questo parametro sono compresi tra 1 e 900000.
`aws_lambda.request_timeout_ms`	Si tratta di un parametro dinamico che imposta il tempo di attesa massimo in attesa della risposta da AWS Lambda. I valori

Parametro	Descrizione
	predefiniti sono`3000`. I valori consentiti per questo parametro sono compresi tra 1 e 900000.
`aws_lambda.endpoint_override`	Speciifica l'endpoint che può essere utilizzato per connettersi a AWS Lambda. Una stringa vuota seleziona l'endpoint AWS Lambda predefinito per la regione. È necessario riavviare il database affinché questa modifica statica dei parametri abbia effetto.

Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs CloudWatch

Puoi configurare il tuo cluster Aurora PostgreSQL DB per esportare regolarmente i dati di log in Amazon Logs. CloudWatch In tal caso, gli eventi del log PostgreSQL del cluster PostgreSQL del cluster Aurora PostgreSQL vengono pubblicati automaticamente su Amazon, come Amazon Logs. CloudWatch CloudWatch In CloudWatch, puoi trovare i dati di log esportati in un gruppo di log per il tuo cluster Aurora PostgreSQL DB. Il gruppo di log contiene uno o più flussi di log che contengono gli eventi del registro PostgreSQL di ogni istanza nel cluster.

La pubblicazione dei log su CloudWatch Logs consente di conservare i record di log PostgreSQL del cluster in uno storage altamente durevole. Con i dati di registro disponibili in CloudWatch Logs, puoi valutare e migliorare le operazioni del cluster. Puoi anche utilizzarli CloudWatch per creare allarmi e visualizzare metriche. Per ulteriori informazioni, consulta Monitoraggio degli eventi di registro in Amazon CloudWatch.

> **Note**
>
> La pubblicazione dei log di PostgreSQL su Logs consuma spazio di archiviazione e comporta dei costi CloudWatch per tale archiviazione. Assicurati di eliminare tutti i log che non ti servono più. CloudWatch

La disattivazione dell'opzione di log di esportazione per un cluster Aurora PostgreSQL DB esistente non influisce sui dati già contenuti nei log. CloudWatch I log esistenti rimangono disponibili in Logs in base alle impostazioni di CloudWatch conservazione dei log. Per ulteriori informazioni sui CloudWatch log, consulta What is Amazon CloudWatch Logs?

Aurora PostgreSQL supporta la pubblicazione di log in Logs per le seguenti versioni. CloudWatch

- 16.1 e tutte le versioni successive
- 15.2 o versioni successive alla 15
- 14.3 o versioni successive alla 14
- 13.3 o versioni successive alla 13
- 12.8 e versioni successive alla 12
- 11.9 o versioni successive alla 11

Per informazioni sull'attivazione dell'opzione di pubblicazione dei log su Logs, sul monitoraggio degli eventi di registro in CloudWatch Logs e sull'analisi dei log utilizzando CloudWatch CloudWatch Logs Insights, consulta i seguenti argomenti.

Argomenti

- Attivazione dell'opzione per pubblicare i log su Amazon CloudWatch
- Monitoraggio degli eventi di registro in Amazon CloudWatch
- Analisi dei log PostgreSQL con Logs Insights CloudWatch

Attivazione dell'opzione per pubblicare i log su Amazon CloudWatch

Per pubblicare il log PostgreSQL del cluster Aurora PostgreSQL DB in Logs, scegli l'opzione di esportazione dei log per il cluster. CloudWatch Puoi scegliere l'impostazione Log export (Esportazione log) durante la creazione del cluster database Aurora PostgreSQL. In alternativa, puoi modificare il cluster in un secondo momento. Quando modifichi un cluster esistente, i relativi log PostgreSQL di ogni istanza vengono pubblicati CloudWatch nel cluster da quel momento in poi. Per Aurora PostgreSQL, PostgreSQL log () è l'unico log che viene pubblicato su Amazon. `postgresql.log` CloudWatch

Puoi utilizzare l'API AWS Management Console AWS CLI, the o RDS per attivare la funzionalità di esportazione dei log per il tuo cluster Aurora PostgreSQL DB.

Console

Scegli l'opzione Log exports per iniziare a pubblicare i log PostgreSQL dal cluster Aurora PostgreSQL DB su Logs. CloudWatch

Per attivare la funzionalità Log export (Esportazione log) dalla console

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli il cluster Aurora PostgreSQL DB di cui desideri pubblicare i dati di registro nei registri. CloudWatch
4. Scegliere Modify (Modifica).
5. Nella sezione Log exports (Esportazioni log) scegli Postgresql log (Log PostgreSQL).
6. Scegliere Continue (Continua) e quindi selezionare Modify cluster (Modifica cluster) nella pagina di riepilogo.

AWS CLI

Puoi attivare l'opzione di esportazione dei log per iniziare a pubblicare i log di Aurora PostgreSQL su Amazon Logs con. CloudWatch AWS CLI A tale scopo, esegui il comando con le seguenti opzioni modify-db-cluster AWS CLI :

- `--db-cluster-identifier`— – L'identificatore del cluster DB.
- `--cloudwatch-logs-export-configuration`—L'impostazione di configurazione per i tipi di log da impostare per l'esportazione in CloudWatch Logs for the DB cluster.

Puoi pubblicare i log Aurora PostgreSQL anche emettendo uno dei seguenti comandi della AWS CLI :

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-istantanea
- restore-db-cluster-to-point-in-time

Esegui uno di questi AWS CLI comandi con le seguenti opzioni:

- `--db-cluster-identifier`— – L'identificatore del cluster DB.
- `--engine`— – Il motore di database.
- `--enable-cloudwatch-logs-exports`—L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Potrebbero essere necessarie altre opzioni a seconda del AWS CLI comando eseguito.

Example

Il comando seguente crea un cluster Aurora PostgreSQL DB per pubblicare i file di registro nei registri. CloudWatch

In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --engine aurora-postgresql \
    --enable-cloudwatch-logs-exports postgresql
```

In Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --engine aurora-postgresql ^
    --enable-cloudwatch-logs-exports postgresql
```

Example

Il comando seguente modifica un cluster Aurora PostgreSQL DB esistente per pubblicare i file di registro nei registri. CloudWatch Il valore `--cloudwatch-logs-export-configuration` è un oggetto JSON. La chiave per questo oggetto è`EnableLogTypes`, e il suo valore è, e. `postgresql instance`

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","instance"]}'
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --cloudwatch-logs-export-configuration '{\"EnableLogTypes\":[\"postgresql\",
\"instance\"]}'
```

> **Note**
>
> Quando usi il prompt comandi di Windows, non devi inserire le doppie virgolette (") nel codice JSON precedendole con il backslash (\).

Example

L'esempio seguente modifica un cluster Aurora PostgreSQL DB esistente per disabilitare la pubblicazione dei file di registro in Logs. CloudWatch Il valore `--cloudwatch-logs-export-configuration` è un oggetto JSON. La chiave per questo oggetto è`DisableLogTypes`, e il suo valore è and. `postgresql instance`

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":
["postgresql","instance"]}'
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"postgresql\",
\"instance\"]}"
```

> **Note**
>
> Quando usi il prompt comandi di Windows, non devi inserire le doppie virgolette (") nel codice JSON precedendole con il backslash (\).

API RDS

Puoi attivare l'opzione Log export (Esportazione log) per iniziare a pubblicare i registri di Aurora PostgreSQL con l'API RDS. A tale scopo, eseguite l'DBClusteroperazione Modifica con le seguenti opzioni:

- `DBClusterIdentifier` - L'identificatore del cluster di database.
- `CloudwatchLogsExportConfiguration`— L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Puoi anche pubblicare i log Aurora MySQL con l'API RDS eseguendo una delle seguenti operazioni dell'API RDS:

- CreaDBCluster
- Ripristina DBCluster da S3
- RipristinaDBClusterFromSnapshot
- RipristinaDBClusterToPointInTime

Esegui l'operazione dell'API RDS con i seguenti parametri:

- `DBClusterIdentifier`— – L'identificatore del cluster DB.
- `Engine`— – Il motore di database.
- `EnableCloudwatchLogsExports`—L'impostazione di configurazione per i tipi di log da abilitare per l'esportazione in CloudWatch Logs for the DB cluster.

Potrebbero essere necessari altri parametri a seconda del AWS CLI comando eseguito.

Monitoraggio degli eventi di registro in Amazon CloudWatch

Con gli eventi di SQL registro di Aurora Postgre pubblicati e disponibili come Amazon CloudWatch Logs, puoi visualizzare e monitorare gli eventi utilizzando Amazon. CloudWatch Per ulteriori informazioni sul monitoraggio, consulta Visualizza i dati di registro inviati ai registri. CloudWatch

Quando attivi le esportazioni di log, viene creato automaticamente un nuovo gruppo di log utilizzando il prefisso `/aws/rds/cluster/` con il nome di Aurora SQL Postgre e il tipo di registro, come nello schema seguente.

```
/aws/rds/cluster/your-cluster-name/postgresql
```

Ad esempio, supponiamo che un cluster Aurora SQL Postgre DB `docs-lab-apg-small` denominato esporti il proprio log in Amazon Logs. CloudWatch Il nome del gruppo di log in Amazon CloudWatch è mostrato di seguito.

```
/aws/rds/cluster/docs-lab-apg-small/postgresql
```

Se esiste un gruppo di log con il nome specificato, Aurora utilizza quel gruppo di log per esportare i dati di log per il cluster DB Aurora. Ogni istanza DB nel cluster Aurora Postgre SQL DB carica il proprio log Postgre nel gruppo di log come flusso di SQL log distinto. Puoi esaminare il gruppo di log e i relativi flussi di log utilizzando i vari strumenti grafici e analitici disponibili in Amazon. CloudWatch

Ad esempio, è possibile cercare informazioni all'interno degli eventi di registro del cluster Aurora Postgre SQL DB e filtrare gli eventi utilizzando la CloudWatch console Logs, AWS CLI o i Logs. CloudWatch API Per ulteriori informazioni, consulta Ricerca e filtraggio dei dati di log nella Amazon CloudWatch Logs User Guide.

Per impostazione predefinita, i nuovi gruppi di log vengono creati utilizzando Never expire (Non scade mai) per il periodo di conservazione. Puoi utilizzare la console CloudWatch Logs, il AWS CLI o CloudWatch Logs API per modificare il periodo di conservazione dei log. Per ulteriori informazioni, consulta Change log data retention in CloudWatch Logs nella Amazon CloudWatch Logs User Guide.

> **Tip**
>
> Puoi utilizzare la configurazione automatizzata, ad esempio AWS CloudFormation, per creare gruppi di log con periodi di conservazione dei log predefiniti, filtri metrici e autorizzazioni di accesso.

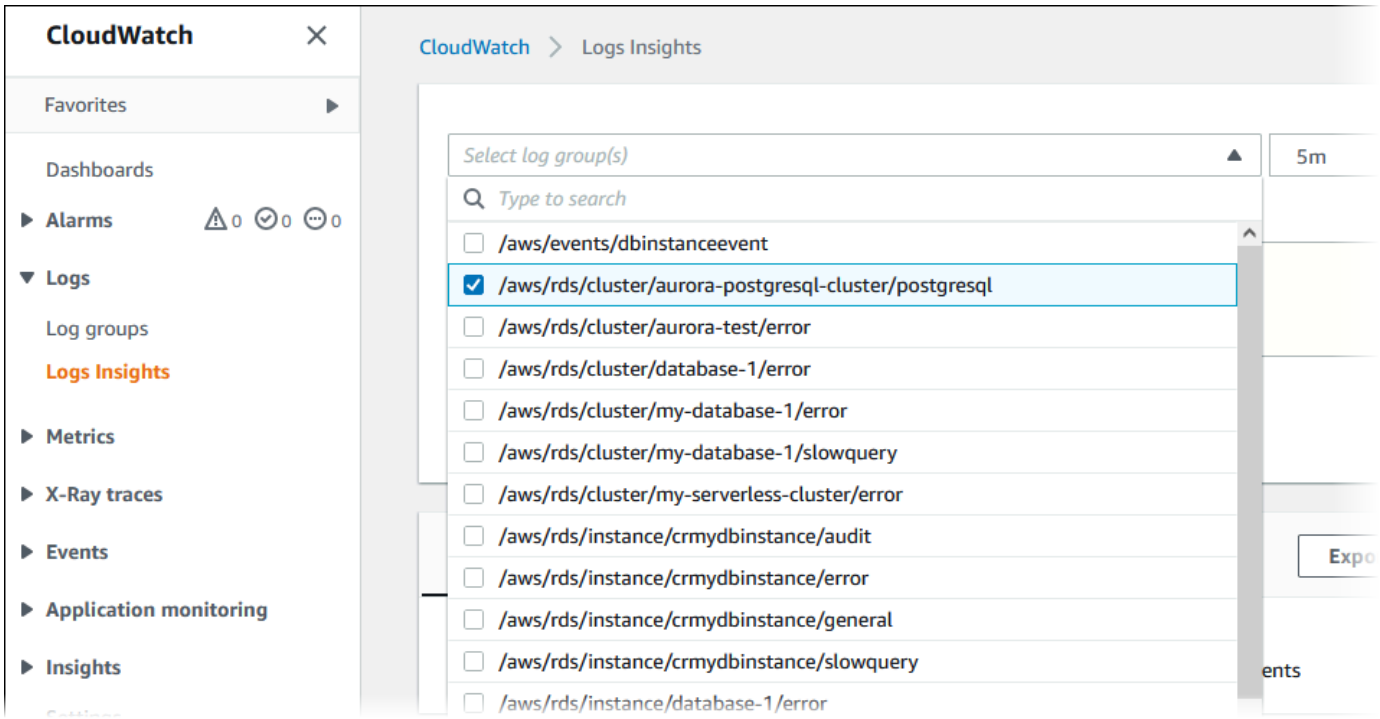
Analisi dei log PostgreSQL con Logs Insights CloudWatch

Con i log PostgreSQL del CloudWatch cluster Aurora PostgreSQL DB pubblicati come Logs, CloudWatch puoi utilizzare Logs Insights per cercare e analizzare in modo interattivo i dati di log in Amazon Logs. CloudWatch CloudWatch Logs Insights include un linguaggio di interrogazione, query di esempio e altri strumenti per l'analisi dei dati di log in modo da poter identificare potenziali problemi e verificare le correzioni. Per ulteriori informazioni, consulta Analyzing log data with CloudWatch Logs Insights nella Amazon CloudWatch Logs User Guide.

Per analizzare i log di PostgreSQL con Logs Insights CloudWatch

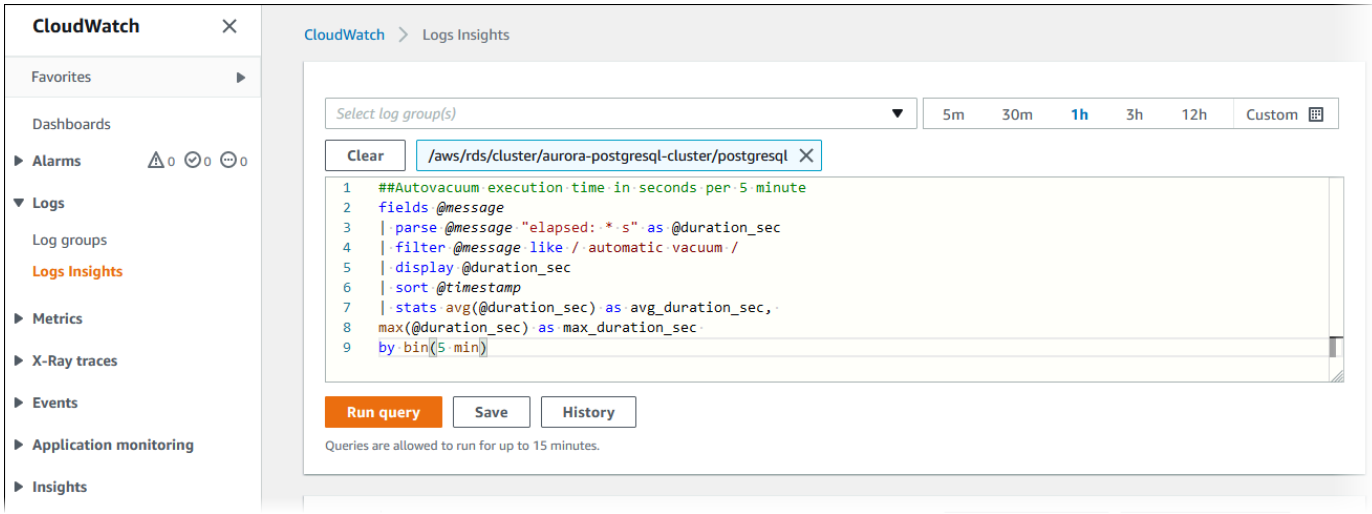
1. Apri la console all'indirizzo. CloudWatch https://console.aws.amazon.com/cloudwatch/
2. Nel pannello di navigazione, aprire Logs (Log), e scegliere Log Insights (Informazioni dettagliate Log.

3. In Select log group(s) (Seleziona uno o più i gruppi di log), seleziona il gruppo di log per il cluster database Aurora PostgreSQL.

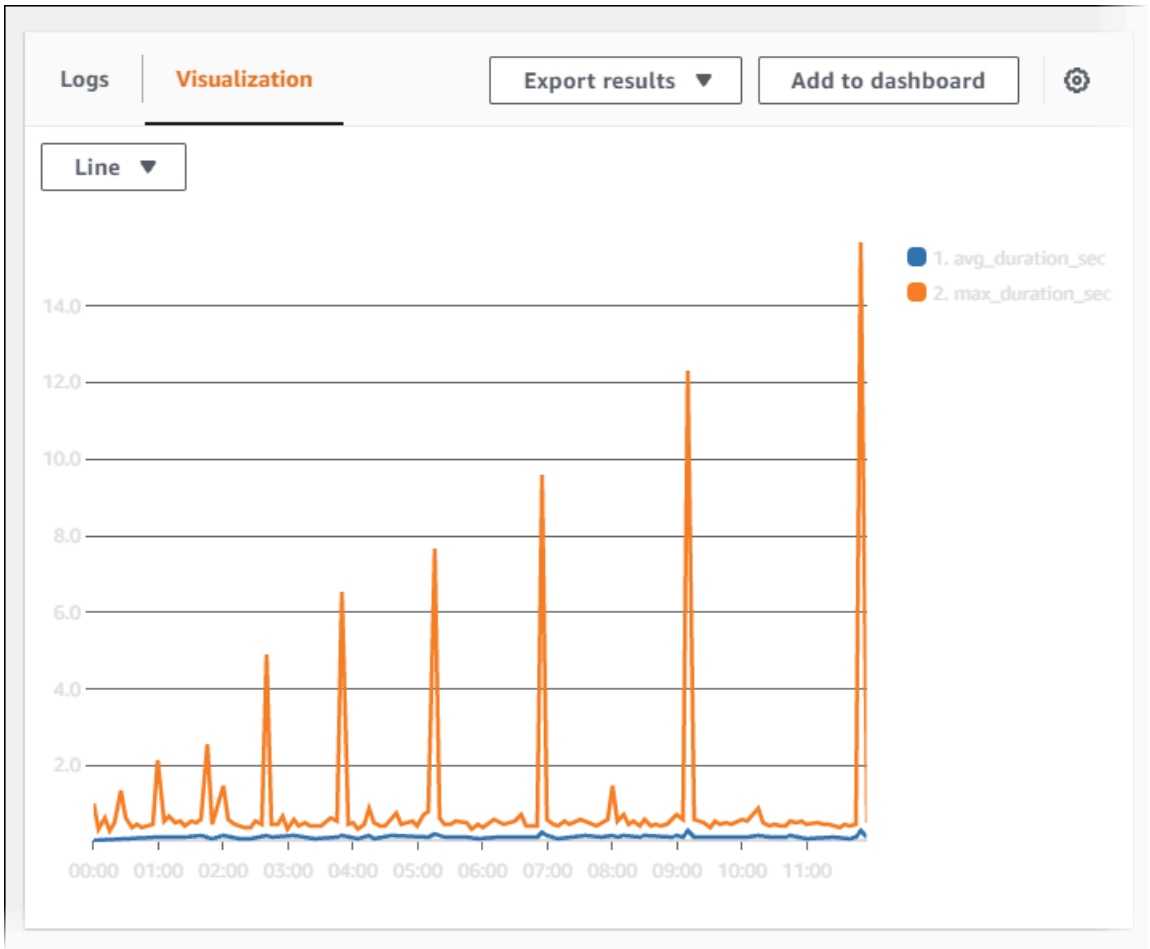


4. Nell'editor di query, elimina la query attualmente visualizzata, quindi immetti quanto segue e scegli Run (Esegui).

```
##Autovacuum execution time in seconds per 5 minute
fields @message
| parse @message "elapsed: * s" as @duration_sec
| filter @message like / automatic vacuum /
| display @duration_sec
| sort @timestamp
| stats avg(@duration_sec) as avg_duration_sec,
max(@duration_sec) as max_duration_sec
by bin(5 min)
```

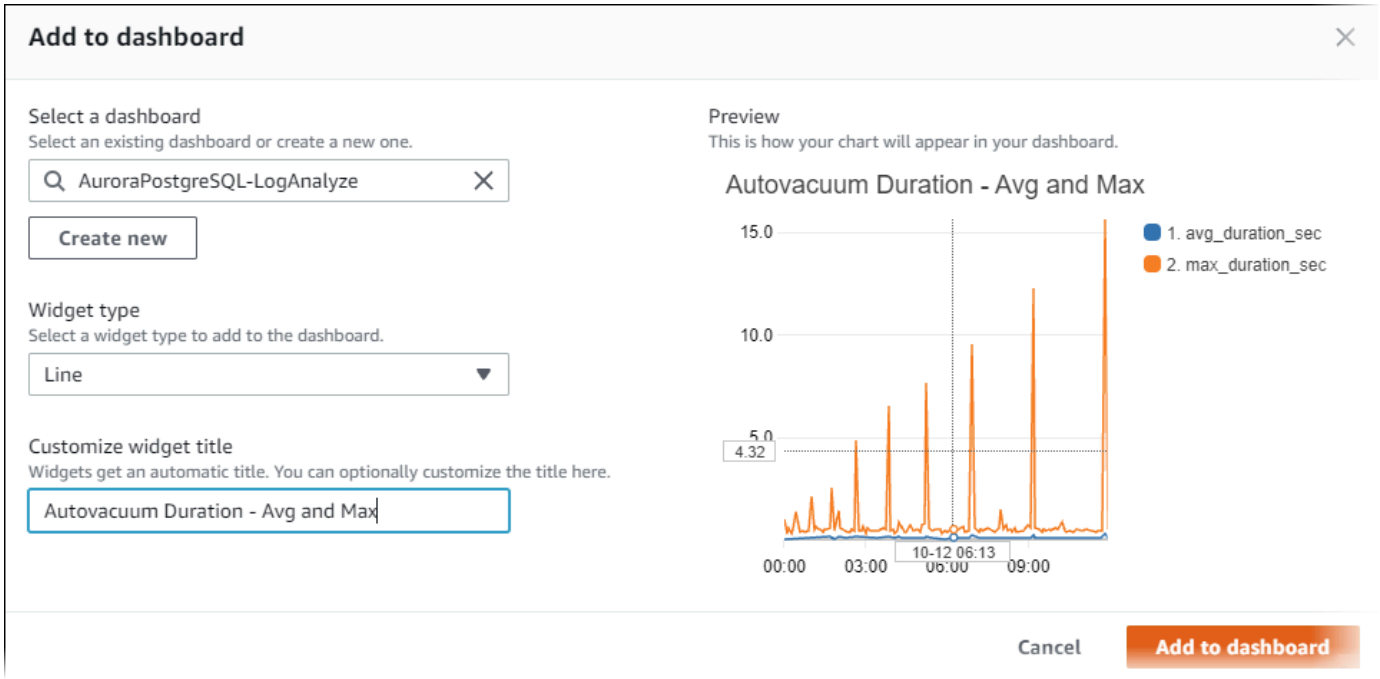


5. Seleziona la scheda Visualization (Visualizzazione).



6. Scegli Add to dashboard (Aggiungi a pannello di controllo).
7. In Seleziona un pannello di controllo, seleziona un pannello di controllo o inserisci un nome per creare un nuovo pannello di controllo.

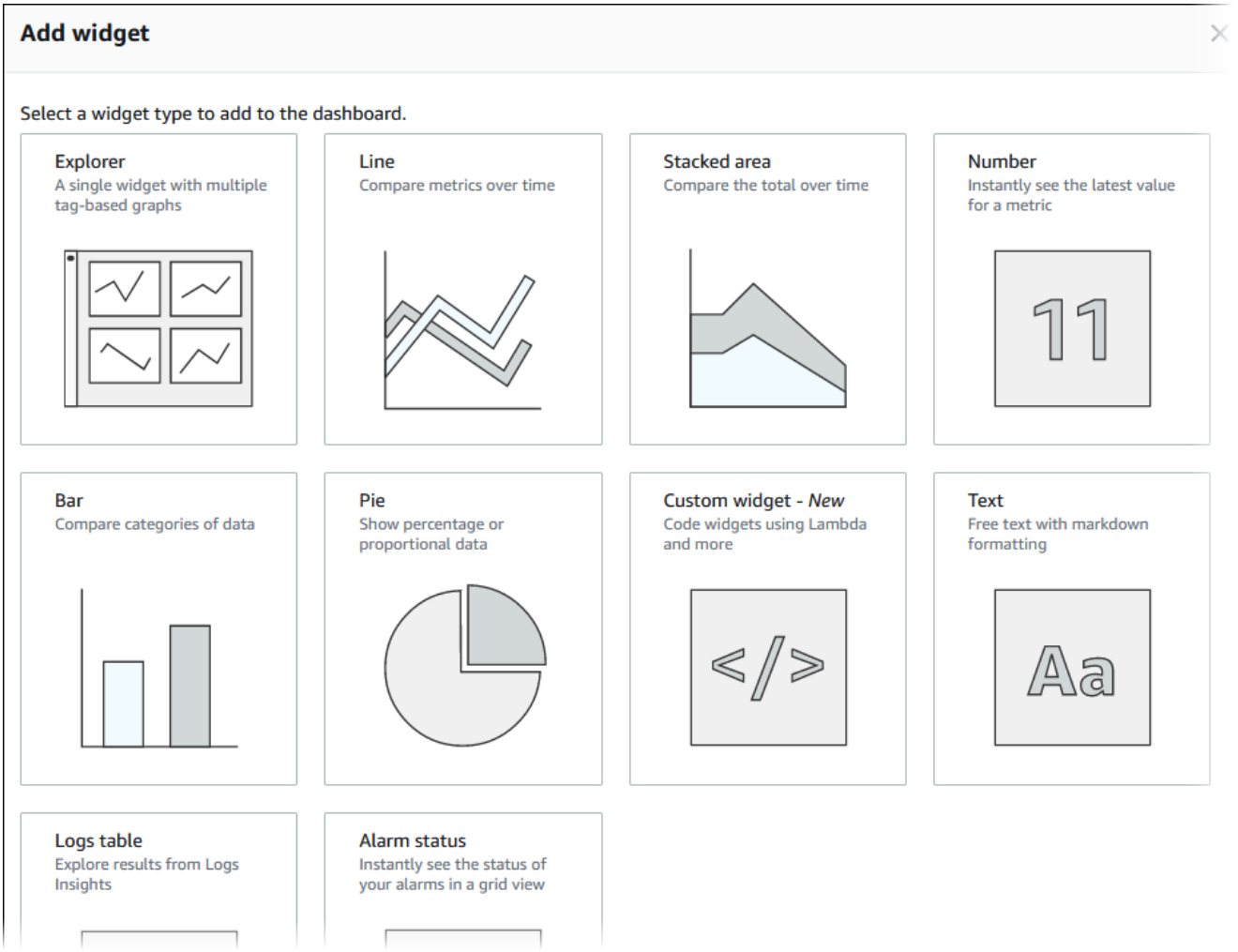
8. In Tipo di widget, scegli un tipo di widget per la tua visualizzazione.



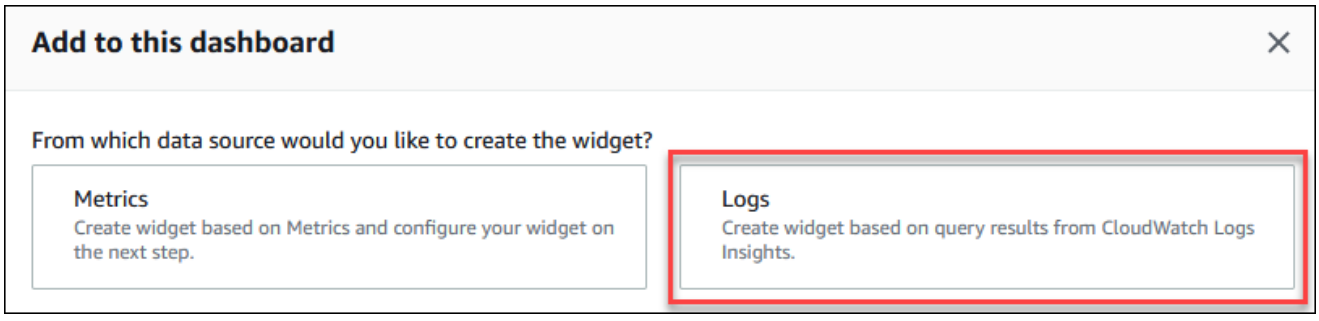
9. (Facoltativo) Aggiungi altri widget in base ai risultati della query di log.

   a. Seleziona Add widget (Aggiungi widget).

   b. Scegli un tipo di widget, ad esempio Line (Linea).



c. Nella finestra Aggiungi a questo pannello di controllo, scegli Log.

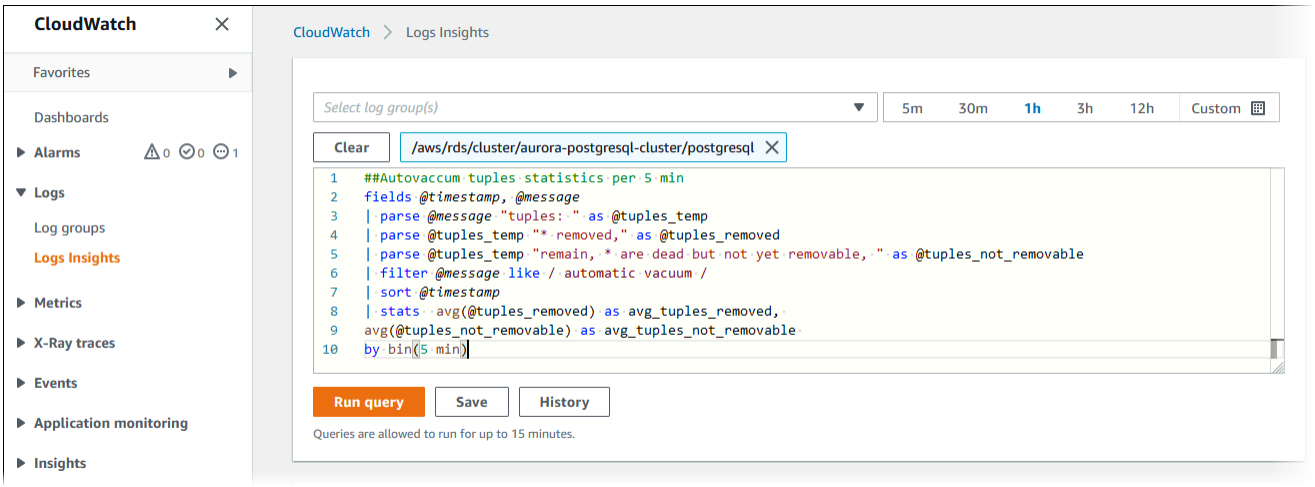


d. Nello stato Seleziona gruppi di log, seleziona il gruppo di log per il cluster di database di.

e. Nell'editor di query, elimina la query attualmente visualizzata, quindi immetti quanto segue e scegli Run (Esegui).

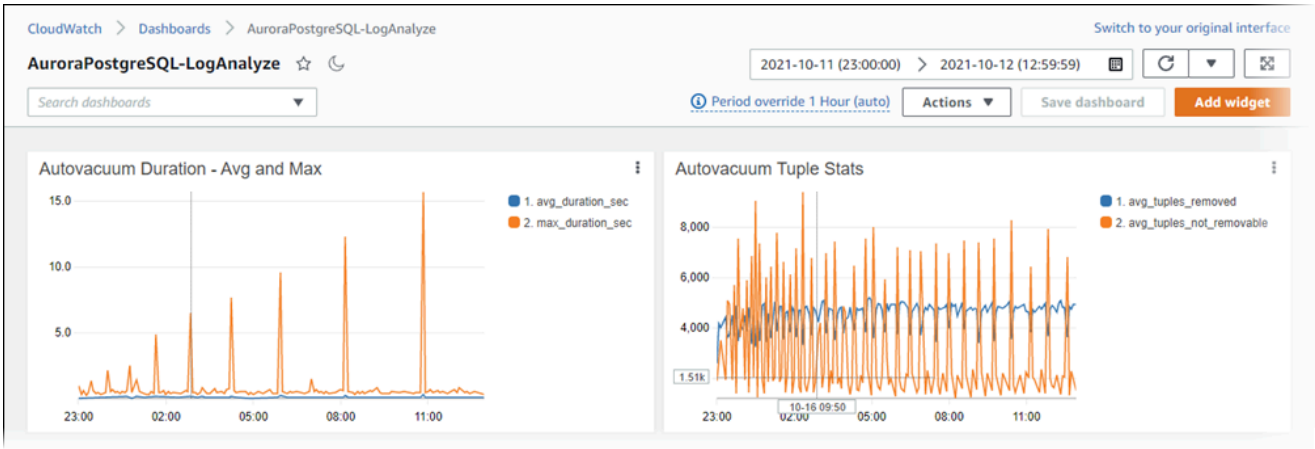
```
##Autovacuum tuples statistics per 5 min
fields @timestamp, @message
| parse @message "tuples: " as @tuples_temp
```

```
| parse @tuples_temp "* removed," as @tuples_removed
| parse @tuples_temp "remain, * are dead but not yet removable, " as
 @tuples_not_removable
| filter @message like / automatic vacuum /
| sort @timestamp
| stats  avg(@tuples_removed) as avg_tuples_removed,
avg(@tuples_not_removable) as avg_tuples_not_removable
by bin(5 min)
```



f. Seleziona Crea widget.

Il pannello di controllo dovrebbe apparire simile alla seguente immagine.



Monitoraggio dei piani di esecuzione delle query e della memoria di picco per Aurora Postgre SQL

È possibile monitorare i piani di esecuzione delle query nell'istanza SQL DB di Aurora Postgre per rilevare i piani di esecuzione che contribuiscono al carico corrente del database e tenere traccia delle statistiche sulle prestazioni dei piani di esecuzione nel tempo utilizzando il parametro. `aurora_compute_plan_id` Ogni volta che viene eseguita una query, al piano di esecuzione utilizzato dalla query viene assegnato un identificatore e lo stesso identificatore viene utilizzato nelle successive esecuzioni dello stesso piano.

`aurora_compute_plan_id`È attivato `OFF` per impostazione predefinita nel gruppo di parametri DB delle SQL versioni 14.10, 15.5 e successive di Aurora Postgre. Per assegnare un identificatore del piano, impostatelo nel gruppo di parametri. `aurora_compute_plan_id` `ON`

Questo identificatore del piano viene utilizzato in diverse utilità che hanno uno scopo diverso.

È possibile monitorare l'utilizzo di picco della memoria delle query nell'istanza DB per rilevare le query che contribuiscono all'elevato utilizzo della memoria del database nelle seguenti versioni:

- 16.3 e tutte le versioni successive
- 15.7 e versioni successive
- 14.12 e versioni successive

Ogni volta che viene eseguita una query, viene tracciata la memoria di picco utilizzata dalla query. Le query vengono in genere eseguite più volte; per ogni query è possibile visualizzare i valori di utilizzo medio, minimo e massimo della memoria in tutte le esecuzioni.

Argomenti

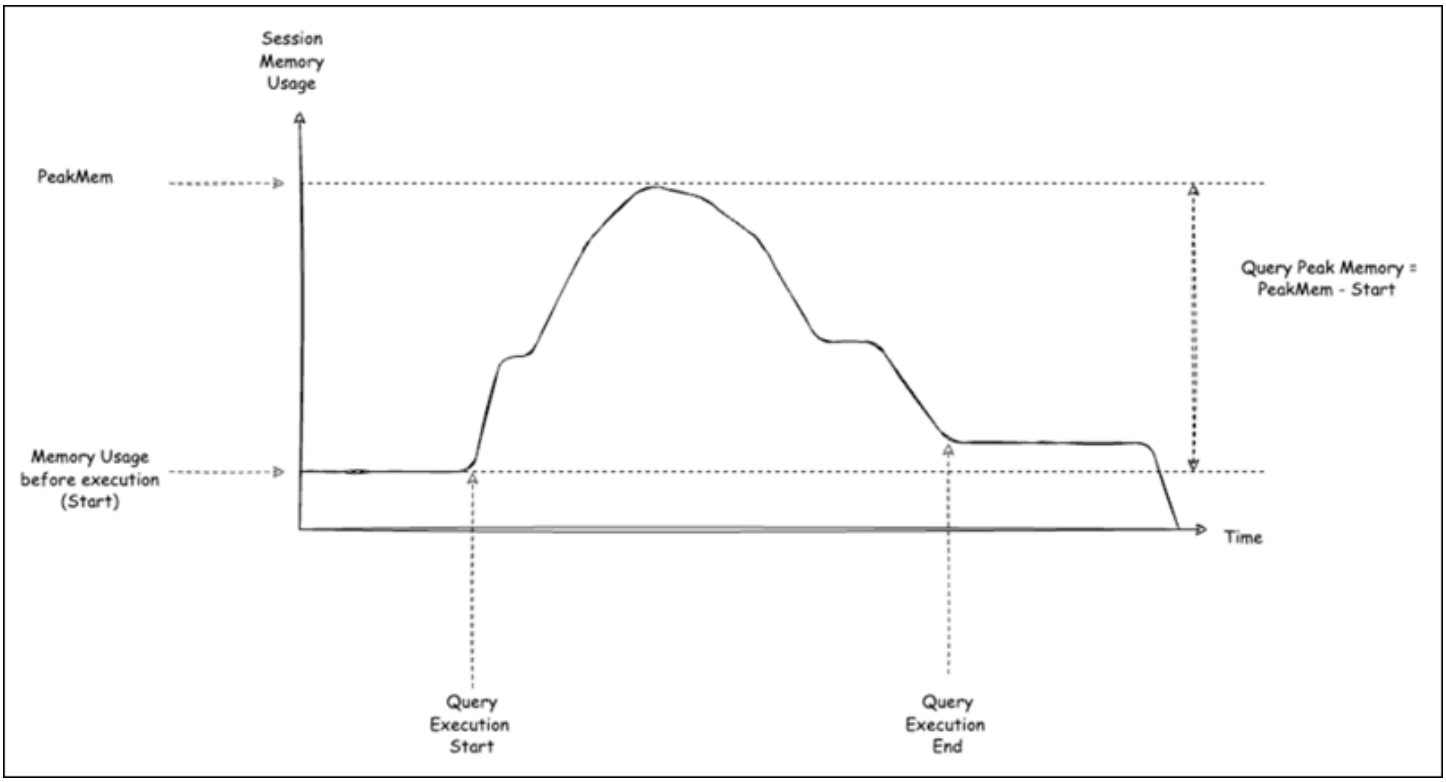
- Accesso ai piani di esecuzione delle query e alla memoria di picco utilizzando le funzioni Aurora
- Riferimento ai parametri per i piani di esecuzione delle query di Aurora SQL Postgre

Accesso ai piani di esecuzione delle query e alla memoria di picco utilizzando le funzioni Aurora

Con`aurora_compute_plan_id`, puoi accedere ai piani di esecuzione utilizzando le seguenti funzioni:

- aurora_stat_activity
- aurora_stat_plans

La memoria di picco delle query non include la memoria allocata prima dell'inizio dell'elaborazione delle query. L'utilizzo di picco della memoria viene monitorato e riportato separatamente per le fasi di pianificazione ed esecuzione di ogni query.



È possibile accedere alle statistiche sulla memoria di picco delle query utilizzando le seguenti funzioni:

- aurora_stat_statements
- aurora_stat_plans

Per ulteriori informazioni su queste funzioni, vedere. Riferimenti relativi alle funzioni Aurora PostgreSQL

Riferimento ai parametri per i piani di esecuzione delle query di Aurora SQL Postgre

È possibile monitorare i piani di esecuzione delle query utilizzando i parametri seguenti in un gruppo di parametri DB.

Parametri

- aurora_compute_plan_id
- aurora_stat_plans.minutes_until_recapture
- aurora_stat_plans.calls_until_recapture
- aurora_stat_plans.with_costs
- aurora_stat_plans.with_analyze
- aurora_stat_plans.with_timing
- aurora_stat_plans.with_buffers
- aurora_stat_plans.with_wal
- aurora_stat_plans.with_triggers

> **Note**
>
> La configurazione `aurora_stat_plans.with_*` dei parametri ha effetto solo per i piani appena acquisiti.

aurora_compute_plan_id

`aurora_compute_plan_id`è un parametro di configurazione che controlla se viene assegnato un identificatore del piano durante l'esecuzione della query.

Predefinita	Valori consentiti	Descrizione
off	0(off)	Impostato `off` per impedire l'assegnazione di un identificatore del piano.
	1(on)	Impostare su per on assegnare un identificatore del piano.

aurora_stat_plans.minutes_until_recapture

Il numero di minuti che devono trascorrere prima che un piano venga ripreso. L'impostazione predefinita è 0, che disabiliterà il recupero di un piano. Una volta `aurora_stat_plans.calls_until_recapture` superata la soglia, il piano verrà recuperato nuovamente.

Predefinita	Valori consentiti	Descrizione
0	0-1073741823	Imposta il numero di minuti che devono trascorrere prima che un piano venga ripreso.

aurora_stat_plans.calls_until_recapture

Il numero di chiamate a un piano prima che venga recuperato. L'impostazione predefinita è 0, che disabiliterà il recupero di un piano dopo un certo numero di chiamate. Una volta `aurora_stat_plans.minutes_until_recapture` superata la soglia, il piano verrà recuperato nuovamente.

Predefinita	Valori consentiti	Descrizione
0	0-1073741823	Imposta il numero di chiamate prima che un piano venga ripristinato.

aurora_stat_plans.with_costs

Acquisisce un piano con costi stimati. EXPLAIN I valori consentiti sono `on` e `off`. Il valore predefinito è `on`.

Predefinita	Valori consentiti	Descrizione
on	0(off)	Non mostra il costo e le righe stimati per ogni nodo del piano.

Predefinita	Valori consentiti	Descrizione
	1(on)	Mostra il costo e le righe stimati per ogni nodo del piano.

aurora_stat_plans.with_analyze

Controlla il piano con. EXPLAIN ANALYZE Questa modalità viene utilizzata solo la prima volta che viene acquisito un piano. I valori consentiti sono `on` e `off`. Il valore predefinito è `off`.

Predefinita	Valori consentiti	Descrizione
off	0(off)	Non include le statistiche effettive sulla durata di esecuzione del piano.
	1(on)	Include le statistiche sulla durata effettiva del piano.

aurora_stat_plans.with_timing

La tempistica del piano verrà riportata nella descrizione di quando viene utilizzata. ANALYZE Il valore predefinito è `on`.

Predefinita	Valori consentiti	Descrizione
on	0(off)	Non include il tempo di avvio effettivo e il tempo impiegato in ogni nodo del piano.
	1(on)	Include il tempo di avvio effettivo e il tempo impiegato in ogni nodo del piano.

aurora_stat_plans.with_buffers

Le statistiche sull'utilizzo del Plan Buffer verranno acquisite nella sezione explain when viene utilizzata. ANALYZE Il valore predefinito è `off`.

Predefinita	Valori consentiti	Descrizione
off	0(off)	Non include informazioni sull'utilizzo del buffer.
	1(on)	Include informazioni sull'utilizzo del buffer.

aurora_stat_plans.with_wal

Le statistiche sull'utilizzo di Plan wal verranno acquisite nella sezione explain when viene utilizzata. ANALYZE Il valore predefinito è `off`.

Predefinita	Valori consentiti	Descrizione
off	0(off)	Non include informazioni sulla generazione di WAL record.
	1(on)	Include informazioni sulla generazione di WAL record.

aurora_stat_plans.with_triggers

Le statistiche sull'esecuzione di Plan Trigger verranno acquisite nella sezione explain when viene utilizzata. `ANALYZE` Il valore predefinito è `off`.

Predefinita	Valori consentiti	Descrizione
off	0(off)	Non include le statistiche di esecuzione dei trigger.
	1(on)	Include le statistiche di esecuzione dei trigger.

Gestione dei piani di esecuzione delle query per Aurora PostgreSQL

La gestione del piano di query per Aurora PostgreSQL è una funzionalità opzionale che è possibile utilizzare con il cluster database Edizione compatibile con PostgreSQL di Amazon Aurora. Questa funzionalità è nel pacchetto come estensione `apg_plan_mgmt` che puoi installare nel cluster database Aurora PostgreSQL. La gestione del piano di query consente di gestire i piani di esecuzione delle query generati dall'ottimizzatore per le applicazioni SQL. L'`apg_plan_mgmt` AWS estensione si basa sulla funzionalità nativa di elaborazione delle query del motore di database PostgreSQL.

Di seguito, sono disponibili informazioni sulle funzionalità di gestione del piano di query per Aurora PostgreSQL, su come configurarlo e su come utilizzarlo con il cluster database Aurora PostgreSQL. Prima di iniziare, ti consigliamo di leggere tutte le note di rilascio per la versione specifica dell'estensione `apg_plan_mgmt` disponibile per la tua versione di Aurora PostgreSQL. Per ulteriori informazioni, consulta Versioni dell'estensione Aurora PostgreSQL apg_plan_mgmt nelle Note di rilascio per Aurora PostgreSQL.

Argomenti

- Panoramica della gestione del piano di query per Aurora PostgreSQL
- Best practice per la gestione del piano di query Aurora PostgreSQL
- Gestione del piano di query PostgreSQL di Aurora
- Acquisizione dei piani di esecuzione Aurora PostgreSQL
- Utilizzo dei piani gestiti per Aurora PostgreSQL
- Esame dei piani di query Aurora PostgreSQL nella vista dba_plans
- Miglioramento dei piani di interrogazione PostgreSQL di Aurora
- Eliminazione dei piani di query PostgreSQL di Aurora
- Esportazione e importazione di piani gestiti per Aurora PostgreSQL
- Documentazione di riferimento dei parametri per la gestione del piano di query Aurora PostgreSQL
- Informazioni di riferimento sulle funzioni per la gestione del piano di query Aurora PostgreSQL
- Riferimento per la vista apg_plan_mgmt.dba_plans per Aurora PostgreSQL Compatible Edition
- Funzionalità avanzate della gestione del piano di query

Panoramica della gestione del piano di query per Aurora PostgreSQL

La gestione del piano di query per Aurora PostgreSQL è progettata per garantire la stabilità del piano a prescindere dalle modifiche apportate al database che potrebbero causare la regressione del piano di query. La regressione del piano di query si verifica quando l'ottimizzatore sceglie un piano non ottimale per una determinata istruzione SQL dopo modifiche al sistema o al database. Le modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database PostgreSQL possono tutte causare la regressione del piano.

La gestione del piano di query per Aurora PostgreSQLconsente di controllare come e quando cambiano i piani di esecuzione delle query. I vantaggi della gestione del piano di query per Aurora PostgreSQL includono i seguenti.

- Migliorare la stabilità del piano forzando l'ottimizzatore a selezionare tra un numero ridotto di piani corretti.
- Ottimizzare i piani centralmente e distribuire i migliori a livello globale.
- Identificare gli indici non utilizzati e valutare l'impatto della creazione o della rimozione di un indice.
- Rilevare automaticamente un nuovo piano a costo minimo individuato dall'ottimizzatore.
- Provare nuove funzionalità dell'ottimizzatore con meno rischi, perché è possibile scegliere di approvare solo le modifiche dei piani che migliorano le performance.

Gli strumenti forniti dalla gestione del piano di query possono essere usati in modo proattivo per specificare il piano migliore per determinate query. In alternativa, è possibile utilizzare la gestione del piano di query per reagire a circostanze mutevoli ed evitare regressioni del piano. Per ulteriori informazioni, consulta Best practice per la gestione del piano di query Aurora PostgreSQL.

Argomenti

- Istruzioni SQL supportate
- Limitazioni della gestione del piano di query
- Terminologia della gestione del piano di query
- Versioni di gestione del piano di query per Aurora PostgreSQL
- Attivazione della gestione del piano di query per Aurora PostgreSQL
- Aggiornamento della gestione del piano di query per Aurora PostgreSQL
- Disattivazione della gestione del piano di query per Aurora PostgreSQL

Istruzioni SQL supportate

La gestione del piano di query supporta i seguenti tipi di istruzioni SQL.

- Qualsiasi istruzione SELECT, INSERT, UPDATE o DELETE, a prescindere dalla complessità.
- Istruzioni preparate. Per ulteriori informazioni, consultare PREPARE nella documentazione di PostgreSQL.
- Istruzioni dinamiche, comprese quelle eseguite in modalità immediata. Per ulteriori informazioni, consultare Dynamic SQL ed EXECUTE IMMEDIATE nella documentazione di PostgreSQL.
- Comandi e istruzioni SQL incorporati. Per ulteriori informazioni, consultare Embedded SQL Commands nella documentazione di PostgreSQL.
- Istruzioni all'interno di funzioni denominate. Per ulteriori informazioni, consultare CREATE FUNCTION nella documentazione di PostgreSQL.
- Istruzioni contenenti tabelle temporanee.
- Istruzioni all'interno di procedure e blocchi DO.

La gestione del piano di query può essere utilizzata con `EXPLAIN` in modalità manuale per acquisire un piano senza eseguirlo effettivamente. Per ulteriori informazioni, consulta Analisi del piano scelto dall'ottimizzatore. Per ulteriori informazioni sulle modalità di gestione del piano di query (manuale, automatica), consultare Acquisizione dei piani di esecuzione Aurora PostgreSQL.

La gestione del piano di query PostgreSQL di Aurora supporta tutte le funzionalità del linguaggio PostgreSQL, tra cui tabelle partizionate, ereditarietà, sicurezza a livello di riga ed espressioni di tabella comuni ricorsive (). CTEs Per ulteriori informazioni su queste caratteristiche del linguaggio PostgreSQL, consultare Table Partitioning, Row Security Policies e WITH Queries (Common Table Expressions) e altri argomenti nella documentazione di PostgreSQL.

Per informazioni sulle diverse versioni della funzionalità di gestione del piano di query di Aurora PostgreSQL, consulta Versioni dell'estensione Aurora PostgreSQL apg_plan_mgmt nelle Note di rilascio per Aurora PostgreSQL.

Limitazioni della gestione del piano di query

La versione corrente della gestione del piano di query di Aurora PostgreSQL include le seguenti limitazioni.

- I piani non vengono acquisiti per le istruzioni che fanno riferimento alle relazioni di sistema: le istruzioni che fanno riferimento alle relazioni di sistema, ad esempio `pg_class`, non vengono

acquisite. Si tratta di un'impostazione predefinita per evitare che venga acquisito un numero elevato di piani generati dal sistema utilizzati internamente. Questo vale anche per le tabelle di sistema all'interno delle viste.

- Potrebbe essere necessaria una classe di istanza DB più grande per il cluster DB Aurora PostgreSQL: a seconda del carico di lavoro, la gestione del piano di query potrebbe richiedere una classe di istanza DB con più di 2 v. CPUs Il numero di `max_worker_processes` è limitato dalla dimensione della classe dell'istanza DB. Il numero di `max_worker_processes` forniti da una classe di istanza database con 2 vCPU (ad esempio db.t3.medium) può non essere sufficiente per un determinato carico di lavoro. Ti consigliamo di scegliere una classe di istanza DB con più di 2 v CPUs per il tuo cluster Aurora PostgreSQL DB se utilizzi la gestione del piano di query.

  Se la classe di istanza database non è in grado di supportare il carico di lavoro, la gestione del piano di query genera un messaggio di errore come quello riportato di seguito.

  ```
  WARNING: could not register plan insert background process
  HINT: You may need to increase max_worker_processes.
  ```

  In questo caso, è necessario aumentare il cluster database Aurora PostgreSQL fino a una dimensione della classe di istanza database con più memoria. Per ulteriori informazioni, consulta Motori DB supportati per classi di istanza database.
- I piani già archiviati nelle sessioni non vengono modificati: la gestione dei piani di query fornisce un modo per influenzare i piani di query senza modificare il codice dell'applicazione. Tuttavia, quando un piano generico è già archiviato in una sessione esistente e se desideri modificarne il piano di query, devi prima impostare `plan_cache_mode` su `force_custom_plan` nel gruppo di parametri del cluster database.
- `queryid` in `apg_plan_mgmt.dba_plans` e `pg_stat_statements` può essere diverso quando:
  - Gli oggetti vengono eliminati e ricreati dopo l'archiviazione in apg_plan_mgmt.dba_plans.
  - La tabella `apg_plan_mgmt.plans` viene importata da un altro cluster.

Per informazioni sulle diverse versioni della funzionalità di gestione del piano di query di Aurora PostgreSQL, consulta Versioni dell'estensione Aurora PostgreSQL apg_plan_mgmt nelle Note di rilascio per Aurora PostgreSQL.

Terminologia della gestione del piano di query

In questo argomento vengono utilizzati i seguenti termini.

istruzione gestita

Un'istruzione SQL acquisita dall'ottimizzatore durante la gestione del piano di query. Un'istruzione gestita ha uno o più piani di esecuzione della query archiviati nella vista `apg_plan_mgmt.dba_plans`.

baseline del piano

L'insieme di piani approvati per un'istruzione gestita specificata. Ovvero, tutti i piani per l'istruzione gestita che contengono "Approvato" per la relativa colonna `status` nella vista `dba_plan`.

cronologia del piano

L'insieme di tutti i piani acquisiti per un'istruzione gestita specificata. La cronologia del piano contiene tutti i piani acquisiti per l'istruzione, a prescindere dallo stato.

regressione del piano di query

Il caso in cui l'ottimizzatore sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente del database, ad esempio una nuova versione di PostgreSQL o modifiche alle statistiche.

Versioni di gestione del piano di query per Aurora PostgreSQL

La gestione del piano di query è supportata da tutte le versioni di Aurora PostgreSQL attualmente disponibili. Per ulteriori informazioni, consultare l'elenco di aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

La funzionalità di gestione del piano di query viene aggiunta al cluster database Aurora PostgreSQL quando si installa l'estensione `apg_plan_mgmt`. Versioni diverse di Aurora PostgreSQL supportano versioni differenti dell'estensione `apg_plan_mgmt`. Ti consigliamo di aggiornare l'estensione gestione del piano di query alla versione più recente per la versione di Aurora PostgreSQL.

> **Note**
>
> Per le note di rilascio per ogni versione dell'estensione `apg_plan_mgmt`, consultare Aurora PostgreSQL apg_plan_mgmt extension versions nelle Note di rilascio per Aurora PostgreSQL.

È possibile identificare la versione in esecuzione sul cluster connettendosi a un'istanza mediante `psql` e utilizzando il metacomando \dx per elencare le estensioni come mostrato di seguito.

```
labdb=> \dx
                        List of installed extensions
     Name      | Version |    Schema     |                              Description
---------------+---------+---------------
+-------------------------------------------------------------------
 apg_plan_mgmt | 1.0     | apg_plan_mgmt | Amazon Aurora with PostgreSQL compatibility
 Query Plan Management
 plpgsql       | 1.0     | pg_catalog    | PL/pgSQL procedural language
(2 rows)
```

L'output mostra che questo cluster utilizza la versione 1.0 dell'estensione. Solo alcune versioni di `apg_plan_mgmt` sono disponibili per una determinata versione di Aurora PostgreSQL. In alcuni casi, potrebbe essere necessario aggiornare il cluster database Aurora PostgreSQL a una nuova versione secondaria o applicare una patch in modo da poter eseguire l'aggiornamento alla versione più recente della gestione del piano di query. La `apg_plan_mgmt` versione 1.0 mostrata nell'output proviene da un cluster database Aurora PostgreSQL versione 10.17 che non dispone di una versione più recente di `apg_plan_mgmt` disponibile. In questo caso, il cluster database Aurora PostgreSQL deve essere aggiornato a una versione più recente di PostgreSQL.

Per ulteriori informazioni sull'aggiornamento di un cluster database Aurora PostgreSQL a una nuova versione di PostgreSQL, consultare Aggiornamenti del motore di database per Amazon Aurora Postgre SQL.

Per informazioni su come aggiornare l'estensione `apg_plan_mgmt`, consultare Aggiornamento della gestione del piano di query per Aurora PostgreSQL.

Attivazione della gestione del piano di query per Aurora PostgreSQL

La configurazione della gestione del piano di query per il cluster database Aurora PostgreSQL implica l'installazione di un'estensione e la modifica di diverse impostazioni dei parametri cluster database. Per installare l'estensione `apg_plan_mgmt` e attivare la funzionalità per il cluster database Aurora PostgreSQL sono necessarie autorizzazioni `rds_superuser`.

L'installazione dell'estensione crea un nuovo ruolo, `apg_plan_mgmt`, che consente agli utenti del database di visualizzare, gestire e mantenere i piani di query. In qualità di amministratore con privilegi `rds_superuser`, assicurati di concedere il ruolo `apg_plan_mgmt` agli utenti del database in base alle esigenze.

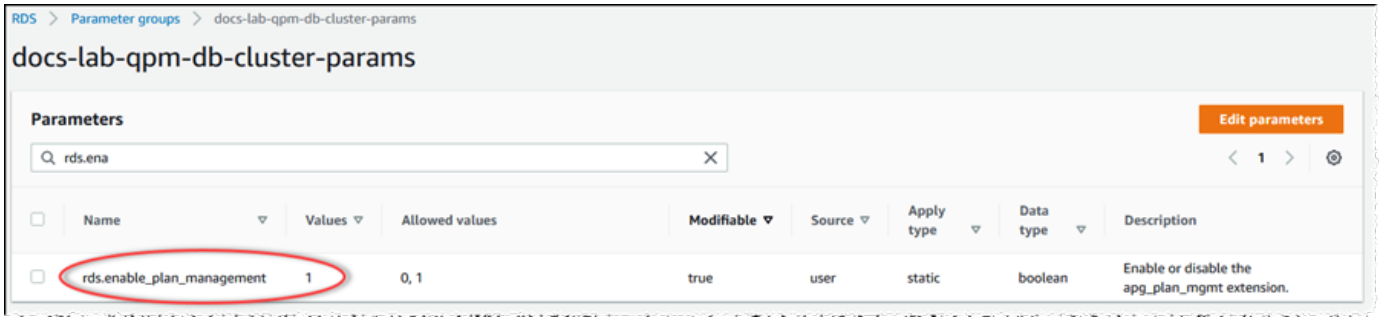
Solo gli utenti con il ruolo `rds_superuser` possono completare la procedura riportata di seguito. Il `rds_superuser` è necessario per creare l'estensione `apg_plan_mgmt` e il relativo ruolo

`apg_plan_mgmt`. Il ruolo `apg_plan_mgmt` deve essere concesso agli utenti per gestire l'estensione `apg_plan_mgmt`.

Come attivare la gestione del piano di query per il cluster database Aurora PostgreSQL

I passaggi seguenti attivano la gestione del piano di query per tutte le istruzioni SQL che vengono inviate al cluster database Aurora PostgreSQL. Questa è nota come modalità automatica. Per ulteriori informazioni sulla differenza tra le modalità, consultare Acquisizione dei piani di esecuzione Aurora PostgreSQL.

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli un gruppo di parametri cluster di database personalizzato per il cluster database Aurora PostgreSQL. Per attivare la gestione del piano di query e impostare il suo comportamento, è necessario modificare determinati parametri. Per ulteriori informazioni, consulta Creazione di un gruppo di parametri DB in Amazon Aurora.
3. Apri il gruppo di parametri cluster di database personalizzato e imposta il parametro `rds.enable_plan_management` su `1`, come mostrato nell'immagine seguente.



   Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.
4. Crea un gruppo di parametri database personalizzato che puoi utilizzare per impostare i parametri del piano di query a livello di istanza. Per ulteriori informazioni, consulta Creazione di un gruppo di parametri del cluster DB in Amazon Aurora.
5. Modifica l'istanza di scrittura del cluster database Aurora PostgreSQL per utilizzare il gruppo di parametri database personalizzato. Per ulteriori informazioni, consulta Modifica di un'istanza database in un cluster database.
6. Modifica il cluster database Aurora PostgreSQL per utilizzare il gruppo di parametri database personalizzato. Per ulteriori informazioni, consulta Modifica del cluster di database tramite la console, la CLI e l'API.
7. Riavvia l'istanza database per abilitare le impostazioni del gruppo di parametri personalizzati.

8. Effettua la connessione all'endpoint dell'istanza database del cluster database Aurora PostgreSQL utilizzando `psql` o pgAdmin. L'esempio seguente utilizza l'account `postgres` predefinito per il ruolo `rds_superuser`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=my-db
```

9. Crea l'estensione `apg_plan_mgmt` per l'istanza database, come mostrato di seguito.

```
labdb=> CREATE EXTENSION apg_plan_mgmt;
CREATE EXTENSION
```

> **Tip**
>
> Installa l'estensione `apg_plan_mgmt` nel database modello per l'applicazione. Il database modello predefinito è denominato `template1`. Per ulteriori informazioni, consultare Template Databases nella documentazione di PostgreSQL.

10. Modifica il parametro `apg_plan_mgmt.capture_plan_baselines` in `automatic`. Questa impostazione consente all'ottimizzatore di generare i piani per ogni istruzione SQL pianificata o eseguita due o più volte.

> **Note**
>
> La gestione del piano di Query dispone anche di una modalità manuale che è possibile utilizzare per istruzioni SQL specifiche. Per ulteriori informazioni, consulta Acquisizione dei piani di esecuzione Aurora PostgreSQL.

11. Modifica il valore del parametro `apg_plan_mgmt.use_plan_baselines` in "on". Questo parametro consente all'ottimizzatore di scegliere un piano per l'istruzione dalla sua baseline del piano. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL.

> **Note**
>
> È possibile modificare il valore di uno di questi parametri dinamici per la sessione senza dover riavviare l'istanza.

Una volta completata la configurazione della gestione del piano di query, assicurati di concedere il ruolo `apg_plan_mgmt` a tutti gli utenti del database che devono visualizzare, gestire o mantenere piani di query.

Aggiornamento della gestione del piano di query per Aurora PostgreSQL

Ti consigliamo di aggiornare l'estensione gestione del piano di query alla versione più recente per la versione di Aurora PostgreSQL.

1. Esegui la connessione all'istanza di scrittura del cluster database Aurora PostgreSQL come un utente che dispone di privilegi `rds_superuser`. Se hai mantenuto il nome predefinito durante la configurazione dell'istanza, esegui la connessione come `postgres`. In questo esempio viene illustrato come utilizzare `psql`, ma puoi anche utilizzare pgAdmin, se preferisci.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Esegui la seguente query per aggiornare l'estensione.

   ```
   ALTER EXTENSION apg_plan_mgmt UPDATE TO '2.1';
   ```

3. Utilizza la funzione apg_plan_mgmt.validate_plans per aggiornare gli hash di tutti i piani. L'ottimizzatore convalida tutti i piani Approvato, Non approvato e Rifiutato per garantire che siano ancora piani validi per la nuova versione dell'estensione.

   ```
   SELECT apg_plan_mgmt.validate_plans('update_plan_hash');
   ```

   Per ulteriori informazioni sull'utilizzo di questa funzione, consultare Convalida dei piani.

4. Utilizza la funzione apg_plan_mgmt.reload per aggiornare eventuali piani nella memoria condivisa con i piani convalidati dalla vista dba_plans.

   ```
   SELECT apg_plan_mgmt.reload();
   ```

Per ulteriori informazioni su tutte le funzioni disponibili per la gestione del piano di query, consultare Informazioni di riferimento sulle funzioni per la gestione del piano di query Aurora PostgreSQL.

Disattivazione della gestione del piano di query per Aurora PostgreSQL

Puoi disabilitare la gestione del piano di query in qualsiasi momento disattivando `apg_plan_mgmt.use_plan_baselines` e `apg_plan_mgmt.capture_plan_baselines`:

```
labdb=> SET apg_plan_mgmt.use_plan_baselines = off;

labdb=> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Best practice per la gestione del piano di query Aurora PostgreSQL

La gestione del piano di query ti permette di controllare come e quando cambiano i piani di esecuzione delle query. In qualità di amministratore di database (DBA), gli obiettivi principali che desideri raggiungere utilizzando QPM sono prevenire le regressioni a seguito di modifiche al database e controllare quando l'ottimizzatore deve utilizzare un nuovo piano. Di seguito sono riportate alcune best practice consigliate su come utilizzare la gestione del piano di query. L'approccio di gestione del piano proattivo differisce da quello reattivo in termini di modalità e tempistiche dell'approvazione dei nuovi piani.

Indice

- Gestione del piano proattiva per prevenire la regressione delle prestazioni
  - Garantire la stabilità del piano dopo un aggiornamento di versione principale
- Gestione del piano reattiva per rilevare e correggere le regressioni delle prestazioni

Gestione del piano proattiva per prevenire la regressione delle prestazioni

Per evitare che si verifichino regressioni delle prestazioni del piano, è possibile far evolvere le baseline del piano eseguendo una procedura che confronta le prestazioni dei piani appena individuati con le prestazioni della baseline esistente dei piani approvati e quindi approva automaticamente il set di piani più rapido come nuova baseline. In questo modo, la baseline dei piani migliora nel tempo man mano che vengono individuati i piani più rapidi.

1. In un ambiente di sviluppo, identificare le istruzioni SQL che hanno il maggiore impatto sulle prestazioni o sulla velocità effettiva del sistema. Quindi acquisire i piani per queste dichiarazioni come descritto in Acquisizione manuale dei piani per specifiche istruzioni SQL e Acquisizione automatica dei piani.

2. Esportare i piani acquisiti dall'ambiente di sviluppo e importarli nell'ambiente di produzione. Per ulteriori informazioni, consulta Esportazione e importazione di piani gestiti per Aurora PostgreSQL.
3. In produzione, eseguire l'applicazione e applicare l'utilizzo di piani gestiti approvati. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL. Durante l'esecuzione dell'applicazione, aggiungere nuovi piani quando l'ottimizzatore li rileva. Per ulteriori informazioni, consulta Acquisizione automatica dei piani.
4. Analizzare i piani non approvati e approvare quelli che funzionano bene. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano.
5. Mentre l'applicazione continua a essere eseguita, l'ottimizzatore inizia a usare i nuovi piani secondo le necessità.

Garantire la stabilità del piano dopo un aggiornamento di versione principale

Ogni versione principale di PostgreSQL include miglioramenti e modifiche all'ottimizzatore di query progettati per potenziare le prestazioni. Tuttavia, i piani di esecuzione delle query generati dal servizio di ottimizzazione nelle versioni precedenti potrebbero causare regressioni delle prestazioni nelle versioni aggiornate più recenti. È possibile utilizzare la gestione del piano di query per risolvere i problemi di prestazione e garantire la stabilità del piano dopo un aggiornamento della versione principale.

L'ottimizzatore utilizza sempre il piano di costo minimo approvato, anche se esistono più piani approvati per la stessa istruzione. Dopo un aggiornamento, l'ottimizzatore potrebbe scoprire nuovi piani, ma questi verranno salvati come piani non approvati. Questi piani vengono eseguiti solo se approvati utilizzando lo stile reattivo di gestione del piano con il parametro unapproved_plan_execution_threshold. È possibile massimizzare la stabilità del piano utilizzando lo stile proattivo di gestione del piano con il parametro evolve_plan_baselines. In questo modo si confrontano le prestazioni dei nuovi piani con quelle dei vecchi piani e si approvano o rifiutano i piani che sono almeno il 10% più veloci rispetto al piano migliore successivo.

Dopo l'aggiornamento, utilizza la funzione `evolve_plan_baselines` con le associazioni dei parametri di query per confrontare le prestazioni del piano prima e dopo l'aggiornamento. I seguenti passaggi presuppongono che tu abbia utilizzato piani gestiti approvati nel tuo ambiente di produzione, come descritto in Utilizzo dei piani gestiti per Aurora PostgreSQL.

1. Prima di procedere con l'aggiornamento, esegui l'applicazione e assicurati che il gestore del piano di query sia in esecuzione. Mentre l'applicazione è in esecuzione, aggiungi i nuovi piani rilevati dall'ottimizzatore. Per ulteriori informazioni, consulta Acquisizione automatica dei piani.

2. Valuta le prestazioni di ciascun piano. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano.
3. Dopo l'aggiornamento, analizza di nuovo i piani approvati utilizzando la funzione `evolve_plan_baselines`. Confronta le prestazioni prima e dopo l'utilizzo delle associazioni dei parametri di query. Se il nuovo piano è veloce, aggiungilo all'elenco dei piani approvati. Se è più veloce di un altro piano in base alle stesse associazioni di parametri, è possibile contrassegnare il piano più lento come Rejected (Rifiutato).

   Per ulteriori informazioni, consulta Approvazione dei piani migliori. Per ulteriori informazioni su questa funzione, consulta apg_plan_mgmt.evolve_plan_baselines.

Per ulteriori informazioni, consulta Garantire prestazioni coerenti dopo gli aggiornamenti delle versioni principali con la gestione del piano di query di Amazon Aurora edizione compatibile con PostgreSQL.

> **Note**
>
> Quando esegui un aggiornamento di una versione principale utilizzando la replica logica di AWS DMS, assicurati di replicare lo schema `apg_plan_mgmt` per garantire che i piani esistenti vengano copiati nell'istanza aggiornata. Per ulteriori informazioni sulla replica logica, consulta Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL.

Gestione del piano reattiva per rilevare e correggere le regressioni delle prestazioni

Rileva quali piani causano regressioni delle prestazioni monitorando l'applicazione mentre è in esecuzione. Quando rilevi regressioni, rifiuta o correggi manualmente i piani inefficienti seguendo questi passaggi:

1. Durante l'esecuzione dell'applicazione, applica l'uso dei piani gestiti e aggiungi automaticamente i piani appena rilevati come non approvati. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL e Acquisizione automatica dei piani.
2. Monitorare l'applicazione in esecuzione per le regressioni delle prestazioni.
3. Quando viene rilevata una regressione del piano, impostare lo stato del piano su `rejected`. La volta successiva che l'ottimizzatore esegue l'istruzione SQL, ignora automaticamente il piano rifiutato e utilizza invece un piano approvato diverso. Per ulteriori informazioni, consulta Rifiuto o disabilitazione dei piani più lenti.

In alcuni casi, si potrebbe preferire correggere un piano danneggiato piuttosto che rifiutarlo, disabilitarlo o cancellarlo. Usare l'estensione `pg_hint_plan` per sperimentare il miglioramento di un piano. Con `pg_hint_plan`, si usano commenti speciali per indicare all'ottimizzatore di ignorare il modo in cui normalmente crea un piano. Per ulteriori informazioni, consulta Correzione dei piani mediante pg_hint_plan.

Gestione del piano di query PostgreSQL di Aurora

Con la gestione del piano di query attivata per il cluster database Aurora PostgreSQL, l'ottimizzatore genera e archivia i piani di esecuzione delle query per qualsiasi istruzione SQL che viene elaborata più di una volta. L'ottimizzatore imposta sempre lo stato del primo piano generato dell'istruzione gestita su `dba_plans` e lo archivia nella vista `Approved`.

Il set di piani approvati salvati per un'istruzione gestita è noto come baseline del piano. Mentre l'applicazione è in corso, l'ottimizzatore potrebbe generare piani aggiuntivi per le istruzioni gestite. L'ottimizzatore imposta i piani acquisiti aggiuntivi su uno stato di `Unapproved`.

In seguito, si può decidere se i piani `Unapproved` vengono eseguiti correttamente e cambiarli in `Approved`, `Rejected` o `Preferred`. A tale scopo, si utilizza la funzione `apg_plan_mgmt.evolve_plan_baselines` o la funzione `apg_plan_mgmt.set_plan_status`.

Quando l'ottimizzatore genera un piano per un'istruzione SQL, la gestione del piano di query salva il piano nella tabella `apg_plan_mgmt.plans`. Gli utenti del database cui è stato concesso il ruolo `apg_plan_mgmt` possono visualizzare i dettagli del piano eseguendo una query sulla vista `apg_plan_mgmt.dba_plans`. Ad esempio, nella seguente query vengono elencati i dettagli dei piani attualmente nella vista per un cluster database Aurora PostgreSQL non di produzione.

- `sql_hash`: un identificatore per l'istruzione SQL che è il valore hash per il testo normalizzato dell'istruzione SQL.
- `plan_hash`: un identificatore univoco per il piano che è una combinazione di `sql_hash` e di un hash del piano.
- `status`: lo stato del piano. L'ottimizzatore può eseguire un piano approvato.
- `enabled`: indica se il piano è pronto per l'uso (true) o no (false).
- `plan_outline`: una rappresentazione del piano che viene utilizzata per ricreare il piano di esecuzione effettivo. Gli operatori nella struttura ad albero vengono mappati agli operatori nell'output EXPLAIN.

La vista `apg_plan_mgmt.dba_plans` contiene molte altre colonne contenenti tutti i dettagli del piano, ad esempio la data dell'ultimo utilizzo del piano. Per tutti i dettagli completi, consultare Riferimento per la vista apg_plan_mgmt.dba_plans per Aurora PostgreSQL Compatible Edition.

Normalizzazione e l'hash SQL

Nella vista `apg_plan_mgmt.dba_plans`, è possibile identificare un'istruzione gestita in base al suo valore hash SQL. L'hash SQL viene calcolato su una rappresentazione normalizzata dell'istruzione SQL che rimuove alcune differenze, ad esempio i valori letterali.

Il processo di normalizzazione per ogni istruzione SQL consente di conservare spazio e maiuscole/minuscole, in modo da poter leggere e comprendere l'essenza dell'istruzione SQL. La normalizzazione rimuove o sostituisce i seguenti elementi.

- Commenti di blocco iniziali
- La parola chiave EXPLAIN e le opzioni EXPLAIN ed EXPLAIN ANALYZE
- Spazi finali
- Tutti i letterali

Ad esempio, considerare la seguente istruzione.

```
/*Leading comment*/ EXPLAIN SELECT /* Query 1 */ * FROM t WHERE x > 7 AND y = 1;
```

La gestione del piano di interrogazione normalizza questa dichiarazione come illustrato:

```
SELECT /* Query 1 */ * FROM t WHERE x > CONST AND y = CONST;
```

La normalizzazione consente di utilizzare lo stesso hash SQL per istruzioni SQL simili che potrebbero differire solo nei loro valori letterali o dei parametri. In altre parole, possono esistere più piani per lo stesso hash SQL, con un piano diverso ottimale in condizioni diverse.

> **Note**
>
> Una singola istruzione SQL utilizzata con schemi diversi dispone di piani diversi perché è associata allo schema specifico in fase di runtime. Il planner utilizza le statistiche per l'associazione dello schema per scegliere il piano ottimale.

Per ulteriori informazioni su come l'ottimizzatore sceglie un piano, consultare Utilizzo dei piani gestiti per Aurora PostgreSQL. In questa sezione vengono fornite informazioni su come utilizzare `EXPLAIN` e `EXPLAIN ANALYZE` per visualizzare l'anteprima di un piano prima che venga effettivamente utilizzato. Per informazioni dettagliate, consultare Analisi del piano scelto dall'ottimizzatore. Per un'immagine che illustra il processo di scelta di un piano, consultare In che modo l'ottimizzatore sceglie quale piano eseguire..

Acquisizione dei piani di esecuzione Aurora PostgreSQL

La gestione del piano di query di Aurora PostgreSQL offre due diverse modalità di acquisizione dei piani di esecuzione delle query: automatica o manuale. La modalità viene scelta impostando il valore di `apg_plan_mgmt.capture_plans_baselines` su `automatic` o su `manual`. Puoi acquisire i piani di esecuzione per specifiche istruzioni SQL utilizzando l'acquisizione manuale del piano. In alternativa, puoi acquisire tutti i piani (o quelli più lenti) che vengono eseguiti due o più volte durante l'esecuzione dell'applicazione utilizzando l'acquisizione automatica del piano.

Durante l'acquisizione dei piani, l'ottimizzatore imposta lo stato del primo piano acquisito dell'istruzione gestita su `approved`. L'ottimizzatore imposta lo stato di qualsiasi altro piano acquisito per un'istruzione gestita su `unapproved`. Tuttavia, a volte, più di un piano potrebbe essere salvato con lo stato `approved`. Ciò può accadere quando vengono creati più piani per un'istruzione in parallelo e prima che venga eseguito il commit del primo piano per l'istruzione.

Per controllare il numero massimo di piani che possono essere acquisiti e archiviati nella visualizzazione `dba_plans`, imposta il parametro `apg_plan_mgmt.max_plans` nel gruppo di parametri a livello di istanza database. La modifica del parametro `apg_plan_mgmt.max_plans` richiede il riavvio dell'istanza database per rendere effettivo il nuovo valore. Per ulteriori informazioni, vedi il parametro apg_plan_mgmt.max_plans.

Acquisizione manuale dei piani per specifiche istruzioni SQL

Se hai un set di istruzioni SQL da gestire, inserisci le istruzioni in un file script SQL e quindi acquisisci manualmente i piani. Di seguito è riportato un esempio di psql per acquisire manualmente i piani di query per un set di istruzioni SQL.

```
psql> SET apg_plan_mgmt.capture_plan_baselines = manual;
psql> \i my-statements.sql
psql> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Dopo aver acquisito un piano per ogni istruzione SQL, l'ottimizzatore aggiunge una nuova riga alla visualizzazione `apg_plan_mgmt.dba_plans`.

Ti consigliamo di utilizzare le istruzioni EXPLAIN o EXPLUTE EXPLAIN nel file di script SQL. Assicurati di includere abbastanza variazioni nei valori dei parametri per acquisire tutti i piani di interesse.

Se conosci un piano migliore del piano a costo minimo dell'ottimizzatore, puoi forzare l'ottimizzatore a utilizzare il piano migliore. Per farlo, specifica uno o più hint dell'ottimizzatore. Per ulteriori informazioni, consulta Correzione dei piani mediante pg_hint_plan. Per confrontare le prestazioni dei piani `unapproved` e `approved` e approvarli, rifiutarli o eliminarli, consulta Valutazione delle prestazioni del piano.

Acquisizione automatica dei piani

Utilizza l'acquisizione automatica dei piani per situazioni come le seguenti:

- Non conosci le istruzioni SQL specifiche da gestire.
- Hai centinaia o migliaia di istruzioni SQL da gestire.
- L'applicazione utilizza un'API client. Ad esempio, JDBC utilizza le istruzioni preparate senza nome o le istruzioni in modalità bulk che non possono essere espresse in psql.

Per acquisire i piani automaticamente

1. Attivare l'acquisizione automatica dei piani impostando `apg_plan_mgmt.capture_plan_baselines` su `automatic` nel gruppo di parametri a livello di istanza database. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri DB in .
2. Durante l'esecuzione dell'applicazione, l'ottimizzatore acquisisce i piani per ogni istruzione SQL eseguita almeno due volte.

   Durante l'esecuzione dell'applicazione con le impostazioni predefinite dei parametri della gestione del piano di query, l'ottimizzatore acquisisce i piani per ogni istruzione SQL eseguita almeno due volte. L'acquisizione di tutti i piani mentre si utilizzano i valori predefiniti ha un overhead di runtime molto ridotto e può essere abilitato in produzione.

Per disattivare l'acquisizione automatica dei piani

- Impostare il parametro `apg_plan_mgmt.capture_plan_baselines` su `off` nel gruppo di parametri a livello di istanza database.

Per misurare le prestazioni dei piani non approvati e approvarli, rifiutarli o eliminarli, consulta Valutazione delle prestazioni del piano.

Utilizzo dei piani gestiti per Aurora PostgreSQL

Per fare in modo che l'ottimizzatore utilizzi i piani acquisiti per le istruzioni gestite, imposta il parametro `apg_plan_mgmt.use_plan_baselines` su `true`. Di seguito è riportato un esempio di istanza locale.

```
SET apg_plan_mgmt.use_plan_baselines = true;
```

Durante l'esecuzione dell'applicazione, questa impostazione fa in modo che l'ottimizzatore utilizzi il piano a costo minimo, preferito o approvato, valido e abilitato per ciascuna istruzione gestita.

Analisi del piano scelto dall'ottimizzatore

Quando il parametro `apg_plan_mgmt.use_plan_baselines` è impostato su `true`, puoi utilizzare le istruzioni SQL EXPLAIN ANALYZE per far sì che l'ottimizzatore mostri il piano che userebbe se dovesse eseguire l'istruzione . Di seguito è riportato un esempio.

```
EXPLAIN ANALYZE EXECUTE rangeQuery (1,10000);
```

```
                                                        QUERY PLAN
-----------------------------------------------------------------------------
 Aggregate  (cost=393.29..393.30 rows=1 width=8) (actual time=7.251..7.251 rows=1
 loops=1)
   ->  Index Only Scan using t1_pkey on t1 t  (cost=0.29..368.29 rows=10000 width=0)
 (actual time=0.061..4.859 rows=10000 loops=1)
Index Cond: ((id >= 1) AND (id <= 10000))
         Heap Fetches: 10000
 Planning time: 1.408 ms
 Execution time: 7.291 ms
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1984047223, Plan Hash: 512153379
```

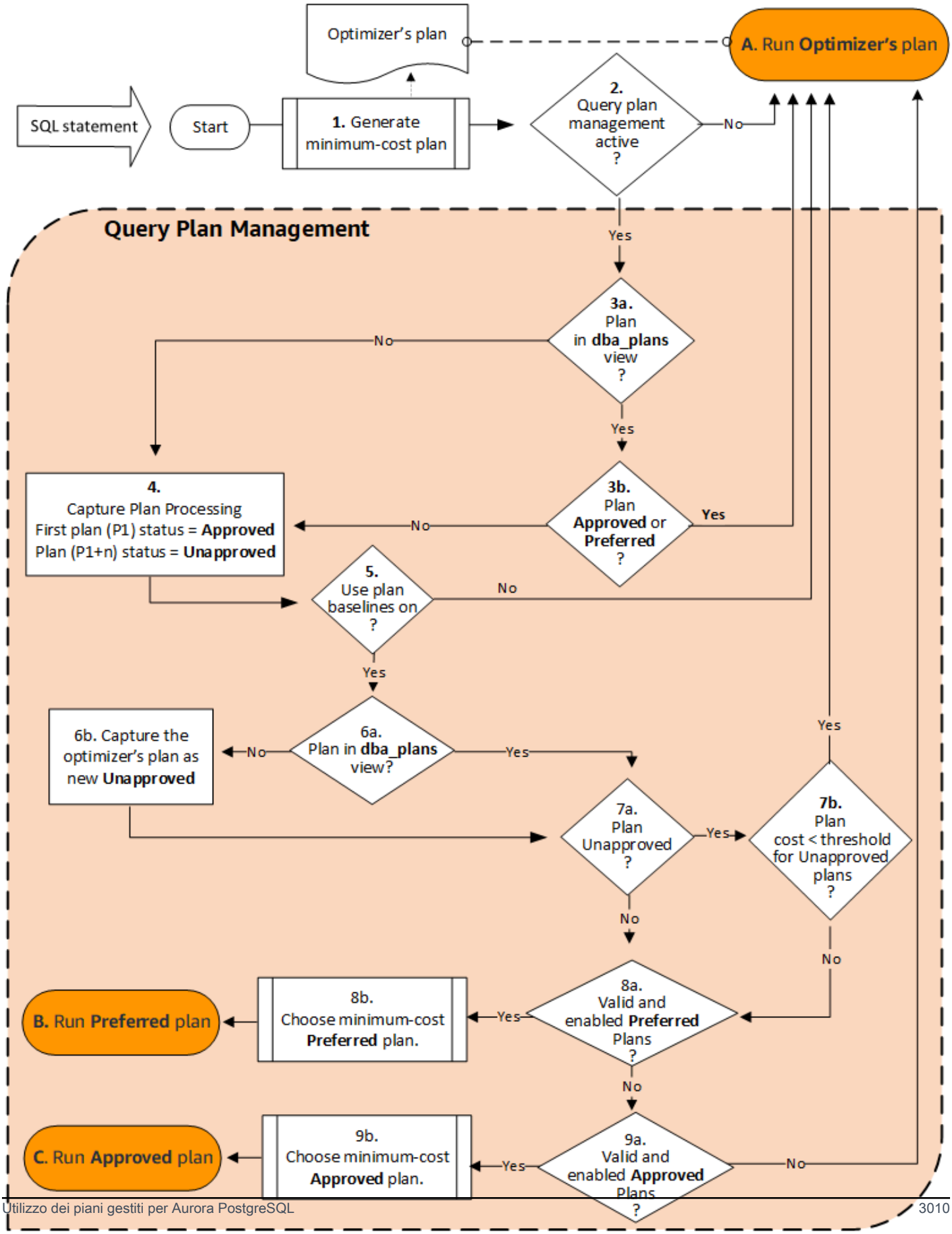
L'output mostra il piano Approvato dalla baseline che viene eseguita. Tuttavia, l'output mostra anche che è stato trovato un piano a costo inferiore. In questo caso, acquisisci questo nuovo piano a costo minimo attivando l'acquisizione automatica del piano come descritto in Acquisizione automatica dei piani.

I nuovi piani vengono sempre acquisiti dall'ottimizzatore come `Unapproved`. Utilizza la funzione `apg_plan_mgmt.evolve_plan_baselines` per confrontare i piani e cambiarli in approvati, rifiutati o disabilitati. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano.

In che modo l'ottimizzatore sceglie quale piano eseguire.

Il costo di un piano di esecuzione è una stima valutata dall'ottimizzatore per confrontare diversi piani. Durante il calcolo del costo di un piano, l'ottimizzatore include fattori quali le operazioni di CPU e I/O richieste da tale piano. Per ulteriori informazioni sui costi del pianificatore query PostgreSQL, consulta Query Planning nella documentazione di PostgreSQL.

Nella seguente immagine viene illustrata la modalità di scelta di un piano per una determinata istruzione SQL, a seconda che la gestione del piano di query sia attiva o non attiva.


Optimizer's plan
A. Run Optimizer's plan
SQL statement
Start
1. Generate minimum-cost plan
2. Query plan management active ?
No
Query Plan Management
Yes
3a. Plan in dba_plans view ?
No
Yes
4. Capture Plan Processing
First plan (P1) status = Approved
Plan (P1+n) status = Unapproved
3b. Plan Approved or Preferred ?
No
Yes
5. Use plan baselines on ?
No
Yes
6a. Plan in dba_plans view?
No
6b. Capture the optimizer's plan as new Unapproved
Yes
7a. Plan Unapproved ?
Yes
7b. Plan cost < threshold for Unapproved plans ?
Yes
No
No
8a. Valid and enabled Preferred Plans ?
Yes
8b. Choose minimum-cost Preferred plan.
B. Run Preferred plan
No
9a. Valid and enabled Approved Plans ?
Yes
9b. Choose minimum-cost Approved plan.
C. Run Approved plan
No

Di seguito è riportato il flusso:

1. L'ottimizzatore genera un piano a costo minimo per l'istruzione SQL.
2. Se la gestione del piano di query non è attiva, il piano dell'ottimizzatore viene eseguito immediatamente (A. Run Optimizer's plan). La gestione del piano di query è inattiva quando si utilizzano le impostazioni predefinite dei parametri `apg_plan_mgmt.capture_plan_baselines` e `apg_plan_mgmt.use_plan_baselines` (rispettivamente "off" e "false").

   In caso contrario, la gestione del piano di query è attiva. In questo caso, l'istruzione SQL e il relativo piano dell'ottimizzatore vengono ulteriormente valutati prima che venga scelto un piano.

   > **Tip**
   >
   > Utenti del database con il ruolo `apg_plan_mgmt` possono confrontare in maniera proattiva i piani, modificare lo stato dei piani e imporre l'uso di piani specifici, in base alle esigenze. Per ulteriori informazioni, consulta Miglioramento dei piani di interrogazione PostgreSQL di Aurora.

3. È possibile che l'istruzione SQL disponga già di piani archiviati da gestione dei piani di query in passato. I piani sono archiviati in `apg_plan_mgmt.dba_plans`, insieme alle informazioni relative alle istruzioni SQL utilizzate per crearli. Le informazioni relative a un piano includono il suo stato. Lo stato di un piano può determinare se è utilizzato o meno, come descritto di seguito.
   a. Se il piano non è incluso tra i piani archiviati per l'istruzione SQL, significa che è la prima volta che questo particolare piano è stato generato dall'ottimizzatore per l'istruzione SQL specificata. Il piano viene inviato all'elaborazione del piano di acquisizione (4).
   b. Se il piano è incluso tra i piani archiviati e il suo stato è Approvato o Preferito, il piano viene eseguito (A. Run Optimizer's plan).

      Se il piano è incluso tra i piani archiviati ma non è né Approvato né Preferito, viene inviato all'elaborazione del piano di acquisizione (4).
4. Quando un piano viene acquisito per la prima volta per una determinata istruzione SQL, il suo stato è sempre impostato su Approvato (P1). Se l'ottimizzatore genera successivamente lo stesso piano per la stessa istruzione SQL, lo stato viene modificato in Non approvato (p1+n).

   Con il piano di acquisito e il relativo stato aggiornato, la valutazione continua nel passaggio successivo (5).

5. La linea di base di un piano è costituita dalla cronologia dell'istruzione SQL e dei relativi piani in vari stati. La gestione del piano di query può tenere conto della linea di base durante la scelta di un piano, a seconda che l'opzione Use plan baselines (Usa linee di base del piano) sia attivata o meno, come illustrato di seguito.
   - L'opzione Use plan baselines (Usa linee di base del piano) è "disattivata" quando il parametro `apg_plan_mgmt.use_plan_baselines` è impostato sul suo valore predefinito (`false`). Il piano non viene confrontato con la linea di base prima di essere eseguito (A. Run Optimizer's plan).
   - L'opzione Use plan baselines (Usa linee di base del piano) è "attivata" quando il parametro `apg_plan_mgmt.use_plan_baselines` è impostato su `true`. Il piano viene ulteriormente valutato utilizzando la linea di base (6).
6. Il piano viene confrontato ad altri piani per l'istruzione nella linea di base.
   a. Se il piano dell'ottimizzatore è incluso tra i piani della linea di base, il suo stato viene controllato (7a).
   b. Se il piano dell'ottimizzatore non è incluso tra i piani della linea di base, viene aggiunto ai piani per l'istruzione come un nuovo piano `Unapproved`.
7. Lo stato del piano viene controllato per determinare solo se è Non è approvato.
   a. Se lo stato del piano è Non approvato, il costo stimato del piano viene confrontato alla stima dei costi specificata per la soglia del piano di esecuzione non approvato.
      - Se il costo stimato del piano è inferiore alla soglia, l'ottimizzatore lo utilizza anche se è Non approvato (A. Run Optimizer's plan). In genere, l'ottimizzatore non esegue un piano Non approvato. Tuttavia, quando il parametro `apg_plan_mgmt.unapproved_plan_execution_threshold` specifica un valore di soglia di costo, l'ottimizzatore confronta il costo del piano Non approvato alla soglia. Se il costo stimato è inferiore alla soglia, l'ottimizzatore esegue il piano. Per ulteriori informazioni, consulta apg_plan_mgmt.unapproved_plan_execution_threshold.
      - Se il costo stimato del piano non è inferiore alla soglia, vengono controllati gli altri attributi del piano (8a).
   b. Se lo stato del piano è diverso da Non approvato, vengono controllati i suoi altri attributi (8a).
8. L'ottimizzatore non utilizzerà un piano che è disabilitato. Ovvero, il piano il cui attributo `enable` è impostato su 'f' (falso). Inoltre, l'ottimizzatore non utilizzerà un piano il cui stato è Rifiutato.

   L'ottimizzatore non può utilizzare piani non validi. I piani possono diventare non validi nel tempo quando gli oggetti da cui dipendono, ad esempio indici e partizioni di tabella, vengono rimossi o eliminati.

a. Se l'istruzione dispone di piani preferiti abilitati e validi, l'ottimizzatore sceglie il piano a costo minimo tra i piani preferiti archiviati per l'istruzione SQL. L'ottimizzatore quindi esegue il piano preferito a costo minimo.
b. Se l'istruzione non dispone di piani preferiti abilitati e validi, viene valutata nel passaggio successivo (9).

9. Se l'istruzione dispone di piani approvati abilitati e validi, l'ottimizzatore sceglie il piano a costo minimo tra i piani approvati archiviati per l'istruzione SQL. L'ottimizzatore quindi esegue il piano approvato a costo minimo.

   Se l'istruzione non dispone di piani approvati validi e abilitati, l'ottimizzatore utilizza il piano a costo minimo (A. Run Optimizer's plan).

Esame dei piani di query Aurora PostgreSQL nella vista dba_plans

Gli utenti e gli amministratori del database cui è stato concesso il ruolo `apg_plan_mgmt` possono visualizzare e gestire i piani archiviati in `apg_plan_mgmt.dba_plans`. Un amministratore di un cluster database Aurora PostgreSQL (un utente con autorizzazioni `rds_superuser`) deve concedere esplicitamente questo ruolo agli utenti del database che devono utilizzare la gestione del piano di query.

La vista `apg_plan_mgmt` contiene la cronologia del piano per tutte le istruzioni SQL gestite per ogni database sull'istanza di scrittura del cluster database Aurora PostgreSQL. Questa vista consente di esaminare i piani, il relativo stato, la data di ultimo utilizzo e tutti gli altri dettagli pertinenti.

Come discusso in Normalizzazione e l'hash SQL, ogni piano gestito è identificato dalla combinazione di un valore hash SQL e un valore hash del piano. Con questi identificatori, puoi usare strumenti come Amazon RDS Performance Insights per tenere traccia delle prestazioni del singolo piano. Per ulteriori informazioni su Performance Insights, consulta Utilizzo di Amazon RDS Performance Insights.

Elenco dei piani gestiti.

Per elencare i piani gestiti, utilizza un'istruzione SELECT sulla visualizzazione `apg_plan_mgmt.dba_plans`. L'esempio seguente mostra alcune colonne nella visualizzazione `dba_plans`, come la colonna `status` che identifica i piani approvati e non approvati.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
 sql_hash   | plan_hash |   status   | enabled | stmt_name
------------+-----------+------------+---------+------------
 1984047223 | 512153379 | Approved   | t       | rangequery
 1984047223 | 512284451 | Unapproved | t       | rangequery
 (2 rows)
```

Per una migliore leggibilità, la query e l'output mostrati elencano solo alcune delle colonne della vista `dba_plans`. Per informazioni complete, consultare Riferimento per la vista apg_plan_mgmt.dba_plans per Aurora PostgreSQL Compatible Edition.

Miglioramento dei piani di interrogazione PostgreSQL di Aurora

Migliora la gestione del piano di interrogazione valutando le prestazioni del piano e definendo i piani. Per ulteriori informazioni sul miglioramento dei piani di query, consulta i seguenti argomenti.

Argomenti

- Valutazione delle prestazioni del piano
- Correzione dei piani mediante pg_hint_plan

Valutazione delle prestazioni del piano

Dopo che l'ottimizzatore acquisisce i piani come non approvati, utilizza la funzione `apg_plan_mgmt.evolve_plan_baselines` per confrontare i piani in base alle prestazioni effettive. A seconda dell'esito degli esperimenti sulle prestazioni, puoi modificare lo stato di un piano da non approvato ad approvato o rifiutato. Puoi invece decidere di utilizzare la funzione `apg_plan_mgmt.evolve_plan_baselines` per disabilitare temporaneamente un piano se non soddisfa i requisiti.

Approvazione dei piani migliori

Nell'esempio seguente viene illustrato come modificare lo stato dei piani gestiti per l'approvazione utilizzando la funzione `apg_plan_mgmt.evolve_plan_baselines`.

```
SELECT apg_plan_mgmt.evolve_plan_baselines (
   sql_hash,
   plan_hash,
   min_speedup_factor := 1.0,
   action := 'approve'
)
```

```
FROM apg_plan_mgmt.dba_plans WHERE status = 'Unapproved';
```

```
NOTICE:     rangequery (1,10000)
NOTICE:     Baseline   [ Planning time 0.761 ms, Execution time 13.261 ms]
NOTICE:     Baseline+1 [ Planning time 0.204 ms, Execution time 8.956 ms]
NOTICE:     Total time benefit: 4.862 ms, Execution time benefit: 4.305 ms
NOTICE:     Unapproved -> Approved
evolve_plan_baselines
----------------------
0
(1 row)
```

L'output mostra un rapporto delle prestazioni per l'istruzione `rangequery` con i binding di parametri 1 e 10.000. Il nuovo piano non approvato (`Baseline+1`) è migliore del piano migliore approvato in precedenza (`Baseline`). Per confermare che il nuovo piano è ora `Approved`, controlla la visualizzazione `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
sql_hash  | plan_hash |  status  | enabled | stmt_name
------------+-----------+----------+---------+------------
1984047223 | 512153379 | Approved | t       | rangequery
1984047223 | 512284451 | Approved | t       | rangequery
(2 rows)
```

Il piano gestito ora include due piani approvati che rappresentano la baseline del piano dell'istruzione. Puoi anche chiamare la funzione `apg_plan_mgmt.set_plan_status` per impostare direttamente il campo di stato di un piano su `'Approved'`, `'Rejected'`, `'Unapproved'` o `'Preferred'`.

Rifiuto o disabilitazione dei piani più lenti

Per rifiutare o disabilitare i piani, passa `'reject'` o `'disable'` come parametro di operazione alla funzione `apg_plan_mgmt.evolve_plan_baselines`. Questo esempio disabilita qualsiasi piano `Unapproved` acquisito che è più lento per almeno il 10 percento del miglior piano `Approved` per l'istruzione.

```
SELECT apg_plan_mgmt.evolve_plan_baselines(
sql_hash,  -- The managed statement ID
plan_hash, -- The plan ID
```

```
1.1,       -- number of times faster the plan must be
'disable'  -- The action to take. This sets the enabled field to false.
)
FROM apg_plan_mgmt.dba_plans
WHERE status = 'Unapproved' AND   -- plan is Unapproved
origin = 'Automatic';       -- plan was auto-captured
```

Puoi anche impostare direttamente un piano su rifiutato o disabilitato. Per impostare direttamente il campo abilitato di un piano su `true` o `false`, chiama la funzione `apg_plan_mgmt.set_plan_enabled`. Per impostare direttamente il campo di stato di un piano su `'Approved'`, `'Rejected'`, `'Unapproved'` o `'Preferred'`, chiama la funzione `apg_plan_mgmt.set_plan_status`.

Per eliminare i piani non validi e che si prevede rimangano non validi, utilizza la funzione `apg_plan_mgmt.validate_plans`. Questa funzione consente di eliminare o disabilitare i piani non validi. Per ulteriori informazioni, consulta Convalida dei piani.

Correzione dei piani mediante pg_hint_plan

L'ottimizzatore di query è progettato per trovare un piano ottimale per tutte le istruzioni e nella maggior parte dei casi trova un buon piano. Tuttavia, occasionalmente potresti realizzare che esiste un piano molto migliore di quello generato dall'ottimizzatore. Due modi consigliati per ottenere che l'ottimizzatore generi un piano desiderato sono l'uso dell'estensione `pg_hint_plan` o l'impostazione delle variabili GUC (Grand Unified Configuration) in PostgreSQL:

- Estensione `pg_hint_plan` – Specifica un "hint" per modificare il funzionamento del pianificatore utilizzando l'estensione `pg_hint_plan` di PostgreSQL. Per installare e ottenere ulteriori informazioni su come utilizzare l'estensione `pg_hint_plan`, consulta la documentazione di pg_hint_plan.
- Variabili GUC – Sostituisci uno o più parametri del modello di costo o altri parametri dell'ottimizzatore, come ad esempio `from_collapse_limit` o `GEQO_threshold`.

Quando utilizzi una di queste tecniche per imporre all'ottimizzatore di query di utilizzare un piano, puoi utilizzare anche la gestione del piano di query per acquisire e imporre l'utilizzo del nuovo piano.

È possibile utilizzare l'estensione `pg_hint_plan` per modificare l'ordine di join, i metodi di join oi percorsi di accesso per un'istruzione SQL. Utilizza un commento SQL con la speciale sintassi `pg_hint_plan` per modificare il modo in cui l'ottimizzatore crea un piano. Ad esempio, supponiamo che l'istruzione SQL del problema abbia un join bidirezionale.

```
SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

Quindi, supponiamo che l'ottimizzatore scelga l'ordine di join (t1, t2), ma si sa che l'ordine di join (t2, t1) è più veloce. Il seguente hint forza l'ottimizzatore a utilizzare l'ordine di join più veloce, (t2, t1). Includi EXPLAIN in modo che l'ottimizzatore generi un piano per l'istruzione SQL ma senza eseguire l'istruzione (output non mostrato).

```
/*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

I passaggi seguenti illustrano come utilizzare `pg_hint_plan`.

Per modificare il piano generato dall'ottimizzatore e acquisire il piano utilizzando pg_hint_plan

1. Attivare la modalità di acquisizione manuale.

   ```
   SET apg_plan_mgmt.capture_plan_baselines = manual;
   ```

2. Specificare un hint per l'istruzione SQL desiderata.

   ```
   /*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
   FROM t1, t2
   WHERE t1.id = t2.id;
   ```

   Al termine dell'esecuzione, l'ottimizzatore acquisisce il piano nella visualizzazione `apg_plan_mgmt.dba_plans`. Il piano acquisito non include la speciale sintassi dei commenti `pg_hint_plan` perché la gestione del piano di query normalizza l'istruzione rimuovendo i commenti iniziali.

3. Visualizzare i piani gestiti utilizzando la visualizzazione `apg_plan_mgmt.dba_plans`.

   ```
   SELECT sql_hash, plan_hash, status, sql_text, plan_outline
   FROM apg_plan_mgmt.dba_plans;
   ```

4. Impostare lo stato del piano su `Preferred`. In questo modo si assicura che l'ottimizzatore scelga di eseguirlo invece di selezionare un piano dall'insieme dei piani approvati quando il piano a costo minimo non è già `Approved` o `Preferred`.

```
SELECT apg_plan_mgmt.set_plan_status(sql-hash, plan-hash, 'preferred' );
```

5. Disattivare l'acquisizione del piano manuale e applicare l'utilizzo dei piani gestiti.

```
SET apg_plan_mgmt.capture_plan_baselines = false;
SET apg_plan_mgmt.use_plan_baselines = true;
```

Ora, quando viene eseguita l'istruzione SQL originale, l'ottimizzatore sceglierà un piano `Approved` o `Preferred`. Se il piano a costo minimo non è `Approved` o `Preferred`, l'ottimizzatore sceglierà il piano `Preferred`.

Eliminazione dei piani di query PostgreSQL di Aurora

Elimina i piani di esecuzione che non utilizzi o i piani non validi. Per ulteriori informazioni sull'eliminazione dei piani, consulta le seguenti sezioni.

Argomenti

- Eliminazione dei piani
- Convalida dei piani

Eliminazione dei piani

I piani vengono eliminati automaticamente se non vengono utilizzati da più di un mese, specificatamente, 32 giorni. Questa è l'impostazione di default per il parametro `apg_plan_mgmt.plan_retention_period`. È possibile modificare il periodo di conservazione del piano in un periodo di tempo più lungo o più breve a partire dal valore di 1. La determinazione del numero di giorni dall'ultimo utilizzo di un piano viene calcolata sottraendo la data `last_used` dalla data corrente. La data `last_used` si riferisce alla data più recente in cui l'ottimizzatore ha scelto un piano come il piano di costo minimo o alla data di esecuzione del piano. La data viene archiviata per il piano nella vista `apg_plan_mgmt.dba_plans`.

Ti consigliamo di eliminare i piani che non sono stati utilizzati per molto tempo o che non sono utili. Ad ogni piano è assegnata una data `last_used` che viene aggiornata dall'ottimizzatore ogni volta che esegue il piano o che sceglie il piano come il piano a costo minimo per un'istruzione. Controlla le date `last_used` più recenti per identificare i piani che è possibile eliminare in modo sicuro.

La seguente query restituisce una tabella a tre colonne con il numero totale di piani, i piani che non sono stati eliminati e i piani eliminati con successo. Include una query annidata che è un esempio di utilizzo della funzione `apg_plan_mgmt.delete_plan` per eliminare tutti i piani che non sono stati scelti come il piano a costo minimo negli ultimi 31 giorni e il cui stato non è `Rejected`.

```
SELECT (SELECT COUNT(*) from apg_plan_mgmt.dba_plans) total_plans,
       COUNT(*) FILTER (WHERE result = -1) failed_to_delete,
       COUNT(*) FILTER (WHERE result = 0) successfully_deleted
       FROM (
            SELECT apg_plan_mgmt.delete_plan(sql_hash, plan_hash) as result
            FROM apg_plan_mgmt.dba_plans
            WHERE last_used < (current_date - interval '31 days')
            AND status <> 'Rejected'
            ) as dba_plans ;
```

```
 total_plans | failed_to_delete | successfully_deleted
-------------+------------------+----------------------
           3 |                0 |                    2
```

Per ulteriori informazioni, consulta apg_plan_mgmt.delete_plan.

Per eliminare i piani non validi e che si prevede rimangano non validi, utilizza la funzione `apg_plan_mgmt.validate_plans`. Questa funzione consente di eliminare o disabilitare i piani non validi. Per ulteriori informazioni, consulta Convalida dei piani.

> **Important**
>
> Se i piani estranei non vengono eliminati, esiste il rischio di esaurimento della memoria condivisa dedicata alla gestione del piano di query. Per controllare la quantità di memoria disponibile per i piani gestiti, utilizzare il parametro `apg_plan_mgmt.max_plans`. Impostare questo parametro nel gruppo di parametri database personalizzati e riavviare l'istanza database per rendere effettive le modifiche. Per ulteriori informazioni, vedi il parametro apg_plan_mgmt.max_plans.

Convalida dei piani

Usa la funzione `apg_plan_mgmt.validate_plans` per eliminare o disabilitare i piani non validi.

I piani possono diventare non validi (obsoleti) quando gli oggetti da cui dipendono vengono rimossi, ad esempio un indice o una tabella. Tuttavia, un piano potrebbe diventare non valido solo temporaneamente se l'oggetto rimosso viene poi ricreato. Se un piano non valido può diventare successivamente valido, potresti preferire di disabilitare un piano non valido o non fare nulla anziché eliminarlo.

Per trovare ed eliminare tutti i piani che non sono validi e non sono stati utilizzati nell'ultima settimana, utilizza la funzione `apg_plan_mgmt.validate_plans` come segue.

```
SELECT apg_plan_mgmt.validate_plans(sql_hash, plan_hash, 'delete')
FROM apg_plan_mgmt.dba_plans
WHERE last_used < (current_date - interval '7 days');
```

Per abilitare o disabilitare un piano direttamente, utilizza la funzione `apg_plan_mgmt.set_plan_enabled`.

Esportazione e importazione di piani gestiti per Aurora PostgreSQL

Puoi esportare i piani gestiti e importarli in un'altra istanza database.

Per esportare i piani gestiti

Un utente autorizzato può copiare qualsiasi sottoinsieme della tabella `apg_plan_mgmt.plans` in un'altra tabella e quindi salvarlo utilizzando il comando pg_dump. Di seguito è riportato un esempio.

```
CREATE TABLE plans_copy AS SELECT *
FROM apg_plan_mgmt.plans [ WHERE predicates ] ;
```

```
% pg_dump --table apg_plan_mgmt.plans_copy -Ft mysourcedatabase > plans_copy.tar
```

```
DROP TABLE apg_plan_mgmt.plans_copy;
```

Per importare i piani gestiti

1. Copiare il file .tar dei piani gestiti esportati nel percorso di sistema in cui devono essere ripristinati i piani.
2. Usare il comando `pg_restore` per copiare il file tar in una nuova tabella.

   ```
   % pg_restore --dbname mytargetdatabase -Ft plans_copy.tar
   ```

3. Unire la tabella `plans_copy` alla tabella `apg_plan_mgmt.plans`, come mostrato nell'esempio seguente.

> **Note**
>
> In alcuni casi, potresti dover eseguire il dump da una versione dell'estensione `apg_plan_mgmt` e ripristinarlo in una versione diversa. In questi casi, le colonne nella tabella dei piani potrebbero essere diverse. In tal caso, denominare le colonne esplicitamente anziché utilizzare SELECT *.

```
INSERT INTO apg_plan_mgmt.plans SELECT * FROM plans_copy
 ON CONFLICT ON CONSTRAINT plans_pkey
 DO UPDATE SET
 status = EXCLUDED.status,
 enabled = EXCLUDED.enabled,
 -- Save the most recent last_used date
 --
 last_used = CASE WHEN EXCLUDED.last_used > plans.last_used
 THEN EXCLUDED.last_used ELSE plans.last_used END,
 -- Save statistics gathered by evolve_plan_baselines, if it ran:
 --
 estimated_startup_cost = EXCLUDED.estimated_startup_cost,
 estimated_total_cost = EXCLUDED.estimated_total_cost,
 planning_time_ms = EXCLUDED.planning_time_ms,
 execution_time_ms = EXCLUDED.execution_time_ms,
 total_time_benefit_ms = EXCLUDED.total_time_benefit_ms,
 execution_time_benefit_ms = EXCLUDED.execution_time_benefit_ms;
```

4. Ricaricare i piani gestiti nella memoria condivisa e rimuovere la tabella dei piani temporanei.

```
SELECT apg_plan_mgmt.reload(); -- refresh shared memory
DROP TABLE plans_copy;
```

Documentazione di riferimento dei parametri per la gestione del piano di query Aurora PostgreSQL

È possibile impostare le preferenze per l'estensione `apg_plan_mgmt` utilizzando i parametri elencati in questa sezione. Questi sono disponibili nel parametro del cluster database personalizzato e

nel gruppo di parametri DB associato al cluster database Aurora PostgreSQL. Questi parametri controllano il comportamento della funzionalità di gestione del piano di query e in che modo influenza l'ottimizzatore. Per informazioni su come impostare la gestione del piano di query, consulta Attivazione della gestione del piano di query per Aurora PostgreSQL. La modifica dei seguenti parametri non ha effetto se l'estensione `apg_plan_mgmt` non è impostata come descritto in tale sezione. Per informazioni sulla modifica dei parametri, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora e Gruppi di parametri DB per DB Amazon Aurora.

Parametri

- apg_plan_mgmt.capture_plan_baselines
- apg_plan_mgmt.plan_capture_threshold
- apg_plan_mgmt.explain_hashes
- apg_plan_mgmt.log_plan_enforcement_result
- apg_plan_mgmt.max_databases
- apg_plan_mgmt.max_plans
- apg_plan_mgmt.plan_hash_version
- apg_plan_mgmt.plan_retention_period
- apg_plan_mgmt.unapproved_plan_execution_threshold
- apg_plan_mgmt.use_plan_baselines
- auto_explain.hashes

apg_plan_mgmt.capture_plan_baselines

Acquisisce i piani di esecuzione delle query generati dall'ottimizzatore per ogni istruzione SQL e li memorizza nella vista `dba_plans`. Per impostazione predefinita, il numero massimo di piani che è possibile memorizzare è 10.000, come specificato dal parametro `apg_plan_mgmt.max_plans`. Per informazioni di riferimento, consulta apg_plan_mgmt.max_plans.

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
off	automatic	Applica questa impostazione a livello di sessione o in un gruppo di parametri per acquisire piani che vengono utilizzati due o più volte.
	manual	Applica questa impostazione a livello di sessione o in un gruppo di parametri per acquisire piani utilizzati una o più volte.
	off	Disattiva l'acquisizione del piano.

Per ulteriori informazioni, consulta Acquisizione dei piani di esecuzione Aurora PostgreSQL.

apg_plan_mgmt.plan_capture_threshold

Specifica una soglia in modo che se il costo totale del piano di esecuzione delle query è inferiore alla soglia, il piano non venga acquisito nella vista `apg_plan_mgmt.dba_plans`.

La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0	0 - 1.79769e+308	Imposta la soglia del costo totale di esecuzione del piano di query `apg_plan_mgmt` per l'acquisizione dei piani.

Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

apg_plan_mgmt.explain_hashes

Specifica se `EXPLAIN [ANALYZE]` mostra `sql_hash` e `plan_hash` alla fine del relativo output. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0	0 (off)	EXPLAIN non mostra sql_hash e plan_hash senza l'opzione true degli hash.
	1 (on)	EXPLAIN mostra l'opzione sql_hash e plan_hash senza l'opzione true degli hash.

apg_plan_mgmt.log_plan_enforcement_result

Specifica se i risultati devono essere registrati per verificare se i piani gestiti dalla funzionalità di gestione dei piani di query vengono utilizzati correttamente. Quando viene utilizzato un piano generico archiviato, non verrà scritto alcun record nei file di log. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
nessuno	nessuno	Non mostra alcun risultato dell'applicazione del piano nei file di log.
	on_error	Mostra i risultati dell'applicazione del piano nei file di log solo quando la funzionalità di gestione dei piani di query non utilizza i piani gestiti.
	tutto	Mostra tutti i risultati dell'applicazione del piano nei file di log, incluse le operazioni riuscite e non riuscite.

apg_plan_mgmt.max_databases

Specifica il numero massimo di database sull'istanza di scrittura del cluster database Aurora PostgreSQL che possono utilizzare la gestione del piano di query. L'impostazione predefinita è 10 database. Se sull'istanza sono presenti più di 10 database, il valore dell'impostazione può essere modificato. Per scoprire quanti database sono presenti in una determinata istanza, connettersi all'istanza utilizzando `psql`. Quindi, utilizzare il metacomando psql, `\l`, per elencare i database.

La modifica del valore di questo parametro richiede il riavvio dell'istanza affinché l'impostazione diventi effettiva.

Predefinita	Valori consentiti	Descrizione
10	10-2147483647	Numero massimo di database che possono utilizzare la gestione del piano di query sull'istanza.

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato.

apg_plan_mgmt.max_plans

Imposta il numero massimo di istruzioni SQL che il gestore del piano di query può mantenere nella visualizzazione `apg_plan_mgmt.dba_plans`. Si consiglia di impostare questo parametro su `10000` o superiore per tutte le versioni di Aurora PostgreSQL.

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato. La modifica del valore di questo parametro richiede il riavvio dell'istanza affinché l'impostazione diventi effettiva.

Predefinita	Valori consentiti	Descrizione
10000	10-2147483647	Numero massimo di piani che possono essere archiviati nella vista `apg_plan_mgmt.dba_plans` . Il valore predefinito per Aurora PostgreSQL versione 10 e versioni precedenti è 1000.

Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

apg_plan_mgmt.plan_hash_version

Specifica i casi d'uso che il calcolo plan_hash è progettato per coprire. Una versione superiore di `apg_plan_mgmt.plan_hash_version` copre tutte le funzionalità della versione inferiore. Ad esempio, la versione 3 copre i casi d'uso supportati dalla versione 2.

La modifica del valore di questo parametro deve essere seguita da una chiamata a `apg_plan_mgmt.validate_plans('update_plan_hash')`. Aggiorna i valori plan_hash in ogni database con apg_plan_mgmt installato e le voci nella tabella dei piani. Per ulteriori informazioni, consulta Convalida dei piani

Predefinita	Valori consentiti	Descrizione
1	1	Calcolo di plan_hash predefinito.
	2	Calcolo di plan_hash modificato per il supporto di più schemi.
	3	Calcolo di plan_hash modificato per il supporto di più schemi e delle tabelle partizionate.
	4	plan_hash calculation modificato per gli operatori paralleli e per supportare i nodi di materializzazione.

apg_plan_mgmt.plan_retention_period

Specifica il numero di giorni per cui si desidera mantenere i piani nella vista `apg_plan_mgmt.dba_plans` prima che vengano eliminati automaticamente. Per impostazione predefinita, un piano viene eliminato quando sono trascorsi 32 giorni dall'ultimo utilizzo del piano (la colonna `last_used` nella vista `apg_plan_mgmt.dba_plans`). Questa impostazione può essere modificata in un numero qualsiasi, 1 e valori superiori.

La modifica del valore di questo parametro richiede il riavvio dell'istanza affinché l'impostazione diventi effettiva.

Predefinita	Valori consentiti	Descrizione
32	1-2147483647	Numero massimo di giorni dall'ultimo utilizzo di un piano prima che venga eliminato.

Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

apg_plan_mgmt.unapproved_plan_execution_threshold

Specifica una soglia di costo al di sotto della quale un piano non approvato può essere utilizzato dall'ottimizzatore. La soglia è 0 per impostazione predefinita, in modo che l'ottimizzatore non esegua piani non approvati. L'impostazione di questo parametro su una soglia di costo estremamente bassa, ad esempio 100, evita il sovraccarico di applicazione del piano nei piani banali. È inoltre possibile impostare questo parametro su un valore estremamente elevato, ad esempio 10000000, utilizzando lo stile reattivo di gestione del piano. Ciò consente all'ottimizzatore di utilizzare tutti i piani scelti senza generare sovraccarichi dovuti all'applicazione del piano. Tuttavia, quando viene trovato un piano errato, è possibile contrassegnarlo manualmente come "rejected" in modo che non venga utilizzato la volta successiva.

Il valore di questo parametro rappresenta una stima dei costi per l'esecuzione di un determinato piano. Se il costo di un piano non approvato è inferiore a tale costo stimato, verrà utilizzato dall'ottimizzatore per l'istruzione SQL. I piani acquisiti e il relativo stato (Approvato, Non approvato) sono visibili nella vista `dba_plans`. Per ulteriori informazioni, consulta Esame dei piani di query Aurora PostgreSQL nella vista dba_plans.

La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0	0-2147483647	Costo del piano stimato al di sotto del quale viene utilizzato un piano non approvato.

Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL.

apg_plan_mgmt.use_plan_baselines

Specifica che l'ottimizzatore deve utilizzare uno dei piani approvati acquisiti e archiviati nella vista `apg_plan_mgmt.dba_plans`. Per impostazione predefinita, questo parametro è disattivato (false). Di conseguenza, l'ottimizzatore utilizzerà il piano a costo minimo che genera senza un'ulteriore valutazione. L'attivazione di questo parametro (impostandolo su true) obbliga l'ottimizzatore a scegliere un piano di esecuzione della query per l'istruzione dalla sua baseline del piano. Per ulteriori informazioni, consulta Utilizzo dei piani gestiti per Aurora PostgreSQL. Per trovare un'immagine che descrive questo processo, consulta In che modo l'ottimizzatore sceglie quale piano eseguire..

Questo parametro può essere impostato nel gruppo di parametri del cluster database personalizzato o nel gruppo di parametri DB personalizzato. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
false	true	Utilizzare un piano Approvato, Preferito o Non approvato del parametro `apg_plan_mgmt.dba_plans` . Se nessuno di questi soddisfa tutti i criteri di valutazione dell'ottimizzatore, è possibile utilizzare il piano a costo minimo generato. Per ulteriori informazioni, consulta In che modo l'ottimizzatore sceglie quale piano eseguire..
	false	Utilizzare il piano a costo minimo generato dall'ottimizzatore.

I tempi di risposta dei diversi piani acquisiti possono essere valutati e lo stato del piano modificato, in base alle esigenze. Per ulteriori informazioni, consulta Miglioramento dei piani di interrogazione PostgreSQL di Aurora.

auto_explain.hashes

Specifica se l'output di auto_explain mostra sql_hash e plan_hash. La modifica del valore di questo parametro non richiede un riavvio.

Predefinita	Valori consentiti	Descrizione
0(off)	0(off)	Il risultato `auto_explain` non mostra `sql_hash` e `plan_hash` .
	1(on)	Il risultato `auto_explain` mostra `sql_hash` e `plan_hash` .

Informazioni di riferimento sulle funzioni per la gestione del piano di query Aurora PostgreSQL

L'estensione `apg_plan_mgmt` fornisce le seguenti funzioni.

Funzioni

- apg_plan_mgmt.copy_outline
- apg_plan_mgmt.delete_plan
- apg_plan_mgmt.evolve_plan_baselines
- apg_plan_mgmt.get_explain_plan
- apg_plan_mgmt.plan_last_used
- apg_plan_mgmt.reload
- apg_plan_mgmt.set_plan_enabled
- apg_plan_mgmt.set_plan_status
- apg_plan_mgmt.update_plans_last_used
- apg_plan_mgmt.validate_plans

apg_plan_mgmt.copy_outline

Copia un determinato hash e una struttura del piano SQL in un hash e una struttura del piano SQL di destinazione, sovrascrivendo così l'hash e la struttura del piano di destinazione. Questa funzione è disponibile in `apg_plan_mgmt` 2.3 e versioni successive.

Sintassi

```
apg_plan_mgmt.copy_outline(
```

```
    source_sql_hash,
    source_plan_hash,
    target_sql_hash,
    target_plan_hash,
    force_update_target_plan_hash
)
```

Valore restituito

Restituisce 0 quando la copia ha esito positivo. Genera eccezioni per gli input non validi.

Parametri

Parametro	Descrizione
`source_sql_hash`	L'ID `sql_hash` associato al `plan_hash` da copiare nella query di destinazione.
`source_plan_hash`	L'ID `plan_hash` da copiare nella query di destinazione.
`target_sql_hash`	L'ID `sql_hash` della query da aggiornare con l'hash e la struttura del piano di origine.
`target_plan_hash`	L'ID `plan_hash` della query da aggiornare con l'hash e la struttura del piano di origine.
`force_update_target_plan_hash`	(Facoltativo) L'`target_plan_hash` ID della query viene aggiornato anche se il piano sorgente non è riproducibile per. `target_sql_hash` Se impostata su true, la funzione può essere utilizzata per copiare i piani tra schemi in cui i nomi delle relazioni e le colonne sono coerenti.

Note per l'utilizzo

Questa funzione consente di copiare un hash e una struttura del piano che utilizza i suggerimenti per altre dichiarazioni simili e quindi evita la necessità di usare istruzioni di suggerimento in linea ad ogni

occorrenza nelle istruzioni di destinazione. Se la query di destinazione aggiornata genera un piano non valido, questa funzione restituisce un errore ed esegue il rollback del tentativo di aggiornamento.

apg_plan_mgmt.delete_plan

Elimina un piano gestito.

Sintassi

```
apg_plan_mgmt.delete_plan(
    sql_hash,
    plan_hash
)
```

Valore restituito

Restituisce 0 se l'eliminazione ha esito positivo o -1 se l'eliminazione non riesce.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.

apg_plan_mgmt.evolve_plan_baselines

Verifica se un piano già approvato è più veloce o se un piano identificato dall'ottimizzatore di query come piano a costo minimo è più veloce.

Sintassi

```
apg_plan_mgmt.evolve_plan_baselines(
    sql_hash,
    plan_hash,
    min_speedup_factor,
    action
)
```

Valore restituito

Il numero di piani che non sono più veloci del miglior piano approvato.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito. Utilizza NULL per indicare tutti i piani che hanno lo stesso valore di ID `sql_hash`.
`min_speedup_factor`	Il fattore di velocità minima indica quante volte un piano deve essere più veloce per essere approvato rispetto al migliore dei piani già approvati In alternativa, questo fattore può corrispondere quante volte deve essere più lento per essere rifiutato o disabilitato. Questo è un valore float positivo.
`action`	L'azione che deve essere eseguita dalla funzione. I valori validi includono i seguenti. Non c'è distinzione tra lettere maiuscole e minuscole. • `'disable'` – Disabilita ogni piano corrispondente che non soddisfa il fattore di accelerazione minimo. • `'approve'` – Abilita ogni piano corrispondente che soddisfa il fattore di accelerazione minimo e imposta lo stato su `approved`. • `'reject'` – Per ogni piano corrispondente che non soddisfa il fattore di accelerazione minimo, imposta lo stato su `rejected`. • NULL – La funzione restituisce semplicemente il numero di piani che non apportano alcun beneficio in termini di prestazioni poiché non soddisfano il fattore di accelerazione minimo.

Note per l'utilizzo

Imposta i piani specificati su approvato, rifiutato o disabilitato in base al fatto che la pianificazione più il tempo di esecuzione sia più veloce del migliore piano approvato da un fattore che puoi impostare. Il parametro action può essere impostato su `'approve'` o `'reject'` per approvare o rifiutare

automaticamente un piano che soddisfa i criteri di prestazione. In alternativa, potrebbe essere impostato su '' (stringa vuota) per eseguire l'esperimento sulle prestazioni e produrre un report, senza intraprendere alcuna azione.

Puoi evitare di rieseguire inutilmente la funzione `apg_plan_mgmt.evolve_plan_baselines` per un piano su cui è stata eseguita di recente. Per farlo, limita i piani solo a quelli creati di recente e non ancora approvati. In alternativa, puoi evitare di eseguire la funzione `apg_plan_mgmt.evolve_plan_baselines` su qualsiasi piano approvato che abbia un timestamp recente `last_verified`.

Conduci un esperimento delle prestazioni per confrontare la pianificazione più il tempo di esecuzione di ciascun piano rispetto agli altri piani nella baseline. In alcuni casi, esiste un solo piano per un'istruzione e il piano è approvato. In tal caso, confronta la pianificazione più il tempo di esecuzione del piano con la pianificazione più il tempo di esecuzione dell'utilizzo di nessun piano.

Il vantaggio (o lo svantaggio) incrementale di ciascun piano è registrato nella visualizzazione `apg_plan_mgmt.dba_plans` nella colonna `total_time_benefit_ms`. Quando questo valore è positivo, c'è un vantaggio misurabile in termini di prestazioni per includere questo piano nella baseline.

Oltre a raccogliere i tempi di pianificazione ed esecuzione di ogni piano candidato, la colonna `last_verified` della visualizzazione `apg_plan_mgmt.dba_plans` viene aggiornata con il `current_timestamp`. È possibile utilizzare il time stamp `last_verified` per evitare di eseguire nuovamente questa funzione su un piano le cui prestazioni sono state verificate di recente.

apg_plan_mgmt.get_explain_plan

Genera il testo di una istruzione `EXPLAIN` per l'istruzione SQL specificata.

Sintassi

```
apg_plan_mgmt.get_explain_plan(
    sql_hash,
    plan_hash,
    [explainOptionList]
)
```

Valore restituito

Restituisce le statistiche di runtime per le istruzioni SQL specificate. Utilizzare senza `explainOptionList` per restituire un piano `EXPLAIN` semplice.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.
`explainOptionList`	Un elenco separato da virgole di opzioni di spiegazione. I valori validi includono `'analyze'` , `'verbose'` , `'buffers'` , `'hashes'` e `'format json'`. Se l'elenco di `explainOptionList` è NULL o una stringa vuota (''), questa funzione genera un'istruzione `EXPLAIN`, senza alcuna statistica.

Note per l'utilizzo

Per il `explainOptionList`, è possibile utilizzare una delle stesse opzioni che si utilizzerebbe con una istruzione `EXPLAIN`. L'ottimizzatore Aurora PostgreSQL concatena l'elenco delle opzioni fornite all'istruzione `EXPLAIN`.

apg_plan_mgmt.plan_last_used

Restituisce la data `last_used` del piano specificato dalla memoria condivisa.

> **Note**
>
> Il valore nella memoria condivisa è sempre aggiornato sull'istanza database primaria nel cluster database. Il valore viene scaricato solo periodicamente nella colonna `last_used` della visualizzazione `apg_plan_mgmt.dba_plans`.

Sintassi

```
apg_plan_mgmt.plan_last_used(
    sql_hash,
```

```
    plan_hash
)
```

Valore restituito

Restituisce la data `last_used`.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.

apg_plan_mgmt.reload

Ricarica i piani nella memoria condivisa dalla visualizzazione `apg_plan_mgmt.dba_plans`.

Sintassi

```
apg_plan_mgmt.reload()
```

Valore restituito

Nessuna.

Parametri

Nessuna.

Note per l'utilizzo

Chiama `reload` per le seguenti situazioni:

- Usalo per aggiornare immediatamente la memoria condivisa di una replica di sola lettura, invece di aspettare che i nuovi piani si propaghino alla replica.
- Usalo dopo l'importazione dei piani gestiti.

apg_plan_mgmt.set_plan_enabled

Abilita o disabilita un piano gestito.

Sintassi

```
apg_plan_mgmt.set_plan_enabled(
    sql_hash,
    plan_hash,
    [true | false]
)
```

Valore restituito

Restituisce 0 se l'impostazione ha esito positivo o -1 se l'impostazione non riesce.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.
`enabled`	Valore booleano true o false: • Il valore `true` abilita il piano. • Il valore `false` disabilita il piano.

apg_plan_mgmt.set_plan_status

Impostare lo stato di un piano gestito su `Approved`, `Unapproved`, `Rejected` o `Preferred`.

Sintassi

```
apg_plan_mgmt.set_plan_status(
    sql_hash,
```

```
    plan_hash,
    status
)
```

Valore restituito

Restituisce 0 se l'impostazione ha esito positivo o -1 se l'impostazione non riesce.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito.
`status`	Stringa con uno dei seguenti valori: • `'Approved'` • `'Unapproved'` • `'Rejected'` • `'Preferred'` Il caso utilizzato non ha importanza, tuttavia il valore di stato è impostato su maiuscole iniziali nella vista `apg_plan_mgmt.dba_plans`. Per ulteriori informazioni su questi valori, consulta `status` in Riferimento per la vista apg_plan_mgmt.dba_plans per Aurora PostgreSQL Compatible Edition.

apg_plan_mgmt.update_plans_last_used

Aggiorna immediatamente la tabella dei piani con la data `last_used` memorizzata nella memoria condivisa.

Sintassi

```
apg_plan_mgmt.update_plans_last_used()
```

Valore restituito

Nessuna.

Parametri

Nessuna.

Note per l'utilizzo

Eseguire una chiamata a `update_plans_last_used` per assicurarsi che le query contro la colonna `dba_plans.last_used` utilizzino le informazioni più aggiornate. Se il la data `last_used` non viene aggiornata immediatamente, un processo in background aggiorna la tabella dei piani con la data `last_used` una volta ogni ora (per impostazione predefinita).

Ad esempio, se un'istruzione con una certa `sql_hash` inizia a funzionare lentamente, è possibile determinare quali piani per quell'istruzione sono stati eseguiti dall'inizio della regressione delle prestazioni. Per fare ciò, innanzitutto svuotare i dati nella memoria condivisa su disco in modo che le date `last_used` siano correnti e quindi interroghino tutti i piani dell'istruzione `sql_hash` con la regressione delle prestazioni. Nella query, assicurati che la data `last_used` sia maggiore o uguale alla data in cui è iniziata la regressione delle prestazioni. La query identifica il piano o l'insieme di piani che potrebbero essere responsabili della regressione delle prestazioni. È possibile utilizzare `apg_plan_mgmt.get_explain_plan` con `explainOptionList` impostato su `verbose, hashes`. È possibile utilizzare anche `apg_plan_mgmt.evolve_plan_baselines` per analizzare il piano e qualsiasi piano alternativo che potrebbe funzionare meglio.

La funzione `update_plans_last_used` ha un effetto solo sull'istanza database primaria del cluster database.

apg_plan_mgmt.validate_plans

Convalida che l'ottimizzatore può ancora ricreare piani. L'ottimizzatore convalida i piani `Approved`, `Unapproved` e `Preferred`, se il piano è abilitato o disabilitato. I piani `Rejected` non sono convalidati. Facoltativamente, puoi utilizzare la funzione `apg_plan_mgmt.validate_plans` per eliminare o disabilitare i piani non validi.

Sintassi

```
apg_plan_mgmt.validate_plans(
    sql_hash,
```

```
    plan_hash,
    action)

apg_plan_mgmt.validate_plans(
    action)
```

Valore restituito

Numero di piani non validi.

Parametri

Parametro	Descrizione
`sql_hash`	L'ID `sql_hash` dell'istruzione SQL gestita del piano.
`plan_hash`	L'ID `plan_hash` del piano gestito. Utilizza NULL per indicare tutti i piani con lo stesso valore di ID `sql_hash`.
`action`	L'azione che deve essere eseguita dalla funzione per i piani non validi. I valori di stringa validi includono i seguenti: Non c'è distinzione tra lettere maiuscole e minuscole. • `'disable'` – Ogni piano non valido è disabilitato. • `'delete'` – Ogni piano non valido è eliminato. • `'update_plan_hash'` – Aggiorna l'ID `plan_hash` per i piani che non possono essere riprodotti esattamente. Consente inoltre di correggere un piano riscrivendo l'SQL. Puoi registrare il piano buono come piano `Approved` per l'SQL di origine. • NULL – La funzione restituisce semplicemente il numero di piani non validi. Non vengono eseguite altre operazioni. • '' – Una stringa vuota genera un messaggio che indica il numero di piani validi e non validi. Qualsiasi altro valore viene considerato come una stringa vuota.

Note per l'utilizzo

Usa il modulo `validate_plans(action)` per convalidare tutti i piani gestiti per tutte le istruzioni gestite nell'intera visualizzazione `apg_plan_mgmt.dba_plans`.

Usa il modulo `validate_plans(sql_hash, plan_hash, action)` per convalidare un piano gestito specificato con `plan_hash`, per un'istruzione gestita specificata con `sql_hash`.

Usa il modulo `validate_plans(sql_hash, NULL, action)` per convalidare tutti i piani gestiti per l'istruzione gestita specificata con `sql_hash`.

Riferimento per la vista apg_plan_mgmt.dba_plans per Aurora PostgreSQL Compatible Edition

Le colonne del piano nella visualizzazione `apg_plan_mgmt.dba_plans` includono le seguenti.

Colonna dba_plans	Descrizione
`cardinality_error`	Misura dell'errore tra la cardinalità stimata e alla cardinalità effettiva. Cardinality è il numero di righe della tabella che saranno elaborate dal piano. Se l'errore di cardinalità è ampio, aumenta la probabilità che il piano non sia ottimale. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`compatibility_level`	Questo parametro mostra quando un piano di query è stato convalidato l'ultima volta. Nelle versioni 12.19, 13.15, 14.12, 15.7, 16.3 e successive di Aurora PostgreSQL, visualizza il numero di versione di Aurora. Per le versioni precedenti, visualizza un numero di versione specifico della funzionalità. **Note** Mantieni il valore di questo parametro all'impostazione predefinita. Aurora PostgreSQL imposta e aggiorna automaticamente questo valore.
`created_by`	L'utente autenticato (`session_user`) che ha creato il piano.

Colonna dba_plans	Descrizione
`enabled`	Indicatore che mostra se il piano è abilitato o disabilitato. Tutte i piani sono abilitati per impostazione predefinita. Puoi disabilit are i piani per impedire che vengano utilizzati dall'ottimizzatore . Per modificare questo valore, utilizza la funzione apg_plan_ mgmt.set_plan_enabled.
`environment_variab les`	Parametri e valori PostgreSQL Grand Unified Configuration (GUC) che l'ottimizzatore ha ignorato nel momento in cui è stato acquisito il piano.
`estimated_startup_ cost`	Il costo di installazione stimato dell'ottimizzatore prima che fornisca le righe di una tabella.
`estimated_total_cost`	Il costo dell'ottimizzatore stimato per la consegna della riga finale della tabella.
`execution_time_ben efit_ms`	Il tempo di esecuzione beneficia in termini di millisecondi dell'abil itazione del piano. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`execution_time_ms`	Il tempo stimato di esecuzione del piano in millisecondi. Questa colonna è popolata dalla funzione apg_plan_mgmt.evol ve_plan_baselines.
`has_side_effects`	Valore che indica che l'istruzione SQL è un'istruzione DML (Data Manipulation Language) o un'istruzione SELECT che contiene una funzione VOLATILE.

Colonna dba_plans	Descrizione
`last_used`	Questo valore viene aggiornato alla data corrente ogni volta che il piano viene eseguito o quando il piano è il piano a costo minimo dell'ottimizzatore di query. Questo valore è archiviato nella memoria condivisa e periodicamente viene riportato su disco. Per ottenere il massimo up-to-date valore, leggi la data dalla memoria condivisa chiamando la funzione `apg_plan_mgmt.plan_last_used(sql_hash, plan_hash)` anziché leggere il valore. `last_used` Per ulteriori informazioni, vedi il parametro apg_plan_mgmt.plan_retention_period.
`last_validated`	La data e l'ora della verifica più recente del piano che potrebbe essere ricreato dalla funzione apg_plan_mgmt.validate_plans o apg_plan_mgmt.evolve_plan_baselines.
`last_verified`	La data e l'ora della verifica più recente di un piano come il piano più performante per i parametri specificati dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`origin`	Come il piano è stato acquisito con il parametro apg_plan_mgmt.capture_plan_baselines. I valori validi includono i seguenti: `M` – Il piano è stato acquisito con l'acquisizione manuale del piano. `A` – Il piano è stato acquisito con l'acquisizione automatica del piano.
`param_list`	I valori dei parametri che sono stati passati all'istruzione se è un'istruzione preparata.
`plan_created`	La data e l'ora di creazione del piano.
`plan_hash`	Identificatore del piano. La combinazione di `plan_hash` e `sql_hash` identifica in modo univoco un piano specifico.

Colonna dba_plans	Descrizione
`plan_outline`	Rappresentazione del piano che viene utilizzata per ricreare il piano di esecuzione effettivo e che è indipendente dal database. Gli operatori nella struttura ad albero corrispondono agli operatori presenti nell'output di EXPLAIN.
`planning_time_ms`	Il tempo effettivo per eseguire il pianificatore, in millisecondi. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.
`queryId`	Un hash dell'istruzione, come calcolato dall'estensione `pg_stat_statements` . Questo non è un identificatore stabile o indipendente dal database perché dipende dagli identificatori di oggetti (). OIDs Il valore sarà `0` se `compute_query_id` è `off` durante l'acquisizione del piano di query.
`sql_hash`	Valore hash del testo dell'istruzione SQL, normalizzato con valori letterali rimossi.
`sql_text`	Il testo completo dell'istruzione SQL.

Colonna dba_plans	Descrizione
`status`	Lo stato di un piano che determina il modo in cui l'ottimizzatore utilizza un piano. I valori validi includono i seguenti. • `Approved`: un piano utilizzabile che l'ottimizzatore può scegliere di eseguire. L'ottimizzatore esegue il piano meno costoso da una serie di piani approvati dell'istruzione gestita (baseline). Per reimpostare un piano su approvato, utilizza la funzione apg_plan_mgmt.evolve_plan_baselines. • `Unapproved` : un piano acquisito che non hai verificato per l'uso. Per ulteriori informazioni, consulta Valutazione delle prestazioni del piano. • `Rejected`: un piano che non viene utilizzato dall'ottimizzatore . Per ulteriori informazioni, consulta Rifiuto o disabilitazione dei piani più lenti. • `Preferred` : un piano che hai indicato come piano preferito da utilizzare per un'istruzione gestita. Se il piano a costo minimo dell'ottimizzatore non è un piano approvato o preferito, puoi ridurre l'overhead derivante dall'applicazione del piano. Per farlo, indica come sottoinsi eme dei piani approvat `Preferred` . Quando il piano a costo minimo dell'ottimizzatore non è un piano `Approved`, viene scelto un piano `Preferred` prima di un piano `Approved`. Per reimpostare un piano su `Preferred` , utilizza la funzione apg_plan_mgmt.set_plan_status.
`stmt_name`	Il nome dell'istruzione SQL all'interno di un'istruzione PREPARE. Questo valore è una stringa vuota per un'istruzione preparata senza nome. Questo valore è NULL per un'istruzione non preparata.

Colonna dba_plans	Descrizione
`total_time_benefit_ms`	Il vantaggio in termini di tempo totale in millisecondi derivato dall'abilitazione di questo piano. Questo valore considera sia il tempo di pianificazione che il tempo di esecuzione. Se il valore è negativo, non è vantaggioso abilitare questo piano. Questa colonna è popolata dalla funzione apg_plan_mgmt.evolve_plan_baselines.

Funzionalità avanzate della gestione del piano di query

Di seguito puoi trovare informazioni sulle funzionalità avanzate di Aurora Postgre SQL Query Plan Management (): QPM

Argomenti

- Acquisizione dei piani di esecuzione Aurora PostgreSQL nelle repliche
- Supporto della partizione di tabelle

Acquisizione dei piani di esecuzione Aurora PostgreSQL nelle repliche

QPM (Query Plan Management) consente di acquisire i piani di query generati dalle repliche di Aurora e di archiviarli sull'istanza database principale del cluster di database Aurora. È possibile raccogliere i piani di query da tutte le repliche Aurora e mantenere un insieme di piani ottimali in una tabella centrale persistente sull'istanza primaria. Potrai applicare tali piani su altre repliche secondo necessità. Ciò consente di mantenere la stabilità dei piani di esecuzione e di migliorare le prestazioni delle query tra i cluster database e le versioni del motore.

Argomenti

- Prerequisiti
- Gestione dell'acquisizione del piano per le repliche di Aurora
- Risoluzione dei problemi

Prerequisiti

Attiva **`capture_plan_baselines parameter`** in una replica di Aurora: imposta il parametro `capture_plan_baselines` su automatico o manuale per acquisire i piani nelle repliche di Aurora. Per ulteriori informazioni, consulta apg_plan_mgmt.capture_plan_baselines.

Installa l'estensione postgres_fdw: è necessario installare l'estensione foreign data wrapper `postgres_fdw` per acquisire i piani nelle repliche di Aurora. Per installare l'estensione, esegui il comando seguente in ogni database.

```
postgres=> CREATE EXTENSION IF NOT EXISTS postgres_fdw;
```

Gestione dell'acquisizione del piano per le repliche di Aurora

Attiva l'acquisizione del piano per le repliche di Aurora

Per creare o rimuovere l'acquisizione del piano nelle repliche Aurora devi disporre dei privilegi di `rds_superuser`. Per ulteriori informazioni sui ruoli e le autorizzazioni degli utenti, consulta Informazioni su ruoli e autorizzazioni di PostgreSQL.

Per acquisire i piani, chiama la funzione apg_plan_mgmt.create_replica_plan_capture nell'istanza database di scrittura, come illustrato di seguito:

```
postgres=> CALL apg_plan_mgmt.create_replica_plan_capture('endpoint', 'password');
```

- endpoint - Aurora Global Database writer endpoint o cluster_endpoint fornisce il supporto di failover per Plan Capture nelle repliche Aurora.

  Per ulteriori informazioni sull'endpoint Aurora Global Database Writer, vedere. Visualizzazione degli endpoint di un database globale Amazon Aurora

  Per ulteriori informazioni sugli endpoint del cluster, consulta. Endpoint del cluster per Amazon Aurora
- password: per migliorare la sicurezza, ti consigliamo di seguire queste linee guida durante la creazione della password:
  - Deve contenere almeno 8 caratteri.
  - Deve contenere almeno una lettera maiuscola, una lettera minuscola e un numero.
  - Deve contenere almeno un carattere speciale (?, !, #, <, >, *, eccetera).

> **Note**
>
> Se si modificano l'endpoint, la password o il numero di porta, è necessario eseguire `apg_plan_mgmt.create_replica_plan_capture()` nuovamente l'operazione con l'endpoint e la password per reinizializzare l'acquisizione del piano. In caso contrario, l'acquisizione dei piani dalle repliche Aurora genererà un errore.

Disattiva l'acquisizione del piano per le repliche Aurora

È possibile disattivare il parametro `capture_plan_baselines` nelle repliche Aurora impostandone il valore su `off` nel gruppo Parametri.

Rimuovi l'acquisizione del piano per le repliche Aurora

È possibile rimuovere completamente l'acquisizione dei piani per le repliche Aurora, ma prima di farlo ti consigliamo di rifletterci bene. Per rimuovere l'acquisizione dei piani, chiama `apg_plan_mgmt.remove_replica_plan_capture` come mostrato:

```
postgres=> CALL apg_plan_mgmt.remove_replica_plan_capture();
```

È necessario chiamare nuovamente apg_plan_mgmt.create_replica_plan_capture () per attivare l'acquisizione del piano in Aurora Replicas con l'endpoint e la password.

Risoluzione dei problemi

Di seguito, è possibile trovare idee per la risoluzione dei problemi e soluzioni alternative se il piano non viene acquisito come previsto nelle repliche Aurora.

- Impostazioni dei parametri: controlla se il parametro `capture_plan_baselines` è impostato sul valore corretto per attivare l'acquisizione del piano.
- L'estensione **`postgres_fdw`** è installata: utilizza la seguente query per verificare se `postgres_fdw` è installata.

  ```
  postgres=> SELECT * FROM pg_extension WHERE extname = 'postgres_fdw'
  ```

- create_replica_plan_capture() è stata chiamata: utilizza il seguente comando per verificare se la mappatura dell'utente è presente. Altrimenti, chiama `create_replica_plan_capture()` per inizializzare la funzionalità.

```
postgres=> SELECT * FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- Numero di endpoint e porta: verificate se l'endpoint e il numero di porta sono appropriati. Quando questi valori sono errati, non viene visualizzato alcun messaggio di errore.

  Utilizza il seguente comando per verificare se l'endpoint è utilizzato in create() e per controllare in quale database risiede:

```
postgres=> SELECT srvoptions FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- reload(): è necessario chiamare apg_plan_mgmt.reload() dopo aver chiamato apg_plan_mgmt.delete_plan() nelle repliche Aurora per rendere effettiva la funzione di eliminazione. Ciò garantisce che la modifica sia stata implementata con successo.
- Password: è necessario inserire la password in create_replica_plan_capture() seguendo le linee guida menzionate. In caso contrario, verrà restituito un errore. Per ulteriori informazioni, consulta Gestione dell'acquisizione del piano per le repliche di Aurora. Utilizza un'altra password che soddisfi i requisiti.
- Connessione tra più regioni: l'acquisizione dei piani nelle repliche di Aurora è supportata anche nel database globale Aurora, dove l'istanza di scrittura e le repliche Aurora possono trovarsi in regioni diverse. Assicurati di utilizzare l'endpoint Aurora Global Database writer per mantenere la connettività dopo eventi di failover o switchover. Per ulteriori informazioni sugli endpoint di Aurora Global Database, consulta Visualizzazione degli endpoint di un database globale di Amazon Aurora. L'istanza di scrittura e la replica che si trovano in regioni diverse devono essere in grado di comunicare utilizzando il Peering VPC. Per ulteriori informazioni, consulta Peering VPC. Se si verifica un failover che coinvolge più regioni, è necessario riconfigurare l'endpoint su un nuovo endpoint primario del cluster di database.

> **Note**
>
> Quando si utilizza un endpoint del cluster anziché un endpoint Aurora Global Database writer, è necessario aggiornare l'endpoint del cluster dopo aver eseguito un'operazione di failover globale o di switchover.

Supporto della partizione di tabelle

Aurora Postgre SQL Query Plan Management (QPM) supporta il partizionamento dichiarativo delle tabelle nelle seguenti versioni:

- 15.3 o versioni successive alla 15
- 14.8 o versioni successive alla 14
- 13.11 o versioni successive alla 13

Per ulteriori informazioni, consulta la pagina relativa al partizionamento delle tabelle.

Argomenti

- Configurazione della partizione delle tabelle
- Acquisizione dei piani per la partizione delle tabelle
- Applicazione di un piano di partizione delle tabelle
- Convenzione di denominazione

Configurazione della partizione delle tabelle

Per configurare la partizione della tabella in Aurora SQL QPM Postgre, procedi come segue:

1. Imposta `apg_plan_mgmt.plan_hash_version` su 3 o un valore maggiore nel gruppo di parametri del cluster database.
2. Accedi a un database che utilizza la funzionalità di gestione dei piani di query e che contiene voci nella vista `apg_plan_mgmt.dba_plans`.
3. Chiama `apg_plan_mgmt.validate_plans('update_plan_hash')` per aggiornare il valore `plan_hash` nella tabella dei piani.
4. Ripeti i passaggi 2-3 per tutti i database con la funzionalità di gestione dei piani di query abilitata contenenti voci nella vista `apg_plan_mgmt.dba_plans`.

Per ulteriori informazioni su questi parametri, consulta Documentazione di riferimento dei parametri per la gestione del piano di query Aurora PostgreSQL.

Acquisizione dei piani per la partizione delle tabelle

NelQPM, i diversi piani si distinguono per il loro valore. `plan_hash` Per capire come il valore `plan_hash` cambia, devi prima familiarizzare con un tipo simile di piani.

La combinazione di metodi di accesso, nomi di indice con rimozione di cifre e nomi di partizione con rimozione di cifre, accumulati a livello di nodo Append deve essere costante affinché i piani siano considerati uguali. Le partizioni specifiche a cui si accede nei piani non sono significative. Nell'esempio seguente, viene creata una tabella `tbl_a` con 4 partizioni.

```
postgres=>create table tbl_a(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table tbl_a1 partition of tbl_a for values from (0) to (1000);
CREATE TABLE
postgres=>create table tbl_a2 partition of tbl_a for values from (1001) to (2000);
CREATE TABLE
postgres=>create table tbl_a3 partition of tbl_a for values from (2001) to (3000);
CREATE TABLE
postgres=>create table tbl_a4 partition of tbl_a for values from (3001) to (4000);
CREATE TABLE
postgres=>create index t_i on tbl_a using btree (i);
CREATE INDEX
postgres=>create index t_j on tbl_a using btree (j);
CREATE INDEX
postgres=>create index t_k on tbl_a using btree (k);
CREATE INDEX
```

I piani seguenti sono considerati uguali perché viene utilizzato un unico metodo di scansione di `tbl_a` indipendentemente dal numero di partizioni cercate dalla query.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 999 and j < 9910 and k > 50;

                        QUERY PLAN
-----------------------------------------------------------------------
Seq Scan on tbl_a1 tbl_a
    Filter: ((i >= 990) AND (i <= 999) AND (j < 9910) AND (k > 50))
SQL Hash: 1553185667, Plan Hash: -694232056
(3 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;
```

```
                        QUERY PLAN
------------------------------------------------------------------
Append
    ->  Seq Scan on tbl_a1 tbl_a_1
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    ->  Seq Scan on tbl_a2 tbl_a_2
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    SQL Hash: 1553185667, Plan Hash: -694232056
    (6 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(8 rows)
```

Anche i seguenti 3 piani sono considerati uguali perché, a livello padre, i metodi di accesso, i nomi degli indici con rimozione di cifre e i nomi delle partizioni con rimozione di cifre sono `SeqScan tbl_a`, `IndexScan (i_idx) tbl_a`.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                               QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_i_idx on tbl_a2 tbl_a_2
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
```

```
(7 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(11 rows)
```

Indipendentemente dalla diversità a livello di ordine e numero di occorrenze nelle partizioni secondarie, i metodi di accesso, i nomi degli indici con rimozione di cifre e i nomi delle partizioni con rimozione di cifre sono costanti a livello padre per ciascuno dei piani precedenti.

Tuttavia, i piani sarebbero considerati diversi se fosse soddisfatta una delle seguenti condizioni:

- Nel piano viene utilizzato qualsiasi metodo di accesso aggiuntivo.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 1134525070
(11 rows)
```

- Nessuno dei metodi di accesso del piano non viene più utilizzato.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(6 rows)
```

- L'indice associato a un metodo di indice viene modificato.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                               QUERY PLAN
--------------------------------------------------------------------------
 Append
```

```
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_j_idx on tbl_a2 tbl_a_2
         Index Cond: (j < 9910)
         Filter: ((i >= 990) AND (i <= 1100) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993343726
(7 rows)
```

Applicazione di un piano di partizione delle tabelle

I piani approvati per le tabelle partizionate vengono applicati con la corrispondenza a livello di posizione. I piani non sono specifici delle partizioni e possono essere applicati a partizioni diverse dai piani a cui si fa riferimento nella query originale. I piani possono essere applicati anche per le richieste che accedono a un numero di partizioni diverso rispetto alla struttura originale approvata.

Ad esempio, se la struttura approvata si riferisce al seguente piano:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

Quindi, questo piano può essere applicato anche SQL alle query che fanno riferimento a 2, 4 o più partizioni. I possibili piani che potrebbero derivare da questi scenari per l'accesso a 2 e 4 partizioni sono:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;
```

```
                                QUERY PLAN
-------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(8 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a4 tbl_a_4
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(12 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
```

```
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(14 rows)
```

Prendi in considerazione un altro piano approvato con metodi di accesso diversi per ogni partizione:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 2032136998
(12 rows)
```

In questo caso, qualsiasi piano che esegue letture da due partizioni non viene applicato. A meno che tutte le combinazioni (metodo di accesso, nome di indice) del piano approvato non siano utilizzabili, il piano non può essere applicato. Ad esempio, i seguenti piani hanno hash diversi e il piano approvato non può essere applicato in questi casi:

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                             QUERY PLAN
-------------------------------------------------------------------------
 Append
```

```
   ->  Bitmap Heap Scan on tbl_a1 tbl_a_1
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a1_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
   ->  Bitmap Heap Scan on tbl_a2 tbl_a_2
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a2_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
  Note: This is not an Approved plan.  No usable Approved plan was found.
  SQL Hash: 1553185667, Plan Hash: -568647260
(13 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1900) AND (j < 9910) AND (k > 50))
 Note: This is not an Approved plan.  No usable Approved plan was found.
 SQL Hash: 1553185667, Plan Hash: -496793743
(8 rows)
```

Convenzione di denominazione

QPMPer applicare un piano con tabelle partizionate dichiarative, è necessario seguire regole di denominazione specifiche per le tabelle principali, le partizioni di tabella e gli indici:

- Nomi delle tabelle principali: questi nomi devono differire in base all'alfabeto o ai caratteri speciali e non solo in base alle cifre. Ad esempio, tA, tB e tC sono nomi accettabili per tabelle padre distinte, mentre t1, t2 e t3 non lo sono.
- Nomi delle tabelle delle partizioni individuali: le partizioni dello stesso elemento principale devono differire l'una dall'altra solo in cifre. Ad esempio, i nomi di partizione accettabili per tA potrebbero essere tA1, tA2 o T1a, T2a o anche un numero maggiore di cifre.

  Eventuali altre differenze (lettere, caratteri speciali) non garantiranno l'applicazione del piano.

- Nomi degli indici: nella gerarchia delle tabelle delle partizioni, assicurati che tutti gli indici abbiano nomi univoci. Ciò significa che le parti non numeriche dei nomi devono essere diverse. Ad esempio, se si dispone di una tabella partizionata denominata `tA` con un indice denominato`tA_col1_idx1`, non è possibile avere un altro indice denominato. `tA_col1_idx2` Tuttavia, è possibile avere un indice chiamato `tA_a_col1_idx2` perché la parte non numerica del nome è univoca. Questa regola si applica agli indici creati sia nella tabella principale che nelle singole tabelle delle partizioni.

La mancata conformità alle convenzioni di denominazione sopra citate può comportare la mancata applicazione dei piani approvati. L'esempio seguente illustra tale mancata applicazione:

```
postgres=>create table t1(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table t1a partition of t1 for values from (0) to (1000);
CREATE TABLE
postgres=>create table t1b partition of t1 for values from (1001) to (2000);
CREATE TABLE
postgres=>SET apg_plan_mgmt.capture_plan_baselines TO 'manual';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 0;

                            QUERY PLAN
------------------------------------------------------------------------------
 Aggregate
   ->  Append
         ->  Seq Scan on t1a t1_1
               Filter: (i > 0)
         ->  Seq Scan on t1b t1_2
               Filter: (i > 0)
 SQL Hash: -1720232281, Plan Hash: -1010664377
(7 rows)
```

```
postgres=>SET apg_plan_mgmt.use_plan_baselines TO 'on';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 1000;

                            QUERY PLAN
------------------------------------------------------------------------------
 Aggregate
   ->  Seq Scan on t1b t1
         Filter: (i > 1000)
```

```
 Note: This is not an Approved plan. No usable Approved plan was found.
 SQL Hash: -1720232281, Plan Hash: 335531806
(5 rows)
```

Anche se i due piani possono apparire identici, `Plan Hash` i loro valori sono diversi a causa dei nomi delle tabelle secondarie. I nomi delle tabelle variano in base ai caratteri alfabetici anziché alle sole cifre, il che comporta un errore di applicazione.

Utilizzo di estensioni e wrapper di dati esterni

Per estendere la funzionalità al cluster database Edizione compatibile con Aurora PostgreSQL, è possibile installare e utilizzare diverse estensioni PostgreSQL. Ad esempio, se il caso d'uso richiede l'immissione intensiva di dati su tabelle molto grandi, è possibile installare l'estensione `pg_partman` per partizionare i dati e quindi distribuire il carico di lavoro.

> **Note**
>
> La funzionalità Trusted Language Extensions per PostgreSQL è supportata da Aurora PostgreSQL a partire dalla versione 14.5. Viene implementata come estensione `pg_tle` che puoi aggiungere ad Aurora PostgreSQL e che può essere utilizzata dagli sviluppatori per creare le proprie estensioni di PostgreSQL in un ambiente sicuro che riduce i requisiti di impostazione e configurazione, nonché gran parte dei test preliminari delle nuove estensioni. Per ulteriori informazioni, consulta Utilizzo di Trusted Language Extensions per PostgreSQL.

In alcuni casi, anziché installare un'estensione, è possibile aggiungere un modulo specifico all'elenco di `shared_preload_libraries` nel gruppo di parametri cluster database personalizzato del cluster database Aurora PostgreSQL. In genere, il gruppo di parametri cluster di database predefinito carica solo `pg_stat_statements`, ma sono disponibili diversi altri moduli da aggiungere all'elenco. Ad esempio, è possibile aggiungere funzionalità di pianificazione aggiungendo il modulo `pg_cron`, come descritto in Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron. Come altro esempio, è possibile registrare i piani di esecuzione delle query caricando il modulo `auto_explain`. Per ulteriori informazioni, consultare Logging execution plans of queries (Registrazione dei piani di esecuzione delle query) nel Knowledge Center di AWS.

Un'estensione che fornisce accesso a dati esterni è più specificatamente conosciuta come un wrapper di dati esterno (FDW). Ad esempio, l'estensione `oracle_fdw` consente al cluster database Aurora PostgreSQL di utilizzare i database Oracle.

È inoltre possibile specificare con precisione le estensioni che possono essere installate sull'istanza database Aurora PostgreSQL, elencandole nel parametro `rds.allowed_extensions`. Per ulteriori informazioni, consultaLimitazione dell'installazione delle estensioni PostgreSQL.

Di seguito sono disponibili informazioni sulla configurazione e l'utilizzo di alcune delle estensioni, dei moduli e degli FDW disponibili per Aurora PostgreSQL. Per semplicità, queste sono tutte denominate "estensioni". Per gli elenchi delle estensioni che è possibile utilizzare con le versioni di Aurora

PostgreSQL attualmente disponibili, consultare la pagina relativa alle versioni delle estensioni per Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

- Gestione di oggetti di grandi dimensioni con il modulo lo
- Gestione dei dati spaziali con l'estensione Post GIS
- Gestione delle partizioni PostgreSQL con l'estensione pg_partman
- Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron
- Utilizzo pgAudit per registrare l'attività del database
- Utilizzo di pglogical per sincronizzare i dati tra le istanze
- Interazione con un database Oracle utilizzando l'estensione oracle_fdw
- Lavorare con i database SQL Server utilizzando l'estensione tds_fdw

Utilizzo del supporto delegato delle estensioni di Amazon Aurora per Postgre SQL

Utilizzando il supporto delle estensioni delegate di Amazon Aurora per PostgreSQL, puoi delegare la gestione delle estensioni a un utente che non deve essere un. `rds_superuser` Con questo supporto delegato per le estensioni, `rds_extension` viene creato un nuovo ruolo chiamato che devi assegnare a un utente per gestire altre estensioni. Questo ruolo può creare, aggiornare e eliminare estensioni.

È possibile specificare le estensioni che possono essere installate sull'istanza SQL DB di Aurora Postgre, elencandole nel parametro. `rds.allowed_extensions` Per ulteriori informazioni, consulta Usare le SQL estensioni Postgre con Amazon RDS for Postgre. SQL

Puoi limitare l'elenco delle estensioni disponibili che possono essere gestite dall'utente con il ruolo utilizzando il parametro. `rds_extension` `rds.allowed_delegated_extensions`

Il supporto per le estensioni delegate è disponibile nelle seguenti versioni:

- Tutte le versioni successive
- 15.5 e versioni successive 15
- 14.10 e versioni successive 14
- 13.13 e versioni successive 13
- 12.17 e versioni successive 12

Argomenti

- Attivazione del supporto per le estensioni delegate a un utente
- Configurazione utilizzata nel supporto delle estensioni delegate Aurora per Postgre SQL
- Disattivazione del supporto per l'estensione delegata
- Vantaggi dell'utilizzo del supporto per le estensioni delegate di Amazon Aurora
- Limitazione del supporto delle estensioni delegate Aurora per Postgre SQL
- Autorizzazioni necessarie per determinate estensioni
- Considerazioni sulla sicurezza
- Drop Extension Cascade è disabilitato
- Esempi di estensioni che possono essere aggiunte utilizzando il supporto di estensioni delegate

Attivazione del supporto per le estensioni delegate a un utente

È necessario eseguire le seguenti operazioni per abilitare il supporto delle estensioni delegate a un utente:

1. Concedi **`rds_extension`** il ruolo a un utente: connettiti al database come `rds_superuser` ed esegui il seguente comando:

   ```
   Postgres => grant rds_extension to user_name;
   ```

2. Imposta l'elenco delle estensioni disponibili per la gestione degli utenti delegati: `rds.allowed_delegated_extensions` consente di specificare un sottoinsieme delle estensioni disponibili utilizzando `rds.allowed_extensions` il parametro del cluster DB. È possibile eseguire questa operazione a uno dei seguenti livelli:

   - Nel cluster o nel gruppo di parametri dell'istanza, tramite AWS Management Console oppureAPI. Per ulteriori informazioni, consulta .
   - Utilizzate il seguente comando a livello di database:

     ```
     alter database database_name set rds.allowed_delegated_extensions =
      'extension_name_1,
                        extension_name_2,...extension_name_n';
     ```

   - Utilizzate il seguente comando a livello di utente:

     ```
     alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
     ```

```
extension_name_2,...extension_name_n';
```

> **Note**
>
> Non è necessario riavviare il database dopo aver modificato il parametro `rds.allowed_delegated_extensions` dinamico.

3. Consenti l'accesso all'utente delegato agli oggetti creati durante il processo di creazione dell'estensione: alcune estensioni creano oggetti che richiedono la concessione di autorizzazioni aggiuntive prima che l'utente con `rds_extension` ruolo possa accedervi. `rds_superuser`Devono concedere all'utente delegato l'accesso a tali oggetti. Una delle opzioni consiste nell'utilizzare un trigger di evento per concedere automaticamente l'autorizzazione all'utente delegato. Per ulteriori informazioni, consulta l'esempio di attivazione di un evento inDisattivazione del supporto per l'estensione delegata.

Configurazione utilizzata nel supporto delle estensioni delegate Aurora per Postgre SQL

Nome di configurazione	Descrizione	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
`rds.allowed_delegated_extensions`	Questo parametro limita le estensioni che un ruolo rds_extension può gestire in un database. Deve essere un sottoinsieme di rds.allowed_extensions.	stringa vuota	• Per impostazione predefinita, questo parametro è una stringa vuota, il che significa che nessuna estensione è stata delegata agli utenti con`rds_extension` .	rds_superuser

Nome di configurazione	Descrizione	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
			• È possibile aggiungere qualsiasi estensione supportata se l'utente dispone dell'autorizzazione necessaria. A tale scopo, impostate il `rds.allowed_delegated_extensions` parametro su una stringa di nomi di estensione separati da virgole. Aggiungendo un elenco di estensioni a questo parametro, identificate esplicitamente le estensioni che l'utente con il `rds_extension` ruolo può installare. • Se impostato su`*`, significa che	

Nome di configurazione	Descrizione	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
			tutte le estensioni elencate in `rds_allowed_extensions` sono delegate a utenti con `rds_extension` ruolo. Per ulteriori informazioni sulla configurazione di questo parametro, consulta Attivazione del supporto per le estensioni delegate a un utente.	

Nome di configurazione	Descrizione	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
`rds.allowed_extensions`	Questo parametro consente al cliente di limitare le estensioni che possono essere installate nell'istanza DB Aurora SQL Postgre. Per ulteriori informazioni, consulta Limitazione dell'installazione delle estensioni Postgre SQL	"*"	Per impostazione predefinita, questo parametro è impostato su «*», il che significa che tutte le estensioni supportate su RDS Postgre e SQL Aurora Postgre possono essere create da utenti con SQL i privilegi necessari. Vuoto significa che nessuna estensione può essere installata nell'istanza DB di Aurora SQL Postgre.	amministratore

Nome di configurazione	Descrizione	Valore predefinito	Note	Chi può modificare o concedere l'autorizzazione
`rds-delegated_extension_allow_drascac`	Questo parametro controlla la possibilità per l'utente di eliminare l'estensione utilizzando un'opzione `rds_extension` a cascata.	off	Per impostazione predefinita, `rds-delegated_extension_allow_drop_cascade` è impostato su `off`. Ciò significa che gli utenti con non `rds_extension` sono autorizzati a eliminare un'estensione utilizzando l'opzione a cascata. Per garantire tale capacità, il `rds.delegated_extension_allow_drop_cascade` parametro deve essere impostato su. on	rds_superuser

Disattivazione del supporto per l'estensione delegata

Spegnimento parziale

Gli utenti delegati non possono creare nuove estensioni ma possono comunque aggiornare le estensioni esistenti.

- Ripristina `rds.allowed_delegated_extensions` il valore predefinito nel gruppo di parametri del cluster DB.
- Utilizzate il seguente comando a livello di database:

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- Utilizzate il seguente comando a livello di utente:

```
alter user user_name reset rds.allowed_delegated_extensions;
```

Spegnimento completo

La revoca del `rds_extension` ruolo a un utente ripristinerà le autorizzazioni standard per l'utente. L'utente non può più creare, aggiornare o eliminare le estensioni.

```
postgres => revoke rds_extension from user_name;
```

Esempio di attivazione di un evento

Se desideri consentire a un utente delegato di utilizzare estensioni che richiedono l'impostazione delle autorizzazioni sugli oggetti creati durante la creazione dell'estensione, puoi personalizzare l'esempio seguente di attivazione di un evento e aggiungere solo le estensioni per le quali desideri che gli utenti delegati abbiano accesso alla piena funzionalità. `rds_extension` Questo trigger di evento può essere creato su template1 (il modello predefinito), pertanto tutti i database creati da template1 avranno quel trigger di evento. Quando un utente delegato installa l'estensione, questo trigger garantirà automaticamente la proprietà degli oggetti creati dall'estensione.

```

CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE

  schemaname TEXT;
  databaseowner TEXT;

  r RECORD;

BEGIN
```

```
 IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
   RAISE NOTICE 'SECURITY INVOKER';
   RAISE NOTICE 'user: %', current_user;
   FOR r IN SELECT * FROM pg_event_trigger_ddl_commands()
   LOOP
       CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
'extension';

       schemaname = (
           SELECT n.nspname
           FROM pg_catalog.pg_extension AS e
           INNER JOIN pg_catalog.pg_namespace AS n
           ON e.extnamespace = n.oid
           WHERE e.oid = r.objid
       );

       databaseowner = (
           SELECT pg_catalog.pg_get_userbyid(d.datdba)
           FROM pg_catalog.pg_database d
           WHERE d.datname = current_database()
       );
       RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
schemaname, databaseowner;
       IF r.object_identity = 'address_standardizer_data_us' THEN
           EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE %I.us_gaz TO
%I WITH GRANT OPTION;', schemaname, databaseowner);
           EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE %I.us_lex TO
%I WITH GRANT OPTION;', schemaname, databaseowner);
           EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE %I.us_rules
TO %I WITH GRANT OPTION;', schemaname, databaseowner);
       ELSIF r.object_identity = 'dict_int' THEN
           EXECUTE format('ALTER TEXT SEARCH DICTIONARY %I.intdict OWNER TO %I;',
schemaname, databaseowner);
       ELSIF r.object_identity = 'pg_partman' THEN
           EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE
%I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
           EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE
%I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
           EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE
%I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname, databaseowner);
       ELSIF r.object_identity = 'postgis_topology' THEN
```

```
            EXECUTE format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL TABLES IN
 SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE format('GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA topology TO
 %I WITH GRANT OPTION;', databaseowner);
            EXECUTE format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA topology TO %I
 WITH GRANT OPTION;', databaseowner);
            EXECUTE format('GRANT USAGE ON SCHEMA topology TO %I WITH GRANT OPTION;',
 databaseowner);
        END IF;
    END LOOP;
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE create_ext();
```

Vantaggi dell'utilizzo del supporto per le estensioni delegate di Amazon Aurora

Utilizzando il supporto delegato delle estensioni di Amazon Aurora per PostgreSQL, deleghi in modo sicuro la gestione delle estensioni a utenti che non ricoprono il ruolo. `rds_superuser` Questa funzionalità offre i seguenti vantaggi:

- Puoi delegare facilmente la gestione delle estensioni a utenti di tua scelta.
- Questo non richiede un `rds_superuser` ruolo.
- Offre la possibilità di supportare diversi set di estensioni per diversi database nello stesso cluster DB.

Limitazione del supporto delle estensioni delegate Aurora per Postgre SQL

- Gli oggetti creati durante il processo di creazione dell'estensione possono richiedere privilegi aggiuntivi per il corretto funzionamento dell'estensione.

Autorizzazioni necessarie per determinate estensioni

Per creare, utilizzare o aggiornare le seguenti estensioni, l'utente delegato deve disporre dei privilegi necessari sulle seguenti funzioni, tabelle e schemi.

Ester i che richie o propr o autor zioni	Funzione	Tabelle	Schema	Dizionario di ricerca testuale	Commento
addre tanda er_da s		us_gaz, us_lex, us_lex, i.US_Rules			
amch	bt_index_check, bt_index_ parent_check				
dict_i				intrezzo	
pg_p n		part_time _partitions, part_config, part_config_sub			
pg_s tatem					
Posta GIS	st_tileenvelope	spatial_ref_sys			
postg aster					
postg opolo		topologia, livello	topologia		l'utente delegato Deve essere il

Ester i che richie o propr o autor zioni	Funzione	Tabelle	Schema	Dizionario di ricerca testuale	Commento
					proprietario del database
log_f	create_fo reign_tab le_for_log_file				
rds_t	tipo_ruol o_passwor d_encryption_				
postg iger_ oder		geocode_s ettings_default, geocode_s ettings	tigre		
pg_fr acem	pg_freespace				
pg_vi lity	pg_visibility				

Considerazioni sulla sicurezza

Tieni presente che un utente con `rds_extension` ruolo sarà in grado di gestire le estensioni su tutti i database su cui ha il privilegio di connessione. Se l'intenzione è fare in modo che un utente delegato gestisca l'estensione su un singolo database, è buona norma revocare tutti i privilegi dal pubblico

su ciascun database, quindi concedere esplicitamente il privilegio di connessione per quel database specifico all'utente delegato.

Esistono diverse estensioni che possono consentire a un utente di accedere alle informazioni da più database. Assicurati che gli utenti `rds_extension` concessi dispongano di funzionalità per più database prima di aggiungere queste estensioni`rds.allowed_delegated_extensions`. Ad esempio, `postgres_fdw` `dblink` fornisci funzionalità per eseguire query su database sulla stessa istanza o su istanze remote. `log_fdw`legge i file di registro del motore Postgres, che riguardano tutti i database dell'istanza, potenzialmente contenenti query lente o messaggi di errore provenienti da più database. `pg_cron`consente l'esecuzione di processi pianificati in background sull'istanza DB e può configurare i lavori per l'esecuzione in un database diverso.

Drop Extension Cascade è disabilitato

La possibilità di eliminare l'estensione con l'opzione a cascata da un utente con il `rds_extension` ruolo è controllata da un `rds.delegated_extension_allow_drop_cascade` parametro. Per impostazione predefinita, `rds-delegated_extension_allow_drop_cascade` è impostato su `off`. Ciò significa che agli utenti con il `rds_extension` ruolo non è consentito eliminare un'estensione utilizzando l'opzione a cascata, come mostrato nella query seguente.

```
DROP EXTENSION CASCADE;
```

In questo modo verranno eliminati automaticamente gli oggetti che dipendono dall'estensione e, a loro volta, tutti gli oggetti che dipendono da tali oggetti. Il tentativo di utilizzare l'opzione a cascata genererà un errore.

Per garantire tale capacità, il `rds.delegated_extension_allow_drop_cascade` parametro deve essere impostato su. on

La modifica del parametro `rds.delegated_extension_allow_drop_cascade` dinamico non richiede il riavvio del database. È possibile eseguire questa operazione a uno dei seguenti livelli:

- Nel cluster o nel gruppo di parametri dell'istanza, tramite AWS Management Console oppureAPI.
- Utilizzando il seguente comando a livello di database:

  ```
  alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
  ```

- Utilizzando il seguente comando a livello di utente:

```
alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
```

Esempi di estensioni che possono essere aggiunte utilizzando il supporto di estensioni delegate

- `rds_tools`

```
extension_test_db=> create extension rds_tools;
CREATE EXTENSION
extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
 rolname = 'pg_read_server_files';
ERROR: permission denied for function role_password_encryption_type
```

- `amcheck`

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- `pg_freespacemap`

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- `pg_visibility`

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- `postgres_fdw`

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

Gestione di oggetti di grandi dimensioni con il modulo lo

Il modulo lo (estensione) è destinato agli utenti e agli sviluppatori di database che utilizzano i database Postgre tramite i nostri driver. SQL JDBC ODBC Entrambi JDBC ODBC si aspettano che il database gestisca la cancellazione di oggetti di grandi dimensioni quando i riferimenti ad essi cambiano. Tuttavia, Postgre SQL non funziona in questo modo. Postgre SQL non presume che un oggetto debba essere eliminato quando il suo riferimento cambia. Il risultato è che gli oggetti rimangono su disco, senza riferimenti. L'estensione lo include una funzione utilizzata per attivare le modifiche dei riferimenti per eliminare gli oggetti se necessario.

> **Tip**
>
> Per determinare se il database può trarre vantaggio dall'estensione lo, utilizzate l'`vacuumlo`utilità per verificare la presenza di oggetti di grandi dimensioni orfani. Per ottenere il conteggio di oggetti orfani di grandi dimensioni senza eseguire alcuna azione, esegui l'utilità con l'opzione `-n`(no-op). Per scoprire come, consulta vacuumlo utility seguente.

Il modulo lo è disponibile per Aurora Postgree SQL 13.7, 12.11, 11.16, 10.21 e versioni secondarie successive.

Per installare il modulo (estensione), sono necessari privilegi `rds_superuser`. L'installazione dell'estensione lo aggiunge quanto segue al database:

- `lo`— Si tratta di un tipo di dati di grandi dimensioni (lo) che è possibile utilizzare per oggetti binari di grandi dimensioni () e altri oggetti di grandi dimensioni. BLOBs Il tipo di dati `lo` è un dominio del tipo di dati `oid`. In altre parole, è un identificatore di oggetti con vincoli opzionali. Per ulteriori informazioni, consulta Identificatori di oggetti nella documentazione di SQL Postgre. In termini semplici, è possibile utilizzare il tipo di `lo` dati per distinguere le colonne del database che contengono riferimenti a oggetti di grandi dimensioni da altri identificatori di oggetti (). OIDs
- `lo_manage` – Questa è una funzione che puoi utilizzare nei trigger sulle colonne di tabella contenenti riferimenti di oggetti di grandi dimensioni. Ogni volta che elimini o modifichi un valore che fa riferimento a un oggetto di grandi dimensioni, il trigger scollega l'oggetto (`lo_unlink`) dal suo riferimento. Utilizza il trigger su una colonna solo se la colonna è l'unico riferimento del database all'oggetto di grandi dimensioni.

Per ulteriori informazioni sul modulo large objects, consultate lo nella documentazione di SQL Postgree.

Installazione dell'estensione lo

Prima di installare l'estensione lo, assicurati di disporre dei privilegi `rds_superuser`.

Installare l'estensione

1. `psql`Utilizzalo per connetterti all'istanza DB principale del tuo cluster DB Aurora SQL Postgre.

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Specifica la password, quando richiesto. Il client `psql` si connette e visualizza il database di connessione amministrativa predefinito `postgres=>` come prompt.

2. Installa l'estensione come segue:

   ```
   postgres=> CREATE EXTENSION lo;
   CREATE EXTENSION
   ```

Ora puoi utilizzare il tipo di dati `lo` per definire le colonne nelle tabelle. Ad esempio, puoi creare una tabella (`images`) contenente dati immagine raster. Puoi utilizzare il tipo di dati `lo` per una colonna `raster`, come mostrato nell'esempio seguente, che crea una tabella.

```
postgres=> CREATE TABLE images (image_name text, raster lo);
```

Utilizzo della funzione di trigger lo_manage per eliminare gli oggetti

Puoi utilizzare la funzione `lo_manage` in un trigger su un `lo` o altre colonne di oggetti di grandi dimensioni per rimuovere (e prevenire oggetti orfani) quando `lo` viene aggiornato o eliminato.

Impostare trigger su colonne che fanno riferimento a oggetti di grandi dimensioni

- Esegui una di queste operazioni:
  - Crea un DELETE trigger BEFORE UPDATE OR su ogni colonna per contenere riferimenti univoci a oggetti di grandi dimensioni, utilizzando il nome della colonna per l'argomento.

    ```
    postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OR DELETE ON images
        FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
    ```
  - Applica un trigger solo quando la colonna viene aggiornata.

    ```
    postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OF images
        FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
    ```

La funzione di trigger `lo_manage` funziona solo nel contesto dell'inserimento o dell'eliminazione dei dati di colonna, a seconda di come definisci il trigger. Non ha alcun effetto quando esegui un'operazione `DROP` o `TRUNCATE` su un database. Ciò significa che devi eliminare le colonne di oggetti da qualsiasi tabella prima del rilascio, per evitare la creazione di oggetti orfani.

Ad esempio, supponi di voler rilasciare il database contenente la tabella `images`. Per eliminare la colonna procedi come segue.

```
postgres=> DELETE FROM images COLUMN raster
```

Ipotizzando che la funzione `lo_manage` sia definita su tale colonna per gestire le eliminazioni, ora puoi rilasciare la tabella in modo sicuro.

Rimozione di oggetti di grandi dimensioni orfani utilizzando **`vacuumlo`**

L'`vacuumlo`outilità identifica e rimuove oggetti di grandi dimensioni orfani dai database. Questa utilità è disponibile a partire da SQL Postgree 9.1.24. Se gli utenti del database lavorano abitualmente con

oggetti di grandi dimensioni, consigliamo di eseguirli di tanto in `vacuumlo` tanto per ripulire oggetti di grandi dimensioni orfani.

Prima di installare l'estensione log, puoi utilizzarla `vacuumlo` per valutare se il tuo cluster Aurora Postgree SQL DB può trarne vantaggio. A questo scopo, esegui `vacuumlo` con l'opzione `-n` (no-op) per mostrare cosa viene rimosso, come illustrato di seguito:

```
$ vacuumlo -v -n -h your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com -
p 5433 -U postgres docs-lab-spatial-db
Password:*****
Connected to database "docs-lab-spatial-db"
Test run: no large objects will be removed!
Would remove 0 large objects from database "docs-lab-spatial-db".
```

Come mostra l'output, gli oggetti orfani di grandi dimensioni non sono un problema per questo particolare database.

Per ulteriori informazioni su questa utilità, consulta la documentazione di `vacuumlo`Postgre. SQL

Introduzione al funzionamento di **`vacuumlo`**

Il `vacuumlo` comando rimuove gli oggetti di grandi dimensioni orfani (LOs) dal SQL database Postgre senza influire o entrare in conflitto con le tabelle utente.

Il comando funziona nel modo seguente:

1. `vacuumlo`inizia creando una tabella temporanea contenente tutti gli Object IDs (OIDs) degli oggetti di grandi dimensioni presenti nel database.
2. `vacuumlo`quindi analizza ogni colonna del database che utilizza i tipi di dati `oid` `olo`. Se `vacuumlo` trova una corrispondenza OID in queste colonne, la rimuove OID dalla tabella temporanea. `vacuumlo`controlla solo le colonne con nome specifico `oid` `olo`, non i domini basati su questi tipi.
3. Le voci rimanenti nella tabella temporanea sono quelle orfaneLOs, che vengono `vacuumlo` quindi rimosse in modo sicuro.

Miglioramento delle prestazioni **`vacuumlo`**

È possibile migliorare le prestazioni di `vacuumlo` aumentando la dimensione del batch utilizzando l'`-l`opzione. Ciò consente `vacuumlo` di elaborarne più LOs contemporaneamente.

Se il sistema dispone di memoria sufficiente ed è possibile inserire completamente la tabella temporanea in memoria, l'aumento dell'`temp_buffers`impostazione a livello di database può migliorare le prestazioni. Ciò consente alla tabella di risiedere interamente in memoria, il che può migliorare le prestazioni complessive.

La seguente query stima la dimensione della tabella temporanea:

```
SELECT
    pg_size_pretty(SUM(pg_column_size(oid))) estimated_lo_temp_table_size
FROM
    pg_largeobject_metadata;
```

Considerazioni relative agli oggetti di grandi dimensioni

Di seguito sono riportate alcune considerazioni importanti da tenere presente quando si lavora con oggetti di grandi dimensioni:

- `Vacuumlo`è l'unica soluzione in quanto attualmente non esiste un altro metodo per rimuovere gli orfaniLOs.
- Strumenti come la replica logica nativa e biologica e AWS DMS che utilizzano tecnologie di replica non supportano la replica di oggetti di grandi dimensioni.
- Durante la progettazione dello schema del database, evita di utilizzare oggetti di grandi dimensioni quando possibile e prendi in considerazione l'utilizzo di tipi di dati alternativi come invece. `bytea`
- Esegui `vacuumlo` regolarmente, almeno una volta alla settimana, per evitare problemi con gli orfaniLOs.
- Utilizza un trigger con la `lo_manage` funzione sulle tabelle che memorizzano oggetti di grandi dimensioni per evitare che vengano creati oggetti orfaniLOs.

Gestione dei dati spaziali con l'estensione Post GIS

Post GIS è un'estensione di Postgre SQL per l'archiviazione e la gestione delle informazioni spaziali. Per saperne di più su PostGIS, vedi Post.net. GIS

A partire dalla versione 10.5, Postgre SQL supporta la libreria libprotobuf 1.3.0 usata da Post GIS per lavorare con i dati delle tile vettoriali map box.

La configurazione dell'estensione Post richiede dei privilegi. GIS `rds_superuser` Ti consigliamo di creare un utente (ruolo) per gestire l'GISestensione Post e i tuoi dati spaziali. L'GISestensione Post e i relativi componenti aggiungono migliaia di funzioni a SQL Postgre. Prendi in considerazione la possibilità di creare GIS l'estensione Post in un proprio schema, se ciò ha senso per il tuo caso d'uso. Nell'esempio seguente viene illustrato come installare l'estensione nel proprio database, ma questa operazione non è necessaria.

Argomenti

- Passaggio 1: crea un utente (ruolo) per gestire l'GISestensione Post
- Passaggio 2: carica le estensioni Post GIS
- Fase 3: Trasferire la proprietà degli schemi di estensione
- Fase 4: Trasferire la proprietà delle GIS tabelle Post
- Passaggio 5: Verificare le estensioni
- Passaggio 6: Aggiorna l'GISestensione Post
- Versioni successive all'estensione GIS
- Aggiornamento da Post 2 a Post 3 GIS GIS

Passaggio 1: crea un utente (ruolo) per gestire l'GISestensione Post

Innanzitutto, connettiti alla tua istanza SQL DB RDS per Postgre come utente con privilegi. `rds_superuser` Se hai mantenuto il nome di default durante la configurazione dell'istanza, esegui la connessione come `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

Crea un ruolo separato (utente) per amministrare l'estensione Post. GIS

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
CREATE ROLE
```

Concedi a questo ruolo i privilegi `rds_superuser` per consentire l'installazione dell'estensione.

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

Crea un database da utilizzare per gli GIS artefatti del tuo Post. Questa fase è facoltativa. Oppure puoi creare uno schema nel tuo database utenti per le GIS estensioni Post, ma anche questo non è necessario.

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

Concedi a `gis_admin` tutti i privilegi per il database `lab_gis`.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

Esci dalla sessione e riconnettiti alla tua istanza SQL DB RDS for Postgre come. `gis_admin`

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

Continua a configurare l'estensione come descritto nei passaggi successivi.

Passaggio 2: carica le estensioni Post GIS

L'GISestensione Post include diverse estensioni correlate che collaborano per fornire funzionalità geospaziali. A seconda del caso d'uso, è possibile che le estensioni create in questo passaggio non siano tutte necessarie.

Usa `CREATE EXTENSION` le istruzioni per caricare le estensioni Post. GIS

```
CREATE EXTENSION postgis;
CREATE EXTENSION
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
CREATE EXTENSION postgis_topology;
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION
```

È possibile verificare i risultati eseguendo la SQL query mostrata nell'esempio seguente, che elenca le estensioni e i relativi proprietari.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Fase 3: Trasferire la proprietà degli schemi di estensione

Usa le ALTER SCHEMA istruzioni per trasferire la proprietà degli schemi al `gis_admin` ruolo.

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

È possibile confermare il cambio di proprietà eseguendo la seguente SQL query. Oppure è possibile utilizzare il meta-comando \dn dalla riga di comando psql.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;

       List of schemas
     Name     |     Owner
--------------+---------------
```

```
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

Fase 4: Trasferire la proprietà delle GIS tabelle Post

> **Note**
>
> Non modificate la proprietà delle GIS funzioni Post. Il corretto funzionamento e i futuri aggiornamenti di Post GIS richiedono che queste funzioni mantengano la proprietà originale. Per ulteriori informazioni sulle GIS autorizzazioni Post, consulta SQLPostgree Security.

Usa la seguente funzione per trasferire la proprietà delle GIS tabelle Post al ruolo. `gis_admin` Eseguire la seguente istruzione dal prompt di psql per creare la funzione.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

Successivamente, eseguire la seguente query per eseguire la funzione `exec` che a sua volta esegue le istruzioni e altera le autorizzazioni.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Passaggio 5: Verificare le estensioni

Per evitare di dover specificare il nome dello schema, aggiungi lo schema `tiger` al percorso di ricerca usando il seguente comando.

```
SET search_path=public,tiger;
```

```
SET
```

Verificate lo `tiger` schema utilizzando la seguente SELECT istruzione.

```
SELECT address, streetname, streettypeabbrev, zip
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Per saperne di più su questa estensione, consulta Tiger Geocoder nella documentazione di Post. GIS

Verifica lo schema `topology` usando la seguente istruzione `SELECT`. Questa richiama la funzione `createtopology` per registrare un nuovo oggetto topologia (my_new_topo) con l'identificatore di riferimento spaziale specificato (26986) e la tolleranza predefinita (0,5). Per saperne di più, consulta la documentazione CreateTopologydi Post. GIS

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Passaggio 6: Aggiorna l'GISestensione Post

Ogni nuova versione di Postgre SQL supporta una o più versioni dell'GISestensione Post compatibili con quella versione. L'aggiornamento del SQL motore Postgre a una nuova versione non aggiorna automaticamente l'estensione Post. GIS Prima di aggiornare il SQL motore Postgre, in genere aggiorni Post GIS alla versione più recente disponibile per la versione corrente di Postgre. SQL Per informazioni dettagliate, consultare Versioni successive all'estensione GIS.

Dopo l'aggiornamento del SQL motore Postgre, aggiorni nuovamente l'GISestensione Post, alla versione supportata per la versione del motore Postgre appena aggiornata. SQL Per ulteriori informazioni sull'aggiornamento del motore Postgre, consulta. SQL Test di un aggiornamento del cluster database di produzione a una nuova versione principale

Puoi verificare gli aggiornamenti delle versioni dell'GISestensione Post disponibili sul tuo cluster Aurora Postgre SQL DB RDS per l'istanza DB in qualsiasi momento. Per farlo, esegui il comando seguente. Questa funzione è disponibile con Post GIS 2.5.0 e versioni successive.

```
SELECT postGIS_extensions_upgrade();
```

Se la tua applicazione non supporta l'ultima GIS versione di Post, puoi installare una versione precedente di Post GIS disponibile nella tua versione principale come segue.

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

Se desideri eseguire l'aggiornamento a una GIS versione di Post specifica da una versione precedente, puoi anche utilizzare il seguente comando.

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

A seconda della versione da cui si esegue l'aggiornamento, potrebbe essere necessario utilizzare nuovamente questa funzione. Il risultato della prima esecuzione della funzione determina se è necessaria una funzione di aggiornamento aggiuntiva. Ad esempio, questo è il caso dell'aggiornamento da Post GIS 2 a Post GIS 3. Per ulteriori informazioni, consulta Aggiornamento da Post 2 a Post 3 GIS GIS.

Se hai aggiornato questa estensione per prepararti all'aggiornamento di una versione principale del SQL motore Postgre, puoi continuare con altre attività preliminari. Per ulteriori informazioni, consulta Test di un aggiornamento del cluster database di produzione a una nuova versione principale.

Versioni successive all'estensione GIS

Si consiglia di installare le versioni di tutte le estensioni, ad esempio PostGIS, elencate in Extension versions for Aurora Postgre SQL -Compatible Edition nelle Note di rilascio per Aurora Postgre. SQL Per ottenere un elenco delle versioni disponibili nella versione, utilizza il comando seguente.

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

È possibile trovare informazioni sulla versione nelle seguenti sezioni delle Note di rilascio di Aurora Postgre: SQL

- Versioni di estensione per Aurora Postgre 14 SQL
- Versioni di estensione per Aurora SQL Postgre -Compatible Edition 13
- Versioni di estensione per Aurora SQL Postgre -Compatible Edition 12
- Versioni di estensione per Aurora SQL Postgre -Compatible Edition 11

- Versioni di estensione per Aurora SQL Postgre -Compatible Edition 10
- Versioni di estensione per Aurora SQL Postgre -Compatible Edition 9.6

Aggiornamento da Post 2 a Post 3 GIS GIS

A partire dalla versione 3.0, la funzionalità Post GIS raster è ora un'estensione separata,. `postgis_raster` Questa estensione dispone di un proprio percorso di installazione e aggiornamento. Ciò rimuove dall'estensione `postgis` core dozzine di funzioni, tipi di dati e altri artefatti necessari per l'elaborazione di immagini raster. Ciò significa che se il caso d'uso non richiede l'elaborazione raster, non è necessario installare l'estensione `postgis_raster`.

Nel seguente esempio di aggiornamento, il primo comando di aggiornamento estrae la funzionalità raster nell'estensione `postgis_raster`. È quindi necessario un secondo comando di aggiornamento per eseguire l'aggiornamento di `postgis_raster` alla nuova versione.

Per eseguire l'aggiornamento da Post GIS 2 a Post GIS 3

1. Identifica la versione predefinita di Post GIS disponibile per la SQL versione Postgre sul tuo cluster Aurora Postgre DB. SQL A questo scopo, esegui la query seguente.

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                          comment
---------+-----------------+-------------------
+-------------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
(1 row)
```

2. Identifica le versioni di Post GIS installate in ogni database sull'istanza writer del tuo cluster Aurora SQL Postgre DB. In altre parole, esegui la query su ogni database utente come riportato di seguito.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
```

```
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                            Description
---------+---------+--------
+------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

Questa mancata corrispondenza tra la versione predefinita (Post GIS 3.1.4) e la versione installata (Post GIS 2.3.7) significa che è necessario aggiornare l'estensione Post. GIS

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. Esegui la seguente query per verificare che la funzionalità raster sia ora contenuta nel proprio pacchetto.

```
SELECT
    probin,
    count(*)
FROM
    pg_proc
WHERE
    probin LIKE '%postgis%'
GROUP BY
    probin;
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
(2 rows)
```

L'output mostra che c'è ancora una differenza tra le versioni. Le GIS funzioni Post sono la versione 3 (postgis-3), mentre le funzioni raster (rtpostgis) sono la versione 2 (rtpostgis-2.3). Per

completare l'aggiornamento, esegui nuovamente il comando di aggiornamento, come riportato di seguito.

```
postgres=> SELECT postgis_extensions_upgrade();
```

Puoi ignorare i messaggi di avviso in sicurezza. Esegui nuovamente la seguente query per verificare che l'aggiornamento sia stato completato. L'aggiornamento è completo quando Post GIS e tutte le estensioni correlate non sono contrassegnate come da aggiornare.

```
SELECT postgis_full_version();
```

4. Utilizza la seguente query per visualizzare il processo di aggiornamento completato e le estensioni impacchettate separatamente e verifica che le relative versioni corrispondano.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
      Name      | Version | Schema |                            Description
----------------+---------+--------
+------------------------------------------------------------------------
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

L'output mostra che l'estensione Post GIS 2 è stata aggiornata a Post GIS 3 `postgis` ed entrambe, nonché l'`postgis_raster`estensione ora separata, sono alla versione 3.1.5.

Al termine dell'aggiornamento, se non prevedi di utilizzare la funzionalità raster, puoi rimuovere l'estensione come segue.

```
DROP EXTENSION postgis_raster;
```

Gestione delle partizioni PostgreSQL con l'estensione pg_partman

Il partizionamento delle tabelle PostgreSQL fornisce un framework per la gestione ad alte prestazioni di input e reporting dei dati. Utilizzare il partizionamento per database che richiedono un input molto veloce di grandi quantità di dati. Il partizionamento fornisce anche query di tabelle di grandi dimensioni più veloci. Il partizionamento consente di conservare i dati senza influire sull'istanza del database perché richiede meno risorse I/O.

Utilizzando il partizionamento, è possibile suddividere i dati in blocchi di dimensioni personalizzate per l'elaborazione. Ad esempio, è possibile partizionare i dati delle serie temporali per intervalli quali orario, giornaliero, settimanale, mensile, trimestrale, annuale, personalizzato o qualsiasi combinazione di questi. Per un esempio di dati di serie temporali, se la tabella è stata partizionata per ora, ogni partizione conterrà un'ora di dati. Se si partiziona la tabella delle serie temporali per giorno, le partizioni conterranno i dati di un giorno e così via. La chiave di partizione controlla le dimensioni di una partizione.

Quando si utilizza un comando SQL `INSERT` o `UPDATE` in una tabella partizionata, il motore database indirizza i dati alla partizione appropriata. Le partizioni di tabella PostgreSQL che memorizzano i dati sono tabelle figlio della tabella principale.

Durante le letture delle query di database, l'ottimizzatore PostgreSQL esamina la clausola `WHERE` della query e, se possibile, indirizza la scansione del database solo alle partizioni pertinenti.

A partire dalla versione 10, PostgreSQL utilizza il partizionamento dichiarativo per implementare il partizionamento delle tabelle. Questo è noto anche come partizionamento PostgreSQL nativo. Prima di PostgreSQL versione 10, per implementare le partizioni venivano utilizzati i trigger.

Il partizionamento delle tabelle PostgreSQL fornisce le seguenti funzionalità:

- Creazione di nuove partizioni in qualsiasi momento.
- Intervalli di partizione variabili.
- Partizioni scollegabili e ricollegabili utilizzando istruzioni DDL (Data Definition Language).

Ad esempio, le partizioni scollegabili sono utili per rimuovere i dati storici dalla partizione principale, conservando i dati storici per l'analisi.

- Le nuove partizioni ereditano le proprietà della tabella di database padre, tra cui:
  - Indici
  - Chiavi primarie, che devono includere la colonna delle chiavi di partizione
  - Chiavi esterne
  - Vincoli check
  - Riferimenti
- Creazione di indici per la tabella completa o per ogni partizione specifica.

Non è possibile modificare lo schema per una singola partizione. Tuttavia, è possibile modificare la tabella padre (ad esempio, aggiungendo una nuova colonna), che si propaga alle partizioni.

Argomenti

- Panoramica dell'estensione PostgreSQL pg_partman
- Abilitazione dell'estensione pg_partman
- Configurazione delle partizioni utilizzando la funzione create_parent
- Configurazione della manutenzione delle partizioni utilizzando la funzione run_maintenance ance_proc

Panoramica dell'estensione PostgreSQL pg_partman

È possibile utilizzare l'estensione `pg_partman` PostgreSQL per automatizzare la creazione e la manutenzione delle partizioni di tabella. Per informazioni più generali, consulta PG Partition Manager nella documentazione di `pg_partman`.

> **Note**
>
> L'estensione `pg_partman` è supportata su Aurora PostgreSQL versioni 12.6 e successive.

Invece di dover creare manualmente ogni partizione, è possibile configurare `pg_partman` con le seguenti impostazioni:

- Tabella da partizionare
- Tipo di partizione
- Chiave di partizione
- Granularità delle partizioni
- Opzioni di pre-creazione e gestione delle partizioni

Dopo aver creato una tabella con partizioni PostgreSQL, la si registra con `pg_partman` chiamando la funzione `create_parent`. In questo modo vengono create le partizioni necessarie in base ai parametri passati alla funzione.

L'estensione `pg_partman` fornisce anche la funzione `run_maintenance_proc`, che è possibile chiamare su base pianificata per gestire automaticamente le partizioni. Per pianificare la creazione delle partizioni appropriate in base alle esigenze, puoi pianificare questa funzione in modo che venga eseguita periodicamente (ad esempio, ogni ora). È inoltre possibile assicurarsi che le partizioni vengano eliminate automaticamente.

Abilitazione dell'estensione pg_partman

Se disponi di più database all'interno della stessa istanza database per cui desideri gestire le partizioni, è necessario abilitare l'estensione `pg_partman` separatamente per ogni database. Per abilitare l'estensione `pg_partman` per un database specifico, crea lo schema di manutenzione delle partizioni, quindi crea l'estensione `pg_partman` nel modo seguente:

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> Per creare l'estensione `pg_partman`, assicurati di disporre dei privilegi `rds_superuser`.

Se viene restituito un errore come il seguente, concedi i privilegi `rds_superuser` all'account o utilizza l'account utente avanzato.

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

Per concedere i privilegi `rds_superuser`, collegati con l'account utente avanzato ed emetti il seguente comando:

```
GRANT rds_superuser TO user-or-role;
```

Per gli esempi che mostrano l'uso dell'estensione pg_partman, si utilizza la tabella di database e la partizione di esempio seguenti. Questo database utilizza una tabella partizionata basata su un timestamp. Uno schema `data_mart` contiene una tabella denominata `events` con una colonna denominata `created_at`. Nella tabella `events` sono incluse le seguenti impostazioni:

- Chiavi primarie `event_id` e `created_at`, che devono avere la colonna utilizzata per guidare la partizione.
- Un vincolo di controllo `ck_valid_operation` per applicare i valori per una colonna della tabella `operation` .
- Due chiavi esterne, dove una (`fk_orga_membership`)) punta alla tabella esterna `organization` e l'altra (`fk_parent_event_id`) è una chiave esterna autoreferenziata.
- Due indici, dove uno (`idx_org_id`) è per la chiave esterna e l'altro (`idx_event_type`) è per il tipo di evento.

Le seguenti istruzioni DDL creano questi oggetti, che verranno inclusi automaticamente in ogni partizione.

```
CREATE SCHEMA data_mart;
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
```

```
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

Configurazione delle partizioni utilizzando la funzione create_parent

Dopo aver abilitato l'estensione `pg_partman`, utilizza la funzione `create_parent` per configurare le partizioni all'interno dello schema di manutenzione delle partizioni. In questo esempio viene utilizzato l'esempio della tabella `events` creato in Abilitazione dell'estensione pg_partman. Richiama la funzione `create_parent` come segue:

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

I parametri sono i seguenti:

- `p_parent_table` – La tabella partizionata padre. Questa tabella deve già esistere ed essere completa, deve ovvero includere lo schema.
- `p_control` – Colonna su cui basare il partizionamento. Il tipo di dati deve essere intero o basato sul tempo.
- `p_type`: il tipo è `'range'` o `'list'`.
- `p_interval` – Intervallo di tempo o intervallo intero per ogni partizione. I valori di esempio includono `1 day1 hour`, e così via.
- `p_premake` – Il numero di partizioni da creare in anticipo per supportare nuovi inserimenti.

Per una descrizione completa della funzione `create_parent`, consulta Funzioni di creazione nella documentazione `pg_partman`.

Configurazione della manutenzione delle partizioni utilizzando la funzione run_maintenance ance_proc

È possibile eseguire operazioni di manutenzione delle partizioni per creare automaticamente nuove partizioni, scollegare partizioni o rimuovere partizioni obsolete. La manutenzione delle partizioni si basa sulla funzione `run_maintenance_proc` dell'estensione `pg_partman` e dell'estensione `pg_cron`, che avvia un pianificatore interno. Lo scheduler `pg_cron` esegue automaticamente le istruzioni SQL, le funzioni e le procedure definite nei database.

Nell'esempio seguente viene utilizzato l'esempio della tabella `events` creata in Abilitazione dell'estensione pg_partman per impostare l'esecuzione automatica delle operazioni di manutenzione delle partizioni. Come prerequisito, aggiungere `pg_cron` al parametro `shared_preload_libraries` nel gruppo di parametri dell'istanza database.

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

Di seguito, è possibile trovare una step-by-step spiegazione dell'esempio precedente:

1. Modifica il gruppo di parametri associato all'istanza database e aggiungi `pg_cron` al valore del parametro `shared_preload_libraries`. Perché questa modifica abbia effetto, è necessario riavviare l'istanza database. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri DB in .
2. Emettere il comando `CREATE EXTENSION pg_cron;` utilizzando un account con le autorizzazioni `rds_superuser`. In questo modo, viene abilitata l'estensione `pg_cron`. Per ulteriori informazioni, consulta Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron.
3. Emettere il comando `UPDATE partman.part_config` per regolare le impostazioni `pg_partman` per la tabella `data_mart.events`.
4. Eseguire il comando `SET . . .` per configurare la tabella `data_mart.events`, con le seguenti clausole:

a. `infinite_time_partitions = true,` – Configura la tabella in modo da poter creare automaticamente nuove partizioni senza limiti.

b. `retention = '3 months',` – Configura la tabella in modo che venga conservata per un massimo di tre mesi.

c. `retention_keep_table=true` – Configura la tabella in modo che quando il periodo di conservazione è scaduto, la tabella non venga eliminata automaticamente. Le partizioni precedenti al periodo di conservazione vengono invece scollegate dalla tabella padre.

5. Eseguire il comando `SELECT cron.schedule . . .` per creare una chiamata di funzione `pg_cron`. Questa chiamata definisce la frequenza con cui lo scheduler esegue la procedura di manutenzione `pg_partman`, `partman.run_maintenance_proc`. Per questo esempio, la procedura viene eseguita ogni ora.

Per una descrizione completa della funzione `run_maintenance_proc`, consulta Funzioni di manutenzione nella documentazione di `pg_partman`.

Pianificazione della manutenzione con l'estensione PostgreSQL pg_cron

Puoi utilizzare l'estensione PostgreSQL `pg_cron` per pianificare i comandi di manutenzione all'interno di un database PostgreSQL. Per ulteriori informazioni sull'estensione, consulta Che cos'è pg_cron? nella documentazione di pg_cron.

L'estensione `pg_cron` è supportata sul motore Aurora PostgreSQL versioni 12.6 e successive

Per ulteriori informazioni sull'uso di `pg_cron`, consulta Schedule jobs with pg_cron on your RDS for PostgreSQL or your Aurora PostgreSQL-Compatible Edition databases (Pianificazione dei processi con pg_cron sui database RDS per PostgreSQL o Aurora edizione compatibile con PostgreSQL).

Argomenti

- Configurazione dell'estensione pg_cron
- Concessione delle autorizzazioni per utilizzare pg_cron agli utenti del database
- Programmazione di processi pg_cron
- Riferimento per l'estensione pg_cron

Configurazione dell'estensione pg_cron

Configura l'estensione `pg_cron` come riportato di seguito:

1. Modifica il gruppo di parametri personalizzati associato all'istanza database PostgreSQL aggiungendo `pg_cron` al valore del parametro `shared_preload_libraries`.

   Per rendere effettive le modifiche apportate al gruppo di parametri, riavvia l'istanza database PostgreSQL. Per ulteriori informazioni sull'utilizzo dei gruppi di parametri, consulta Amazon Aurora PostgreSQL parametri.

2. Dopo il riavvio dell'istanza database PostgreSQL, esegui il comando riportato di seguito utilizzando un account che dispone delle autorizzazioni `rds_superuser`. Ad esempio, se hai utilizzato le impostazioni di default quando hai creato il cluster di database Aurora PostgreSQL, connettiti come utente `postgres` e crea l'estensione.

   ```
   CREATE EXTENSION pg_cron;
   ```

   Lo scheduler `pg_cron` è impostato nel database PostgreSQL predefinito denominato `postgres`. Gli oggetti `pg_cron` vengono creati in questo database `postgres` e tutte le azioni di pianificazione vengono eseguite in questo database.

3. Puoi usare le impostazioni predefinite oppure puoi pianificare i processi da eseguire in altri database all'interno dell'istanza database di PostgreSQL. Per pianificare i processi per altri database all'interno dell'istanza database di PostgreSQL, consulta l'esempio in Pianificazione di un processo cron per un database diverso da quello predefinito.

Concessione delle autorizzazioni per utilizzare pg_cron agli utenti del database

L'installazione dell'estensione `pg_cron` richiede privilegi `rds_superuser`. Tuttavia, le autorizzazioni per utilizzare `pg_cron` possono essere concesse (da un membro del gruppo/ruolo `rds_superuser`) ad altri utenti del database, in modo che possano pianificare i propri lavori. Ti consigliamo di concedere le autorizzazioni allo schema `cron` solo se in base alle esigenze se migliora le operazioni nell'ambiente di produzione.

Per concedere a un database l'autorizzazione utente nello schema `cron`, esegui il seguente comando:

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

Questo dà *`db-user`* il permesso di accedere allo `cron` schema per pianificare i cron job per gli oggetti a cui hanno i permessi di accesso. Se l'utente del database non dispone di autorizzazioni, il

processo non va a buon fine dopo aver pubblicato il messaggio di errore nel file `postgresql.log`, come mostrato di seguito:

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

In altre parole, assicuratevi che gli utenti del database a cui sono concesse le autorizzazioni sullo `cron` schema dispongano anche delle autorizzazioni sugli oggetti (tabelle, schemi e così via) che intendono pianificare.

Nella tabella vengono inoltre registrati i dettagli del cron job e del suo esito positivo o negativo. `cron.job_run_details` Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .

Programmazione di processi pg_cron

Nelle sezioni seguenti viene illustrato come è possibile pianificare varie attività di gestione utilizzando le attività `pg_cron`.

> **Note**
>
> Quando crei processi `pg_cron`, controlla che l'impostazione `max_worker_processes` sia maggiore del numero di `cron.max_running_jobs`. Un processo `pg_cron` non riesce se i processi worker in background vengono esauriti. Il numero predefinito di processi `pg_cron` è 5. Per ulteriori informazioni, consulta Parametri per la gestione dell'estensione pg_cron.

Argomenti

- Vacuum di una tabella
- Eliminazione della tabella della cronologia di pg_cron
- Registrazione degli errori solo nel file postgresql.log
- Pianificazione di un processo cron per un database diverso da quello predefinito

Vacuum di una tabella

Autovacuum gestisce la manutenzione del vacuum per la maggior parte dei casi. Tuttavia, potresti voler programmare un vacuum di una tabella specifica in un momento di tua scelta.

Di seguito è riportato un esempio di utilizzo della funzione `cron.schedule` per impostare un processo in modo da utilizzare `VACUUM FREEZE` su una tabella specifica ogni giorno alle 22:00 (GMT).

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

Dopo l'esecuzione dell'esempio precedente, è possibile controllare la cronologia nella tabella `cron.job_run_details` come riportato di seguito.

```
postgres=> SELECT * FROM cron.job_run_details;
jobid  | runid | job_pid | database | username | command                        |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+--------------------------------
+-----------+----------------+-------------------------------
+-------------------------------
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
(1 row)
```

Di seguito è riportata un'interrogazione della `cron.job_run_details` tabella per vedere i job falliti.

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
jobid | runid | job_pid | database | username | command                        | status
 | return_message                                | start_time                    |
 end_time
------+-------+---------+----------+----------+--------------------------------+--------
+-----------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .

Eliminazione della tabella della cronologia di pg_cron

La tabella `cron.job_run_details` contiene una cronologia di processi cron che può crescere a dismisura nel tempo. Si consiglia di pianificare un processo che elimini questa tabella. Ad esempio, conservare una settimana di voci può essere sufficiente per la risoluzione dei problemi.

Nell'esempio seguente viene utilizzata la funzione cron.schedule per pianificare un processo che viene eseguito ogni giorno a mezzanotte per rimuovere la tabella `cron.job_run_details`. Il processo conserva solo gli ultimi sette giorni. Utilizza l'account `rds_superuser` per pianificare il processo come riportato di seguito.

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .

Registrazione degli errori solo nel file postgresql.log

Per disattivare la scrittura nella tabella `cron.job_run_details`, modifica il gruppo di parametri associato all'istanza database PostgreSQL e disattiva il parametro `cron.log_run`. L'estensione `pg_cron` non scrive più nella tabella e acquisisce gli errori solo nel file `postgresql.log`. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri DB in .

Utilizza il seguente comando per controllare il valore del parametro `cron.log_run`.

```
postgres=> SHOW cron.log_run;
```

Per ulteriori informazioni, consulta Parametri per la gestione dell'estensione pg_cron.

Pianificazione di un processo cron per un database diverso da quello predefinito

I metadati per `pg_cron` sono tutti contenuti nel database predefinito PostgreSQL denominato `postgres`. Poiché i worker in background vengono utilizzati per l'esecuzione dei processi cron di manutenzione, puoi pianificare un processo in uno qualsiasi dei database all'interno dell'istanza database PostgreSQL.

> **Note**
>
> Solo gli utenti con `rds_superuser` ruolo o `rds_superuser` privilegi possono elencare tutti i cron job nel database. Gli altri utenti possono visualizzare solo i propri lavori nella `cron.job` tabella.

1. Nel database cron, pianifica il processo come si farebbe normalmente utilizzando cron.schedule.

```
postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
 freeze test_table');
```

2. Con il ruolo `rds_superuser`, aggiorna la colonna del database per il processo appena creato in modo che venga eseguito in un altro database all'interno dell'istanza database PostgreSQL.

```
postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
```

3. Verifica eseguendo una query sulla tabella `cron.job`.

```
postgres=> SELECT * FROM cron.job;
jobid | schedule    | command                        | nodename  | nodeport |
 database | username  | active | jobname
------+-------------+--------------------------------+-----------+----------
+----------+-----------+--------+------------------------
106   | 29 03 * * * | vacuum freeze test_table       | localhost | 8192     |
 database1| adminuser | t      | database1 manual vacuum
  1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
 postgres | adminuser | t      | manual vacuum
(2 rows)
```

> **Note**
>
> In alcune situazioni, è possibile aggiungere un processo cron che si intende eseguire in un database diverso. In questi casi, il processo potrebbe essere eseguito nel database predefinito (`postgres`) prima di aggiornare la colonna del database corretta. Se il nome utente dispone delle autorizzazioni, il processo viene eseguito correttamente nel database predefinito.

Riferimento per l'estensione pg_cron

È possibile utilizzare i seguenti parametri, funzioni e tabelle con l'estensione `pg_cron`. Per ulteriori informazioni, consultare Che cos'è pg_cron? nella documentazione di pg_cron.

Argomenti

- Parametri per la gestione dell'estensione pg_cron
- Riferimento alla funzione: cron.schedule
- Riferimento alla funzione: cron.unschedule
- Tabelle per la pianificazione dei processi e l'acquisizione dello stato

Parametri per la gestione dell'estensione pg_cron

Di seguito è riportato l'elenco dei parametri che consentono di controllare il comportamento dell'estensione `pg_cron`.

Parametro	Descrizione
cron.database_name	Il database in cui vengono conservati i metadati `pg_cron`.
`cron.host`	Il nome host per connettersi a PostgressQL. Non è possibile modificare questo valore.
`cron.log_run`	Registra tutti i processi eseguiti nella tabella `job_run_details` . I valori sono `on` o `off`. Per ulteriori informazioni, consulta Tabelle per la pianificazione dei processi e l'acquisizione dello stato .
`cron.log_statement`	Registra tutte le istruzioni cron prima di eseguirle. I valori sono `on` o `off`.
`cron.max_running_jobs`	Numero massimo di processi che possono essere eseguiti contemporaneamente.

Parametro	Descrizione
`cron.use_background_workers`	Utilizza i worker in background anziché le sessioni client. Non è possibile modificare questo valore.

Puoi utilizzare il comando SQL seguente per visualizzare questi parametri e i relativi valori.

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

Riferimento alla funzione: cron.schedule

Questa funzione pianifica un processo cron. Il processo viene inizialmente pianificato nel database `postgres` predefinito. La funzione restituisce un valore `bigint` che rappresenta l'identificativo del processo. Per pianificare l'esecuzione di processi in altri database all'interno dell'istanza database di PostgreSQL, consulta l'esempio in Pianificazione di un processo cron per un database diverso da quello predefinito.

La funzione ha due diverse sintassi.

Sintassi

```
cron.schedule (job_name,
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

Parametri

Parametro	Descrizione
`job_name`	Il nome del processo cron.

Parametro	Descrizione
`schedule`	Il testo che indica la pianificazione per il processo cron. Il formato è il formato cron standard.
`command`	Testo del comando da eseguire.

Esempi

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)

postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

Riferimento alla funzione: cron.unschedule

Questa funzione elimina un processo cron. Puoi specificare `job_name` o `job_id`. Una policy assicura che l'utente sia il proprietario e possa rimuovere la pianificazione per il processo. La funzione restituisce un valore Booleano che indica la riuscita o l'errore.

La funzione ha la seguente sintassi.

Sintassi

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

Parametri

Parametro	Descrizione
`job_id`	Un identificativo di processo restituito dalla funzione `cron.schedule` quando è stato pianificato il processo cron.
`job_name`	Il nome di un processo cron pianificato con la funzione `cron.schedule` .

Esempi

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)

postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

Tabelle per la pianificazione dei processi e l'acquisizione dello stato

Le seguenti tabelle vengono create e utilizzate per pianificare i processi cron e registrare il modo in cui i processi vengono completati.

Tabella	Descrizione
`cron.job`	Contiene i metadati relativi a ciascun processo pianificato. La maggior parte delle interazioni con questa tabella dovrebbe essere eseguita tramite le funzioni `cron.schedule` e `cron.unschedule` .

Tabella	Descrizione
	Important Si sconsiglia di concedere privilegi di aggiornamento/inserimento direttamente a questa tabella. In questo modo, si consente all'utente di aggiornare la colonna `username` da eseguire come `rds-superuser` .
`cron.job_run_details`	Contiene informazioni cronologiche sulle esecuzioni precedenti dei processi pianificati. Ciò è utile per analizzare lo stato, i messaggi restituiti e l'ora di inizio e di fine dall'esecuzione del processo. **Note** Per evitare che questa tabella cresca in maniera indefinita, è necessario eliminarla su base regolare. Per un esempio, consulta Eliminazione della tabella della cronologia di pg_cron.

Utilizzo pgAudit per registrare l'attività del database

Gli istituti finanziari, gli enti governativi e molti settori devono conservare i log di audit per soddisfare i requisiti normativi. Utilizzando l'estensione Postgre SQL Audit (pgAudit) con il cluster Aurora Postgre DB Postgre SQL DB, è possibile acquisire i record dettagliati che in genere sono necessari ai revisori o per soddisfare i requisiti normativi. SQL Ad esempio, puoi configurare l' pgAudit estensione per tenere traccia delle modifiche apportate a database e tabelle specifici, per registrare l'utente che ha apportato la modifica e molti altri dettagli.

L' pgAudit estensione si basa sulla funzionalità dell'infrastruttura di SQL registrazione nativa di Postgre estendendo i messaggi di registro con maggiori dettagli. In altre parole, l'approccio utilizzato per visualizzare il registro di audit è lo stesso utilizzato per visualizzare i messaggi di registro. Per ulteriori informazioni sulla registrazione di Postgre, consulta. SQL File di log del database Aurora PostgreSQL

L' pgAudit estensione rimuove i dati sensibili, come le password in chiaro, dai log. Se il cluster Aurora Postgre SQL DB per l'istanza SQL Postgre DB registrare le istruzioni Data Manipulation Language (DML) come descritto in dettaglio inAttivazione della registrazione delle query per il cluster database Aurora PostgreSQL, è possibile evitare il problema della password in chiaro utilizzando l'estensione Postgre Audit. SQL

È possibile configurare l'audit sulle istanze database con un elevato grado di specificità. Puoi eseguire l'audit di tutti i database e di tutti gli utenti. In alternativa, è possibile scegliere di eseguire l'audit solo di determinati database, utenti e altri oggetti. È inoltre possibile escludere esplicitamente determinati utenti e database dall'audit. Per ulteriori informazioni, consulta Esclusione di utenti o database dalla registrazione di audit.

Data la quantità di dettagli che è possibile acquisire, ti consigliamo, se lo utilizzi, di monitorare il consumo di storage. pgAudit

L' pgAudit estensione è supportata su tutte le versioni di Aurora SQL Postgre disponibili. Per un elenco delle pgAudit versioni supportate dalla SQL versione Aurora Postgre, consulta Versioni di estensione per Amazon Aurora Postgre SQL nelle Note di rilascio per Aurora Postgre. SQL

Argomenti

- Configurazione dell' pgAudit estensione
- Audit di oggetti di database
- Esclusione di utenti o database dalla registrazione di audit
- Riferimento per l'pgAuditestensione

Configurazione dell' pgAudit estensione

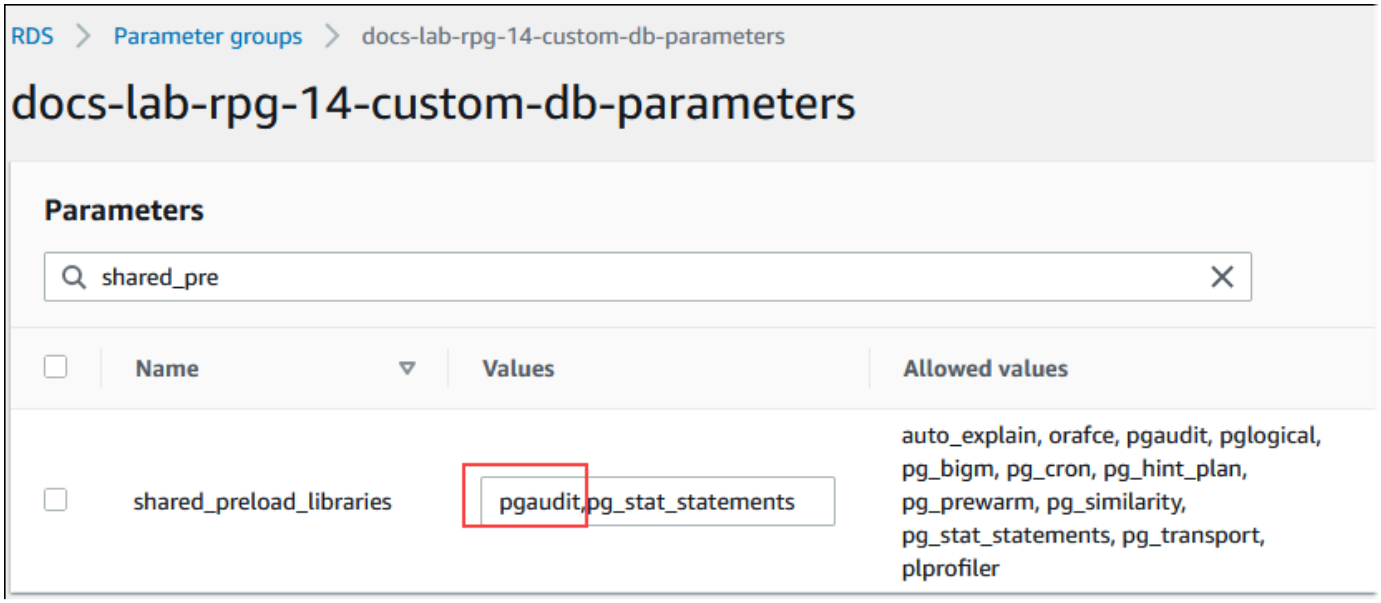
Per configurare l' pgAudit estensione sul cluster l'istanza Postgre SQL DB, devi prima aggiungerla pgAudit alle librerie condivise nel Postgre. RDS SQL gruppo di parametri del cluster DB personalizzato per il cluster DB Aurora SQL Postgre. Per informazioni sulla creazione di un gruppo di parametri del·cluster database, consulta . Successivamente, si installa l'estensione. pgAudit Infine, specifica i database e gli oggetti di cui eseguire l'audit. Le procedure in questa sezione mostrano come fare. Puoi utilizzare il plugin AWS Management Console o il AWS CLI.

Per eseguire tutte queste attività, sono richieste autorizzazioni come il ruolo `rds_superuser`.

Console

Per configurare l'estensione pgAudit

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli l'istanza Writer del cluster Aurora Postgre SQL DB l'istanza DB Postgre. SQL
3. Apri la scheda Configurazione per l'istanza del cluster writer Aurora Postgre SQL DB. Tra i dettagli dell'istanza, individua il collegamento Gruppo di parametri.
4. Scegli il link per aprire i parametri personalizzati associati al tuo cluster Aurora SQL Postgre DB.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pgaudit` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Riavviare l'istanza writer del in modo che la modifica al parametro abbia effetto. `shared_preload_libraries`
9. Quando l'istanza è disponibile, verifica che sia stata inizializzata. pgAudit Usalo `psql` per connetterti all'istanza writer del tuo cluster Aurora Postgre SQL DB, RDS per l'istanza Postgre SQL DB, seguente.

```
SHOW shared_preload_libraries;
```

```
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. Con pgAudit initialized, ora puoi creare l'estensione. È necessario creare l'estensione dopo aver inizializzato la libreria perché l'`pgaudit`estensione installa i trigger di eventi per il controllo delle istruzioni del linguaggio di definizione dei dati (). DDL

```
CREATE EXTENSION pgaudit;
```

11. Chiudi la sessione `psql`.

```
labdb=> \q
```

12. Accedi al AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

13. Trova il parametro `pgaudit.log` nell'elenco e impostalo sul valore appropriato per il caso d'uso. Ad esempio, se si imposta il parametro `pgaudit.log` su `write` come mostrato nell'immagine seguente, gli inserimenti, gli aggiornamenti, le eliminazioni e alcuni altri tipi di modifiche vengono acquisiti nel registro.

Puoi anche scegliere uno dei seguenti valori per il parametro `pgaudit.log`.

- none: valore predefinito. Non viene registrata alcuna modifica al database.
- all: registra tutto (read, write, function, role, ddl, misc).

- ddl — Registra tutte le istruzioni del linguaggio di definizione dei dati (DDL) che non sono incluse nella `ROLE` classe.
- function: registra le chiamate di funzione e blocchi `DO`.
- misc: registra vari comandi come `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` e `SET`.
- read: registra `SELECT` e `COPY` quando l'origine è una relazione (ad esempio una tabella) o una query.
- role: registra le istruzioni correlate a ruoli e privilegi, ad esempio `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando la destinazione è una relazione (tabella).

14. Scegli Save changes (Salva modifiche).
15. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
16. Scegli l'istanza writer del cluster Aurora Postgre SQL DB RDS per l'istanza SQL dall'elenco Database.

AWS CLI

Per configurare pgAudit

Per eseguire la configurazione pgAudit utilizzando il AWS CLI, si richiama l'modify-db-parameter-groupoperazione per modificare i parametri del registro di controllo nel gruppo di parametri personalizzato, come illustrato nella procedura seguente.

1. Utilizzare quanto segue AWS CLI comando da aggiungere `pgaudit` al `shared_preload_libraries` parametro.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Usa quanto segue AWS CLI comando per riavviare l'istanza writer del RDS cluster Aurora Postgre DB per l'istanza SQL Postgre DB sia inizializzata.

```
aws rds reboot-db-instance \
```

```
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando l'istanza è disponibile, verifica che `pgaudit` sia stato inizializzato. Usalo `psql` per connetterti all'istanza writer del tuo cluster Aurora Postgre SQL DB, RDS per l'istanza Postgre SQL DB, seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

   Con pgAudit initialized, ora puoi creare l'estensione.

```
CREATE EXTENSION pgaudit;
```

4. Chiudi la `psql` sessione in modo da poter utilizzare il AWS CLI.

```
labdb=> \q
```

5. Usa quanto segue AWS CLI comando per specificare le classi di istruzioni che si desidera registrare mediante la registrazione di controllo della sessione. L'esempio imposta il parametro `pgaudit.log` su `write`, che acquisisce inserimenti, aggiornamenti ed eliminazioni nel registro.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
   --region aws-region
```

   Puoi anche scegliere uno dei seguenti valori per il parametro `pgaudit.log`.

   - none: valore predefinito. Non viene registrata alcuna modifica al database.
   - all: registra tutto (read, write, function, role, ddl, misc).
   - ddl — Registra tutte le istruzioni del linguaggio di definizione dei dati (DDL) che non sono incluse nella classe. `ROLE`
   - function: registra le chiamate di funzione e blocchi `DO`.
   - misc: registra vari comandi come `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` e `SET`.

- read: registra `SELECT` e `COPY` quando l'origine è una relazione (ad esempio una tabella) o una query.
- role: registra le istruzioni correlate a ruoli e privilegi, ad esempio `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
- write: registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando la destinazione è una relazione (tabella).

Riavviare l'istanza writer del RDS cluster Aurora Postgre DB per l'istanza DB SQL Postgre AWS CLI comando.

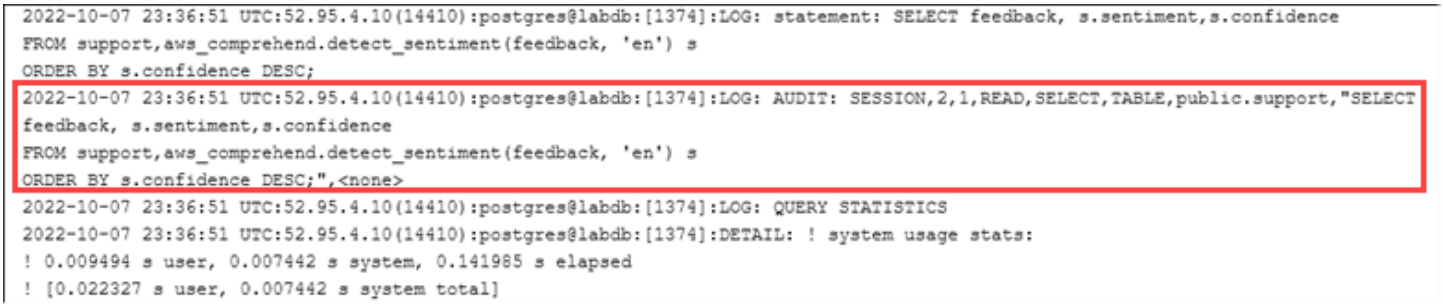
```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Audit di oggetti di database

Con la pgAudit configurazione sul cluster Aurora Postgre SQL DB RDS per l'istanza Postgre SQL DB vengono acquisite informazioni più dettagliate. SQL Ad esempio, mentre la configurazione di SQL registrazione predefinita di Postgre identifica la data e l'ora in cui è stata apportata una modifica in una tabella del database, con l'pgAuditestensione la voce di registro può includere lo schema, l'utente che ha apportato la modifica e altri dettagli a seconda di come sono configurati i parametri dell'estensione. È possibile configurare l'audit per tenere traccia delle modifiche nei modi seguenti.

- Per ogni sessione, per utente. Per il livello di sessione, è possibile acquisire il testo del comando completo.
- Per ogni oggetto, per utente e per database.

La funzionalità di audit degli oggetti viene attivata quando il ruolo `rds_pgaudit` viene creato nel sistema e quindi aggiunto al parametro `pgaudit.role` nel gruppo di parametri personalizzati. Per impostazione predefinita, l'impostazione del parametro `pgaudit.role` viene annullata e l'unico valore consentito è `rds_pgaudit`. Nelle fasi seguenti si presume che `pgaudit` sia stato inizializzato e che l'estensione `pgaudit` sia stata creata seguendo la procedura in Configurazione dell' pgAudit estensione.



Come mostrato in questo esempio, la riga "LOG:AUDIT:SESSION" fornisce informazioni sulla tabella e sul suo schema, tra gli altri dettagli.

Per configurare l'audit degli oggetti

1. Utilizzalo `psql` per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

   ```
   psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
   username=postgrespostgres --password --dbname=labdb
   ```

2. Crea un ruolo del database denominato `rds_pgaudit` utilizzando il comando seguente.

   ```
   labdb=> CREATE ROLE rds_pgaudit;
   CREATE ROLE
   labdb=>
   ```

3. Chiudi la sessione `psql`.

   ```
   labdb=> \q
   ```

   Nei prossimi passaggi, usa il AWS CLI per modificare i parametri del registro di controllo nel gruppo di parametri personalizzato.

4. Utilizzate quanto segue AWS CLI comando su cui impostare il `pgaudit.role` parametro`rds_pgaudit`. Per impostazione predefinita, questo parametro è vuoto e `rds_pgaudit` è l'unico valore consentito.

   ```
   aws rds modify-db-parameter-group \
      --db-parameter-group-name custom-param-group-name \
      --parameters
    "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
    \
      --region aws-region
   ```

5. Usa quanto segue AWS CLI comando per riavviare l'istanza writer del in modo che le modifiche ai parametri abbiano effetto.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

6. Esegui il comando seguente per verificare che `pgaudit.role` sia impostato su `rds_pgaudit`.

```
SHOW pgaudit.role;
pgaudit.role
------------------
rds_pgaudit
```

Per testare la pgAudit registrazione, puoi eseguire diversi comandi di esempio che desideri controllare. Ad esempio, si potrebbero eseguire i comandi seguenti.

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

I registri del database devono contenere una voce simile alla seguente:

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

Per informazioni sulla visualizzazione dei registri, consulta Monitoraggio dei file di log di Aurora.

Per ulteriori informazioni sull' pgAudit estensione, consulta pgAuditsu GitHub.

Esclusione di utenti o database dalla registrazione di audit

Come discusso in precedenzaFile di log del database Aurora PostgreSQL, i SQL log di Postgre consumano spazio di archiviazione. L'utilizzo dell' pgAudit estensione aumenta il volume di dati

raccolti nei log in misura diversa, a seconda delle modifiche che si registrano. Potrebbe non essere necessario controllare ogni utente o database nel cluster DB Aurora SQL Postgre.

Per ridurre al minimo l'impatto sull'archiviazione ed evitare di acquisire inutilmente i record di audit, è possibile escludere utenti e database dall'audit. È anche possibile modificare la registrazione all'interno di una determinata sessione. Negli esempi seguenti viene mostrato come fare.

> **Note**
>
> Le impostazioni dei parametri a livello di sessione hanno la precedenza sulle impostazioni nel gruppo di parametri del cluster DB personalizzato per l'istanza writer del cluster Aurora SQL Postgre DB, gruppo di parametri DB Postgre. SQL Per evitare che gli utenti del database ignorino le impostazioni di configurazione della registrazione di audit, accertati di modificare le loro autorizzazioni.

Supponiamo che il . Decidi quindi di non voler eseguire l'audit dell'utente `myuser`. È possibile disattivare il controllo per con il seguente comando. `myuser` SQL

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

Quindi, puoi utilizzare la seguente query per controllare la colonna `user_specific_settings` per `pgaudit.log` per verificare che sia impostata su `NONE`.

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

Viene visualizzato l'output riportato di seguito.

```
 user_name | user_specific_settings
-----------+------------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

Puoi disattivare la registrazione per un determinato utente durante la sessione con il database con il seguente comando.

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

Utilizza la seguente query per controllare la colonna delle impostazioni per pgaudit.log per una combinazione specifica di utente e database.

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

L'output visualizzato è simile al seguente.

```
  user_name | database_name |     settings
-----------+---------------+-----------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

Dopo aver disattivato l'audit per `myuser`, decidi di non voler tenere traccia delle modifiche apportate a `mydatabase`. Puoi disattivare l'audit per il database specifico utilizzando il comando seguente.

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

Quindi, utilizzate la seguente query per controllare la colonna database_specific_settings per confermare che pgaudit.log è impostato su. NONE

```
SELECT
a.datname AS database_name,
```

```
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
a.datname = 'mydatabase';
```

Viene visualizzato l'output riportato di seguito.

```
 database_name | database_specific_settings
---------------+----------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

Per ripristinare le impostazioni predefinite di myuser, utilizza il comando seguente:

```
ALTER USER myuser RESET pgaudit.log;
```

Per ripristinare i valori predefiniti delle impostazioni di un database, utilizza il comando seguente.

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

Per ripristinare l'impostazione predefinita di utente e database, utilizza il comando seguente.

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

Puoi anche acquisire eventi specifici nel registro impostando `pgaudit.log` su uno degli altri valori consentiti per il parametro `pgaudit.log`. Per ulteriori informazioni, consulta Elenco delle impostazioni consentite per il parametro `pgaudit.log`.

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

Riferimento per l'pgAuditestensione

Puoi specificare il livello di dettaglio desiderato per il registro di audit modificando uno o più dei parametri elencati in questa sezione.

pgAudit Comportamento di controllo

Puoi controllare la registrazione di audit modificando uno o più dei parametri elencati nella tabella seguente.

Parametro	Descrizione
`pgaudit.log`	Specifica quali classi di istruzioni verranno registrate per registrazione di audit della sessione. I valori consentiti includono ddl, function, misc, read, role, write, none, all. Per ulteriori informazioni, consulta Elenco delle impostazioni consentite per il parametro `pgaudit.log` .
`pgaudit.log_catalog`	Quando è attivato (impostato su 1), aggiunge istruzioni all'audit trail se tutte le relazioni in un'istruzione si trovano in pg_catalog.
`pgaudit.log_level`	Specifica il livello di registro che verrà utilizzato per le voci di registro. Valori consentiti: debug5, debug4, debug3, debug2, debug1, info, notice, warning, log
`pgaudit.log_parameter`	Quando è attivato (impostato su 1), i parametri passati con l'istruzione vengono acquisiti nel registro di audit.
`pgaudit.log_relation`	Quando è attivato (impostato su 1), il registro di controllo per la sessione crea una voce di registro separata per ogni relazione (TABLEVIEW, e così via) a cui si fa riferimento in un'DMList ruzione SELECT or.
`pgaudit.log_statement_once`	Specifica se la registrazione includerà il testo dell'istruzione e i parametri con la prima voce di registro per una combinazione istruzione/istruzione secondaria o con ogni voce.
`pgaudit.role`	Specifica il ruolo principale da utilizzare per la registrazione di verifica degli oggetti. L'unica voce consentita è `rds_pgaudit` .

Elenco delle impostazioni consentite per il parametro **`pgaudit.log`**

Valore	Descrizione
nessuno	Questa è l'impostazione predefinita. Non viene registrata alcuna modifica al database.
tutto	Registra tutto (read, write, function, role, ddl, misc).
ddl	Registra tutte le istruzioni del linguaggio di definizione dei dati (DDL) che non sono incluse nella `ROLE` classe.
funzione	Registra le chiamate di funzione e i blocchi `DO`.
misc	Registra vari comandi, ad esempio `DISCARD`, `FETCH`, `CHECKPOINT` , `VACUUM` e `SET`.
read	Registra `SELECT` e `COPY` quando l'origine è una relazione (ad esempio una tabella) o una query.
role	Registra le istruzioni relative a ruoli e privilegi, ad esempio `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` e `DROP ROLE`.
write	Registra `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` e `COPY` quando la destinazione è una relazione (tabella).

Per registrare più tipi di eventi con controllo di sessioni, utilizzare un elenco separato da virgole. Per registrare tutti i tipi di eventi, impostare `pgaudit.log` su `ALL`. Riavviare l'istanza database per applicare le modifiche.

Con il controllo dell'oggetto, è possibile perfezionare la registrazione di controllo per lavorare con relazioni specifiche. Ad esempio, è possibile richiedere la registrazione di audit per le operazioni `READ` su una o più tabelle.

Utilizzo di pglogical per sincronizzare i dati tra le istanze

Tutte le versioni Aurora Postgre attualmente disponibili RDS per SQL le versioni Postgre . `pglogical` L'estensione pglogical è precedente alla funzionalità di replica logica funzionalmente simile introdotta da Postgre nella versione 10. SQL Per ulteriori informazioni, consulta Panoramica della replica SQL logica di Postgree con Aurora.

L'`pglogical`estensione supporta la replica logica tra due o più cluster Aurora SQL Postgre DB. Supporta anche la replica tra diverse SQL versioni di Postgre e tra database in esecuzione su RDS SQL istanze Postgre DB e cluster Aurora Postgre DB. SQL L'estensione `pglogical` utilizza un modello publish-subscribe per replicare le modifiche apportate alle tabelle e ad altri oggetti, come le sequenze, da un publisher in un subscriber. Si basa su uno slot di replica per garantire la sincronizzazione delle modifiche da un nodo publisher a un nodo subscriber, definiti come indicato di seguito.

- Il nodo editore è il cluster Aurora Postgre SQL DB RDS che è la fonte dei dati da replicare su altri nodi. Il nodo publisher definisce le tabelle da replicare in un set di pubblicazione.
- Il nodo sottoscrittore è il WAL L'abbonato crea un abbonamento per connettersi all'editore e ottenere i dati decodificati. WAL contemporaneamente nel nodo publisher viene creato lo slot di replica.

Di seguito sono riportati gli argomenti sull'impostazione dell'estensione `pglogical`.

Argomenti

- Requisiti e limitazioni dell'estensione pglogical
- Impostazione dell'estensione pglogical
- Impostazione della replica logica per il cluster database Aurora PostgreSQL
- Riconnessione della replica logica dopo un aggiornamento principale
-
- Riferimento sui parametri dell'estensione pglogical

Requisiti e limitazioni dell'estensione pglogical

Tutte le versioni attualmente disponibili di Aurora Postgre SQL . `pglogical`

Sia il nodo publisher che il nodo subscriber devono essere impostati per la replica logica.

Le tabelle che si desidera replicare da un editore a un sottoscrittore devono avere gli stessi nomi e lo stesso schema. Inoltre devono contenere le stesse colonne e le colonne devono utilizzare gli stessi tipi di dati. Le tabelle del publisher e del subscriber devono avere le stesse chiavi primarie. Si consiglia di utilizzare solo il PRIMARY KEY come vincolo univoco.

Le tabelle sul nodo subscriber possono avere più vincoli permissivi rispetto a quelle sul nodo publisher per quanto riguarda vincoli e vincoli. CHECK NOT NULL

L'`pglogical`estensione offre funzionalità come la replica bidirezionale che non sono supportate dalla funzionalità di replica logica integrata in Postgre (versione 10 e successive). SQL Per ulteriori informazioni, consulta Replica bidirezionale di SQLPostgre con pglogical.

Impostazione dell'estensione pglogical

Per configurare l'`pglogical`estensione sul cluster l'istanza Postgre SQL DB, aggiungila `pglogical` alle librerie condivise nel Postgre. RDS SQL gruppo di parametri del cluster DB personalizzato per il cluster DB Aurora SQL Postgre. È inoltre necessario impostare il valore del parametro `rds.logical_replication` su `1` per attivare la decodifica logica. Infine, crei l'estensione nel database. È possibile utilizzare il AWS Management Console o AWS CLI per queste attività.

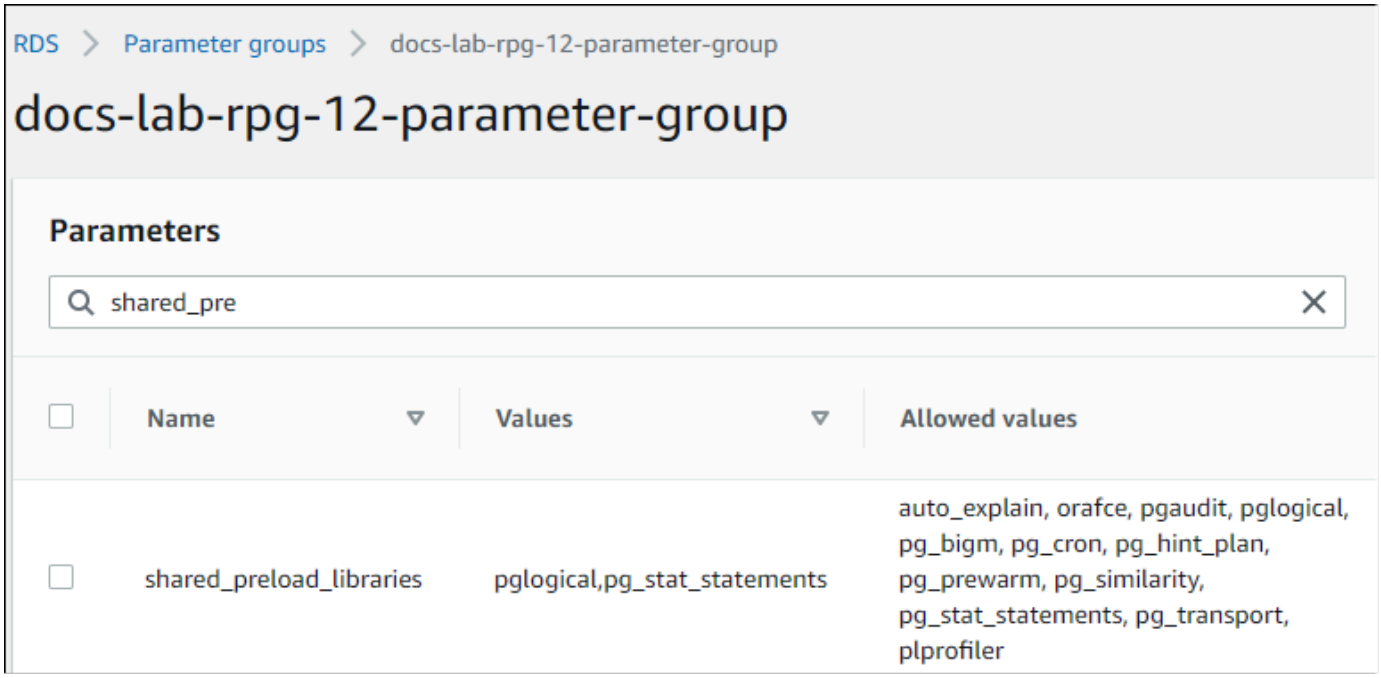
Per eseguire queste attività sono richieste le autorizzazioni del ruolo `rds_superuser`.

Per informazioni sulla creazione di un gruppo di parametri del·cluster database, consulta .

Console

Per impostare l'estensione pglogical

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli l'istanza Writer del cluster Aurora Postgre SQL DB l'istanza DB Postgre. SQL
3. Apri la scheda Configurazione per la tua istanza di masterizzatore di cluster Aurora Postgre SQL DB. Tra i dettagli dell'istanza, individua il collegamento Gruppo di parametri.
4. Scegli il link per aprire i parametri personalizzati associati al tuo cluster Aurora SQL Postgre DB.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pglogical` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Individua il parametro `rds.logical_replication` e impostalo su 1 per attivare la replica logica.
9. Riavviare l'istanza writer del RDS cluster Aurora Postgre DB per l'istanza SQL Postgre DB effetto.
10. Quando l'istanza è disponibile, puoi usare `psql` (opgAdmin) per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. Per verificare che l'estensione pglogical sia inizializzata, esegui il seguente comando.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pglogical
(1 row)
```

12. Verifica l'impostazione che abilita la decodifica logica, come indicato di seguito.

```
SHOW wal_level;
wal_level
-----------
 logical
```

```
(1 row)
```

13. Crea l'estensione, come indicato di seguito.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. Scegli Save changes (Salva modifiche).
15. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
16. Scegli l'istanza writer del cluster Aurora Postgre SQL DB RDS per l'istanza dall'elenco Database per selezionarla, quindi scegli Reboot dal menu Azioni.

AWS CLI

Per impostare l'estensione pglogical

Per configurare pglogical utilizzando AWS CLI, richiamate l'modify-db-parameter-groupoperazione per modificare determinati parametri nel gruppo di parametri personalizzato, come illustrato nella procedura seguente.

1. Utilizzate il AWS CLI comando seguente per aggiungere `pglogical` al parametro. `shared_preload_libraries`

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilizzate il AWS CLI comando seguente `rds.logical_replication` `1` per impostare l'attivazione della funzionalità di decodifica logica per l'istanza writer del cluster Aurora SQL Postgre DB.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
   --region aws-region
```

3. Usa il seguente AWS CLI comando per riavviare l'istanza writer del tuo RDS cluster Aurora Postgre DB per l'istanza SQL Postgre DB inizializzata.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

4. Quando l'istanza è disponibile, usala `psql` per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. Crea l'estensione, come indicato di seguito.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. Riavviare l'istanza writer del RDS cluster Aurora Postgre DB per l'istanza SQL Postgre DB AWS CLI

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Impostazione della replica logica per il cluster database Aurora PostgreSQL

La seguente procedura mostra come avviare la replica logica tra due cluster database Aurora PostgreSQL. I passaggi presuppongono che sia l'origine (publisher) che la destinazione (subscriber) abbiano l'estensione `pglogical` impostata come descritto dettagliatamente in Impostazione dell'estensione pglogical.

> **Note**
>
> Il nodo `node_name` di un abbonato non può iniziare con`rds`.

Per creare il nodo publisher e definire le tabelle da replicare

Questi passaggi presuppongono che il cluster database Aurora PostgreSQL abbia un'istanza di scrittura con un database contenente una o più tabelle che desideri replicare in un altro nodo. È necessario ricreare la struttura delle tabelle dal publisher nel subscriber, quindi prima, se occorre, recupera la struttura delle tabelle. Puoi farlo utilizzando il metacomando `psql \d` *`tablename`* e quindi creando la stessa tabella nell'istanza subscriber. Nella procedura seguente viene illustrato come creare una tabella di esempio nel publisher (origine) a scopo dimostrativo.

1. Utilizza `psql` per connetterti all'istanza che include la tabella da usare come origine per i subscriber.

   ```
   psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=labdb
   ```

   Se non hai una tabella esistente da replicare, puoi creare una tabella di esempio come indicato di seguito.

   a. Crea una tabella di esempio utilizzando la seguente istruzione SQL.

      ```
      CREATE TABLE docs_lab_table (a int PRIMARY KEY);
      ```

   b. Popola la tabella con i dati generati utilizzando la seguente istruzione SQL.

      ```
      INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
      INSERT 0 5000
      ```

   c. Verifica che i dati siano presenti nella tabella utilizzando la seguente istruzione SQL.

      ```
      SELECT count(*) FROM docs_lab_table;
      ```

2. Identifica il cluster database Aurora PostgreSQL come nodo publisher, come indicato di seguito.

   ```
   SELECT pglogical.create_node(
       node_name := 'docs_lab_provider',
       dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
    dbname=labdb');
    create_node
   -------------
      3410995529
   ```

```
(1 row)
```

3. Aggiungi la tabella da replicare al set di replica predefinito. Per ulteriori informazioni sui set di replica, consulta Replication sets (Set di replica) nella documentazione di pglogical.

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  ---------------------------
  t
  (1 row)
```

L'impostazione del nodo publisher è completata. Ora puoi impostare il nodo subscriber per ricevere gli aggiornamenti dal publisher.

Per impostare il nodo subscriber e creare una sottoscrizione per ricevere gli aggiornamenti

Questi passaggi presuppongono che sia stata eseguita l'impostazione del cluster database Aurora PostgreSQL con l'estensione `pglogical`. Per ulteriori informazioni, consulta Impostazione dell'estensione pglogical.

1. Utilizza `psql` per connetterti all'istanza per cui vuoi ricevere gli aggiornamenti dal publisher.

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Nel cluster database Aurora PostgreSQL del subscriber crea la stessa tabella presente nel publisher. In questo esempio, la tabella è `docs_lab_table`. È possibile creare la tabella come indicato di seguito.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. Verifica che questa tabella sia vuota.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. Identifica il cluster database Aurora PostgreSQL come nodo subscriber, come indicato di seguito.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. Crea la sottoscrizione.

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
(1 row)
```

Una volta completato questo passaggio, i dati della tabella del publisher vengono creati nella tabella del subscriber. È possibile verificare questa operazione utilizzando la seguente query SQL.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
(1 row)
```

Da questo momento in poi, le modifiche apportate alla tabella del publisher vengono replicate nella tabella del subscriber.

Riconnessione della replica logica dopo un aggiornamento principale

Prima di poter eseguire un aggiornamento della versione principale di un cluster Aurora Postgre SQL DB RDS configurata come nodo publisher per la replica logica, è necessario eliminare tutti gli slot di

replica, anche quelli non attivi. Si consiglia di deviare temporaneamente le transazioni del database dal nodo publisher, eliminare gli slot di replica, aggiornare il cluster Aurora Postgre DB, aggiornare l'istanza RDS per Postgre SQL DB e .

Gli slot di replica sono ospitati solo nel nodo publisher. Il nodo di SQL sottoscrizione Aurora Postgre in uno scenario di replica logica non ha slot da eliminare. Il processo di aggiornamento della versione SQL principale di Aurora Postgre supporta l'aggiornamento dell'abbonato a una nuova versione principale di Postgre indipendente dal nodo editore. SQL Tuttavia, il processo di aggiornamento interrompe il processo di replica e interferisce con la sincronizzazione dei dati tra il nodo editore e il nodo sottoscrittore. WAL È necessario riconnettere la replica logica tra publisher e subscriber dopo aver aggiornato il publisher, il subscriber o entrambi. Nella procedura seguente viene illustrato come determinare se la replica è stata interrotta e come risolvere il problema.

Determinazione della replica logica interrotta

È possibile determinare che il processo di replica è stato interrotto eseguendo una query sul nodo publisher o sul nodo subscriber, come indicato di seguito.

Per controllare il nodo publisher

- Utilizza `psql` per connetterti al nodo publisher e quindi esegui la query sulla funzione `pg_replication_slots`. Osserva il valore nella colonna attiva. Normalmente, la query restituisce `t` (true) per indicare che la replica è attiva. Se restituisce `f` (false), indica che la replica nel subscriber è stata interrotta.

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

Per controllare il nodo subscriber

Nel nodo subscriber è possibile verificare lo stato della replica in tre modi diversi.

- Esamina i SQL log di Postgre sul nodo sottoscrittore per trovare i messaggi di errore. Il log identifica gli errori nei messaggi che includono il codice di uscita 1, come illustrato di seguito.

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
```

```
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- Esegui la query sulla funzione pg_replication_origin. Connettiti al database sul nodo subscriber utilizzando psql ed esegui la query sulla funzione pg_replication_origin, come indicato di seguito.

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

  Se il set di risultati è vuoto, la replica è stata interrotta. Normalmente, viene restituito l'output riportato di seguito.

```
   roident |                       roname
  ---------+-------------------------------------------------
         1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
  (1 row)
```

- Esegui la query sulla funzione pglogical.show_subscription_status come illustrato nell'esempio seguente.

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
---====----------------+--------+-----------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

  Questo output mostra che la replica è stata interrotta. Lo stato è down e normalmente l'output mostra lo stato replicating.

Se il processo di replica logica è stato interrotto, è possibile riconnettere la replica seguendo questi passaggi.

Per riconnettere la replica logica tra i nodi publisher e subscriber

Per riconnettere la replica, devi prima disconnettere il subscriber dal nodo publisher e quindi riconnettere la sottoscrizione, come descritto in questi passaggi.

1. Connettiti al nodo subscriber utilizzando `psql`, come indicato di seguito.

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Disattiva la sottoscrizione utilizzando la funzione `pglogical.alter_subscription_disable`.

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. Ottieni l'identificatore del nodo publisher eseguendo la query su `pg_replication_origin`, come indicato di seguito.

```
SELECT * FROM pg_replication_origin;
 roident |               roname
---------+-------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. Utilizza la risposta restituita dal passaggio precedente con il comando `pg_replication_origin_create` per assegnare l'identificatore che può essere usato dalla sottoscrizione una volta riconnessa.

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. Attiva la sottoscrizione specificando il nome con lo stato `true`, come illustrato nell'esempio seguente.

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
 t
(1 row)
```

Controlla lo stato del nodo. Lo stato deve essere `replicating` come mostrato nell'esempio.

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+-----------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

Verifica lo stato dello slot di replica del subscriber sul nodo publisher. La colonna `active` dello slot deve restituire `t` (true) per indicare che la replica è stata riconnessa.

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
-------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
(1 row)
```

Prima di poter eseguire un aggiornamento della versione principale su un'istanza writer del cluster Aurora Postgre SQL DB RDS per un'istanza . Il processo di verifica preliminare dell'aggiornamento alla versione principale avvisa che l'aggiornamento non può procedere fino a quando gli slot non vengono rimossi.

Per identificare gli slot di replica creati utilizzando l'estensione `pglogical`, accedi a ciascun database e recupera il nome dei nodi. Quando esegui la query sul nodo subscriber, nell'output viene restituito sia il nodo publisher che il nodo subscriber, come mostrato nell'esempio seguente.

```
SELECT * FROM pglogical.node;
node_id   |     node_name
------------+------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
(2 rows)
```

Puoi ottenere i dettagli sulla sottoscrizione con la seguente query.

```
SELECT sub_name,sub_slot_name,sub_target
```

```
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

A questo punto puoi rimuovere la sottoscrizione, come indicato di seguito.

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

Dopo aver rimosso la sottoscrizione, puoi eliminare il nodo.

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
 drop_node
-----------
 t
(1 row)
```

Puoi verificare che il nodo sia stato eliminato, come indicato di seguito.

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

Riferimento sui parametri dell'estensione pglogical

Nella tabella sono illustrati i parametri associati all'estensione `pglogical`. Parametri come `pglogical.conflict_log_level` e `pglogical.conflict_resolution` vengono utilizzati per gestire i conflitti di aggiornamento. I conflitti possono emergere quando vengono apportate modifiche localmente alle stesse tabelle che hanno una sottoscrizione con il publisher. I conflitti possono verificarsi anche in altri scenari, ad esempio la replica bidirezionale o quando più subscriber eseguono la replica dallo stesso publisher. Per ulteriori informazioni, consulta Replica SQLbidirezionale di Postgree con pglogical.

Parametro	Descrizione
pglogical.batch_inserts	Esegue inserimenti batch, se possibile. Non impostato per impostazione predefinita. Imposta "1" per attivarlo, "0" per disattivarlo.
pglogical.conflict_log_level	Imposta il livello di log da utilizzare per la registrazione dei conflitti risolti. I valori di stringa supportati sono debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal, panic.
pglogical.conflict_resolution	Imposta il metodo da utilizzare per risolvere i conflitti quando sono risolvibili. I valori di stringa supportati sono error, apply_remote, keep_local, last_update_wins, first_update_wins.
pglogical.extra_connection_ options	Specifica le opzioni di connessione da aggiungere a tutte le connessioni dei nodi peer.
pglogical.synchronous_commit	Valore di commit sincrono specifico pglogical
pglogical.use_spi	Usa SPI (interfaccia di programmazione del server) anziché di basso livello per applicare le modifiche. API Imposta "1" per attivarlo, "0" per disattivarlo. Per maggiori informazioni suSPI, consulta Server Programming Interface nella documentazione di SQL Postgre.

Utilizzo dei wrapper di dati esterni supportati per Amazon Aurora Postgre Amazon for SQL

Un foreign data wrapper (FDW) è un tipo specifico di estensione che fornisce l'accesso a dati esterni. Ad esempio, l'`oracle_fdw`estensione consente all'istanza Aurora Postgre SQL DB RDS per il cluster Postgre DB

Di seguito, puoi trovare informazioni su diversi wrapper di dati esterni Postgre supportati. SQL

Argomenti

- Utilizzo dell'estensione log_fdw per accedere al registro DB utilizzando SQL

- Utilizzo dell'estensione postgres_fw per accedere a dati esterni
- Lavorare con i miei SQL database utilizzando l'estensione mysql_fdw
- Interazione con un database Oracle utilizzando l'estensione oracle_fdw
- Lavorare con i database SQL Server utilizzando l'estensione tds_fdw

Utilizzo dell'estensione log_fdw per accedere al registro DB utilizzando SQL

Il cluster Aurora Postgre SQL DB RDS supporta l'`log_fdw`estensione, che è possibile utilizzare per accedere al registro del motore di database utilizzando un'interfaccia. SQL L'estensione `log_fdw` offre due nuove funzioni che semplificano la creazione di tabelle esterne per i registri di database:

- `list_postgres_log_files` – Elenca i file nella directory dei registri di database e le dimensioni dei file in byte.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – Crea una tabella esterna per il file specificato nel database corrente.

Tutte le funzioni create da `log_fdw` sono di proprietà di `rds_superuser`. I membri del ruolo `rds_superuser` possono concedere l'accesso a queste funzioni ad altri utenti del database.

Per impostazione predefinita, i file di log vengono generati da Amazon Aurora in formato `stderr` (errore standard), come specificato nel `log_destination` parametro. Esistono solo due opzioni per questo parametro `stderr` e `csvlog` (valori separati da virgole,). CSV Se aggiungi l'`csvlog`opzione al parametro, Amazon Aurora genera entrambi `stderr` i `csvlog` log. Ciò può influire sulla capacità di archiviazione del cluster di database, quindi è necessario tenere conto degli altri parametri che influiscono sulla gestione dei log. Per ulteriori informazioni, consulta Impostazione della destinazione del registro (`stderr`, `csvlog`).

Un vantaggio della generazione dei registri `csvlog` è che l'estensione `log_fdw` consente di costruire le tabelle esterne con i dati suddivisi in diverse colonne. Per eseguire questa operazione, l'istanza deve essere associata a un gruppo parametri del database personalizzato in modo da poter modificare l'impostazione per `log_destination`. Per ulteriori informazioni su come fare, consulta .

L'esempio seguente presuppone che il parametro `log_destination` includa `cvslog`.

Per utilizzare l'estensione log_fdw

1. Installa l'estensione `log_fdw`.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Creare un server log come wrapper di dati esterno.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Selezionare tutti gli elementi da un elenco di file di registro.

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

Di seguito è riportata una risposta di esempio.

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
(4 rows)
```

4. Creare una tabella con una singola colonna "log_entry" per i file selezionato.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

La risposta non fornisce dettagli diversi da quello che la tabella ora esiste.

```
---------------------------------
(1 row)
```

5. Selezionare un campione del file di registro. Il seguente codice recupera l'ora del log e la descrizione del messaggio di errore.

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

Di seguito è riportata una risposta di esempio.

```
             log_time             |                                 message
----------------------------------
+-------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

Utilizzo dell'estensione postgres_fdw per accedere a dati esterni

È possibile accedere ai dati in una tabella su un server di database remoto con l'estensione postgres_fdw. Se si imposta una connessione remota dall'istanza database di PostgreSQL, l'accesso è disponibile anche alla replica di lettura.

Usare postgres_fdw per accedere al server remoto del database

1. Installare l'estensione postgres_fdw.

```
CREATE EXTENSION postgres_fdw;
```

2. Creare un server di dati esterni utilizzando CREATE SERVER.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. Creare una mappatura dell'utente per identificare il ruolo da utilizzare sul server remoto.

> **Important**
>
> Per oscurare la password in modo che non compaia nei registri, impostala `log_statement=none` a livello di sessione. L'impostazione a livello di parametro non comporta l'oscuramento della password.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. Creare una tabella che esegua la mappatura della tabella sul server remoto.

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

Lavorare con i miei SQL database utilizzando l'estensione mysql_fw

Per accedere a un database SQL compatibile con My dal utilizzare l'estensione. `mysql_fdw` Questo wrapper di dati esterni ti consente di lavorare con RDS My, SQL Aurora My, SQL MariaDB e altri database compatibili con My. SQL La connessione dal cluster Aurora Postgre SQL DB RDS al SQL database My viene crittografata al meglio, a seconda delle configurazioni del client e del server. Tuttavia, se lo si desidera, è possibile imporre l'utilizzo della crittografia. Per ulteriori informazioni, consulta Utilizzo della crittografia in transito con l'estensione.

L'`mysql_fdw`estensione è supportata nelle SQLversioni 15.4, 14.9, 13.12, 12.16 di Amazon Aurora Postgre, nelle versioni 14.2, 13.6 e successive di Amazon RDS . Supporta selezioni, inserimenti, aggiornamenti ed eliminazioni da e per Postgre DB in tabelle su un'istanza di database compatibile con My. RDS SQL SQL

Argomenti

- Configurazione di Aurora Postgre SQL RDS per Postgre
-
- Utilizzo della crittografia in transito con l'estensione

Configurazione di Aurora Postgre SQL RDS per Postgre

La configurazione dell'`mysql_fdw`estensione sul cluster Aurora Postgre SQL DB RDS per l'istanza DB del cluster DB My DB. SQL A tale scopo, è necessario disporre dei seguenti dettagli sull'istanza My DB: SQL

- Nome host o endpoint. Per un cluster Aurora My SQL DB RDS , puoi trovare l'endpoint utilizzando la Console. Scegliere la scheda Connectivity & security (Connettività e sicurezza) e cercare nella sezione Endpoint and port (Endpoint e porta).
- Numero della porta. Il numero di porta predefinito per My SQL è 3306.
- Nome del database. L'identificatore del database.

È inoltre necessario fornire l'accesso al gruppo di sicurezza o all'elenco di controllo degli accessi (ACL) per la SQL porta My, 3306. Sia il cluster Aurora Postgre SQL DB che il cluster Aurora My DB per l'istanza SQL DB . Se l'accesso non è configurato correttamente, quando provi a connetterti a My SQL -compatible table viene visualizzato un messaggio di errore simile al seguente:

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-
region.rds.amazonaws.com:3306' (110)
```

Nella seguente procedura, l'utente (utilizzando l'account `rds_superuser`) crea il server esterno. Quindi concede l'accesso al server esterno a specifici utenti. Questi utenti creano quindi le proprie mappature sugli account SQL utente My appropriati per lavorare con l'istanza My SQL DB.

Per utilizzare mysql_fdw per accedere a un server di database My SQL

1. Connect alla tua istanza Postgree SQL DB usando un account con il `rds_superuser` ruolo. Se hai accettato le impostazioni predefinite quando hai creato il cluster Aurora Postgre DB RDS per l'istanza Postgre SQL DB puoi connetterti utilizzando lo strumento da `postgres` riga di comando come segue: `psql`

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Installare l'estensione `mysql_fdw` come segue:

   ```
   postgres=> CREATE EXTENSION mysql_fdw;
   CREATE EXTENSION
   ```

Dopo aver installato l'estensione sul cluster Aurora Postgre SQL DB RDS , si configura il server esterno che fornisce la connessione a un database My. SQL

Per creare il server esterno

Esegui queste attività sul cluster Aurora Postgre SQL DB l'istanza Postgre DB. SQL La procedura presuppone che l'utente sia connesso come utente con i privilegi di `rds_superuser`, come `postgres`.

1. Crea un server esterno nel cluster Aurora Postgre SQL DB l'istanza di Postgre DB: SQL

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Concedere agli utenti appropriati l'accesso al server esterno. Questi dovrebbero essere utenti non amministratori, cioè utenti senza il ruolo `rds_superuser`.

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

SQLGli utenti Postgre creano e gestiscono le proprie connessioni al database My tramite il server esterno. SQL

Supponiamo di avere una tabella semplice su un'istanza Aurora Postgre DB per un' DB SQL Postgre. SQL SQLGli utenti di Aurora Postgre SQL RDS vogliono interrogare (`SELECT`) `INSERTUPDATE`, e gli elementi su quella tabella. `DELETE` Si supponga che l'`mysql_fdw`estensione sia stata creata sulla propria istanza SQL DB RDS for Postgre, come descritto nella procedura precedente. Dopo esserti connesso all'istanza SQL DB RDS for Postgre come utente con `rds_superuser` privilegi, puoi procedere con i seguenti passaggi.

1. Sull'istanza Aurora Postgre SQL DB RDS per l'istanza DB esterno:

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Concedere l'utilizzo a un utente che non dispone delle autorizzazioni `rds_superuser`, ad esempio `user1`:

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. Connect as *user1*, quindi crea una mappatura per l'SQLutente My:

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
CREATE USER MAPPING
```

4. Crea una tabella esterna collegata alla SQL tabella My:

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. Eseguire una semplice query sulla tabella esterna:

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. È possibile aggiungere, modificare e rimuovere dati dalla SQL tabella My. Per esempio:

```
test=> INSERT INTO mytab values (2, 'mango');
INSERT 0 1
```

   Eseguire nuovamente la query SELECT per visualizzare i risultati:

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
(2 rows)
```

Utilizzo della crittografia in transito con l'estensione

La connessione a My SQL da Aurora Postgre SQL RDS per Postgre SQL) per impostazione predefinita. TLS SSL Tuttavia, la connessione torna a essere non crittografata quando la configurazione di client e server differiscono. È possibile applicare la crittografia per tutte le

connessioni in uscita specificando l'opzione per i `REQUIRE SSL` miei account utente. RDS SQL Questo stesso approccio funziona anche per gli account utente MariadB e Aurora MySQL.

Per i miei account SQL utente configurati su`REQUIRE SSL`, il tentativo di connessione fallisce se non è possibile stabilire una connessione sicura.

Per applicare la crittografia per gli account utente esistenti di My SQL database, puoi utilizzare il `ALTER USER` comando. La sintassi varia a seconda della SQL versione My, come illustrato nella tabella seguente. Per ulteriori informazioni, vedere ALTERUSERMy SQL Reference Manual.

My SQL 5.7, My 8.0 SQL	Il mio 5.6 SQL
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

Per ulteriori informazioni sull'estensione `mysql_fdw`, consultare la documentazione di mysql_fdw.

Interazione con un database Oracle utilizzando l'estensione oracle_fdw

Per accedere a un database Oracle dal cluster di database Aurora PostgreSQL è possibile installare e utilizzare l'estensione `oracle_fdw`. Questa estensione è un wrapper di dati esterni per database Oracle. Per ulteriori informazioni sull'estensione, consultare la documentazione di oracle_fdw.

L'estensione `oracle_fdw` è supportata su Aurora PostgreSQL 12.7 (Amazon Aurora versione 4.2) e versioni successive.

Argomenti

- Attivazione dell'estensione oracle_fdw
- Esempio: utilizzo di un server esterno collegato a un database Amazon RDS for Oracle
- Utilizzo della crittografia in transito
- Informazioni sulla visualizzazione pg_user_mappings e sulle autorizzazioni

Attivazione dell'estensione oracle_fdw

Per utilizzare l'estensione oracle_fdw, eseguire la procedura riportata di seguito.

Come attivare l'estensione oracle_fdw

- Eseguire il seguente comando utilizzando un account con le autorizzazioni `rds_superuser`.

```
CREATE EXTENSION oracle_fdw;
```

Esempio: utilizzo di un server esterno collegato a un database Amazon RDS for Oracle

L'esempio seguente mostra l'utilizzo di un server esterno collegato a un database Amazon RDS for Oracle.

Come creare un server esterno collegato a un database RDS for Oracle

1. Annotare le seguenti informazioni sull'istanza database RDS for Oracle:
   - Endpoint
   - Porta
   - Nome del database
2. Creare un server esterno.

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. Concedere l'utilizzo a un utente che non dispone dei privilegi `rds_superuser`, ad esempio`user1`.

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. Connettersi come `user1` e creare una mappatura a un utente Oracle.

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Creare una tabella esterna collegata a una tabella Oracle.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
```

```
CREATE FOREIGN TABLE
```

6. Eseguire una query sulla tabella esterna.

```
test=>  SELECT * FROM mytab;
a
---
1
(1 row)
```

Se la query segnala il seguente errore, controllare il gruppo di sicurezza e la lista di controllo degli accessi (ACL) per assicurarsi che entrambe le istanze possano comunicare.

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

Utilizzo della crittografia in transito

PostgreSQL-to-Oracle la crittografia in transito si basa su una combinazione di parametri di configurazione client e server. Per un esempio di utilizzo di Oracle 21c, consultare Informazioni sui valori per la negoziazione di crittografia e integrità nella documentazione Oracle. Il client utilizzato per oracle_fdw su Amazon RDS è configurato `ACCEPTED` con, il che significa che la crittografia dipende dalla configurazione del server di database Oracle e utilizza Oracle Security Library (libnnz) per la crittografia.

Se il database si trova su RDS for Oracle, consultare Crittografia di rete nativa Oracle per configurare la crittografia.

Informazioni sulla visualizzazione pg_user_mappings e sulle autorizzazioni

Il catalogo PostgreSQL `pg_user_mapping` archivia la mappatura da un utente Aurora PostgreSQL all'utente in un server remoto di dati esterni. L'accesso al catalogo è limitato, ma puoi usare la visualizzazione `pg_user_mappings` per vedere le mappature. Di seguito è possibile trovare un esempio che mostra come si applicano le autorizzazioni con un database Oracle, sebbene le stesse informazioni si applichino più in generale a qualsiasi wrapper di dati esterno.

Nel seguente output sono presenti ruoli e autorizzazioni mappati su tre diversi utenti di esempio. Gli utenti `rdssu1` e `rdssu2` sono membri del ruolo `rds_superuser`, mentre `user1` non lo è. Nell'esempio viene utilizzato il metacomando `psql \du` per elencare i ruoli esistenti.

```
test=>  \du
                                                              List of roles
    Role name    |                         Attributes                         |
                 Member of
-----------------+------------------------------------------------------------
+-----------------------------------------------------------
 rdssu1          |                                                            |
 {rds_superuser}
 rdssu2          |                                                            |
 {rds_superuser}
 user1           |                                                            | {}
```

Tutti gli utenti, inclusi gli utenti che godono dei privilegi `rds_superuser`, sono autorizzati a visualizzare le proprie mappature utente (`umoptions`) nella tabella `pg_user_mappings`. Come mostrato nell'esempio seguente, quando `rdssu1` cerca di ottenere tutte le mappature utente, viene generato un errore anche se gode dei privilegi `rdssu1rds_superuser`:

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

Di seguito vengono riportati alcuni esempi.

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)
```

```
test=> SET SESSION AUTHORIZATION user1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

A causa delle differenze nell'implementazione di `information_schema._pg_user_mappings` e `pg_catalog.pg_user_mappings`, un `rds_superuser` creato manualmente richiede autorizzazioni aggiuntive per visualizzare le password in `pg_catalog.pg_user_mappings`.

Non sono necessarie autorizzazioni aggiuntive per un `rds_superuser` che desideri visualizzare le password in `information_schema._pg_user_mappings`.

Gli utenti che non dispongono del ruolo `rds_superuser` possono visualizzare le password in `pg_user_mappings` solo nelle seguenti condizioni:

- L'utente corrente è l'utente mappato e possiede il server oppure detiene il privilegio `USAGE` su di esso.
- L'utente corrente è il proprietario del server e la mappatura è per `PUBLIC`.

Lavorare con i database SQL Server utilizzando l'estensione tds_fdw

È possibile utilizzare l'SQL`tds_fdw`estensione Postgre per accedere ai database che supportano il protocollo tabular data stream (TDS), come i database Sybase e Microsoft SQL Server. Questo wrapper di dati esterno ti consente di connetterti dal tuo cluster Postgre SQL DB a database che utilizzano il protocollo, TDS incluso Amazon for Microsoft Server. RDS SQL Per ulteriori informazioni, consulta la documentazione di tds-fdw/tds_fdw su. GitHub

L'`tds_fdw`estensione è supportata nella SQL versione 13.6 e successive di Amazon Aurora Postgre.

Configurazione del SQL database Aurora Postgre per l'utilizzo dell'estensione tds_fdw

Nelle procedure seguenti, è possibile trovare un esempio di configurazione e utilizzo del cluster . SQL Prima di poterti connettere a un database SQL Server utilizzando`tds_fdw`, devi ottenere i seguenti dettagli per l'istanza:

- Nome host o endpoint. Per un'istanza di RDS for SQL Server DB, puoi trovare l'endpoint utilizzando la Console. Scegliere la scheda Connectivity & security (Connettività e sicurezza) e cercare nella sezione Endpoint and port (Endpoint e porta).
- Numero della porta. Il numero di porta predefinito per Microsoft SQL Server è 1433.
- Nome del database. L'identificatore del database.

È inoltre necessario fornire l'accesso al gruppo di sicurezza o all'elenco di controllo degli accessi (ACL) per la porta del SQL server, 1433. Sia il cluster Aurora Postgre SQL DB RDS devono accedere alla porta 1433. Se l'accesso non è configurato correttamente, quando si tenta di eseguire una query su Microsoft SQL Server viene visualizzato il seguente messaggio di errore:

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

Per utilizzare tds_fdw per connettersi a un database Server SQL

1. Connect all'istanza principale del cluster dell'istanza Postgre SQL DB utilizzando un account con il ruolo: `rds_superuser`

   ```
   psql --host=your-cluster-name-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=test --password
   ```

2. Installare l'estensione `tds_fdw`:

   ```
   test=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

Dopo aver installato l'estensione sul cluster Aurora Postgre SQL DB , si configura il server esterno.

Per creare il server esterno

Esegui queste attività sul cluster Aurora Postgre SQL DB utilizzando un account con privilegi. `rds_superuser`

1. Crea un server esterno nel cluster Aurora Postgre SQL DB l'istanza di Postgre DB: SQL

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
 (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
 'tds_fdw_testing');
CREATE SERVER
```

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
 character_set 'UTF-8');
CREATE SERVER
```

2. Concedere le autorizzazioni a un utente che non dispone del ruolo `rds_superuser`, ad esempio `user1`:

```
test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
```

3. Connect come user1 e crea una mappatura per un utente SQL del Server:

```
test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
 'sqlserveruser', password 'password');
CREATE USER MAPPING
```

4. Crea una tabella esterna collegata a una tabella SQL Server:

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. Eseguire una query sulla tabella esterna:

```
test=> SELECT * FROM mytab;
 a
---
 1
(1 row)
```

Utilizzo della crittografia in transito per la connessione

La connessione da Aurora Postgre SQL al SQL Server utilizza la crittografia in transito (TLS/SSL) a seconda della configurazione del database del Server. SQL Se il SQL Server non è configurato

per la crittografia, il SQL client RDS for Postgre che effettua la richiesta al database Server torna alla modalità non crittografataSQL.

È possibile applicare la crittografia per la connessione a RDS istanze DB del SQL server impostando il parametro. `rds.force_ssl` Per informazioni su come, consulta Forzare le connessioni all'istanza DB da utilizzare. SSL Per ulteriori informazioni suSSL/TLSconfiguration RDS for SQL Server, vedere Utilizzo SSL con un'istanza DB di Microsoft SQL Server.

Utilizzo di Trusted Language Extensions per PostgreSQL

Trusted Language Extensions per PostgreSQL è un kit di sviluppo open source per la creazione di estensioni di PostgreSQL. Ti consente di creare estensioni di PostgreSQL ad alte prestazioni ed eseguirle in sicurezza sul cluster database Aurora PostgreSQL. Con Trusted Language Extensions (TLE) per PostgreSQL, puoi creare estensioni di PostgreSQL che seguono l'approccio documentato per estendere le funzionalità di PostgreSQL. Per ulteriori informazioni, consulta Packaging Related Objects into an Extension (Creazione di pacchetti di oggetti correlati in un'estensione) nella documentazione PostgreSQL.

Uno dei principali vantaggi di TLE è che è possibile utilizzarlo in ambienti che non forniscono accesso al file system alla base dell'istanza PostgreSQL. In precedenza, l'installazione di una nuova estensione richiedeva l'accesso al file system. TLE rimuove questo vincolo. Fornisce un ambiente di sviluppo per creare nuove estensioni per qualsiasi database PostgreSQL, compresi quelli in esecuzione sui cluster database Aurora PostgreSQL.

TLE è progettato per impedire l'accesso a risorse non sicure per le estensioni create utilizzando TLE. Il suo ambiente di esecuzione limita l'impatto di qualsiasi difetto dell'estensione a una singola connessione al database. TLE inoltre offre agli amministratori di database un controllo dettagliato su chi può installare le estensioni e fornisce un modello di autorizzazioni per eseguirle.

TLE è supportato su Aurora PostgreSQL versione 14.5 e versioni successive.

L'ambiente di sviluppo e il runtime di Trusted Language Extensions sono compressi come estensione `pg_tle` di PostgreSQL versione 1.0.1. Supporta la creazione di estensioni in Perl JavaScript, Tcl, PL/pgSQL e SQL. L'estensione `pg_tle` si installa nel cluster database Aurora PostgreSQL nello stesso modo in cui si installano altre estensioni di PostgreSQL. Dopo l'impostazione di `pg_tle`, gli sviluppatori possono utilizzarlo per creare nuove estensioni di PostgreSQL, note come estensioni TLE.

Negli argomenti seguenti sono disponibili informazioni su come impostare Trusted Language Extensions e come iniziare a creare le proprie estensioni TLE.

Argomenti

- Terminology
- Requisiti per l'utilizzo di Trusted Language Extensions per Postgre SQL
-

- Panoramica delle estensioni linguistiche affidabili per Postgre SQL
-
- Eliminare le TLE estensioni da un database
- Disinstallazione di Trusted Language Extensions per Postgre SQL
- Utilizzo di hook PostgreSQL con le estensioni TLE
- Riferimento alla funzione per Trusted Language Extensions for Postgre SQL
- Riferimento sugli hook per Trusted Language Extensions per PostgreSQL

Terminology

Per comprendere meglio Trusted Language Extensions, consulta il seguente glossario dei termini usati in questo argomento.

Estensioni linguistiche affidabili per Postgre SQL

Trusted Language Extensions for Postgre SQL è il nome ufficiale del kit di sviluppo open source fornito come estensione. `pg_tle` È disponibile per l'uso su qualsiasi sistema Postgre. SQL Per ulteriori informazioni, consulta aws/pg_tle su. GitHub

Trusted Language Extensions

Trusted Language Extensions è il nome abbreviato di Trusted Language Extensions for Postgre. SQL Questo nome abbreviato e la sua abbreviazione (TLE) vengono utilizzati anche in questa documentazione.

linguaggio attendibile

Un linguaggio attendibile è un linguaggio di programmazione o di script con attributi di sicurezza specifici. Ad esempio, i linguaggi attendibili in genere limitano l'accesso al file system e l'uso di proprietà di rete specificate. Il kit di TLE sviluppo è progettato per supportare linguaggi affidabili. Postgre SQL supporta diverse lingue utilizzate per creare estensioni affidabili o non attendibili. Per un esempio, vedi Trusted and Untrusted PL/Perl nella documentazione di Postgre. SQL Quando crei un'estensione utilizzando Trusted Language Extensions, l'estensione utilizza intrinsecamente meccanismi di linguaggio attendibile.

TLEestensione

Un'TLEestensione è un'SQLestensione Postgre creata utilizzando il kit di sviluppo Trusted Language Extensions (TLE).

Requisiti per l'utilizzo di Trusted Language Extensions per Postgre SQL

Di seguito sono riportati i requisiti per la configurazione e l'utilizzo del kit di TLE sviluppo.

- Versioni Aurora Postgre RDS per SQL le versioni Extensions è supportata nella SQLversione 14.5 di Aurora Postgre RDS per le versioni Postgre 13.12 e successive 13 versioni, 14.5 .
  - Se è necessario aggiornare il cluster Aurora Postgre SQL DB, RDS per l'istanza Postgre, . SQL Aggiornamento dei cluster database Amazon Aurora PostgreSQL
  - Se non disponi ancora di un' DB del cluster Aurora DB che esegue PostgreSQL, puoi crearne una. Per ulteriori informazioni, consulta Creazione e connessione di un cluster di database Aurora PostgreSQL.
- Richiede i privilegi **`rds_superuser`**: per impostare e configurare l'estensione `pg_tle`, il ruolo utente del database deve disporre delle autorizzazioni del ruolo `rds_superuser`. Per impostazione predefinita, questo ruolo viene concesso all'`postgres`utente che crea il cluster Aurora SQL Postgre DB.
- Richiede un gruppo di parametri DB personalizzato: il cluster Aurora Postgre SQL DB RDS per l'istanza Postgre DB personalizzato. Si utilizza il gruppo di parametri DB personalizzato per l'istanza writer del cluster Aurora SQL Postgre DB.
  - Se il cluster Aurora Postgre SQL DB RDS per l'istanza SQL DB non è configurato con un gruppo di parametri DB personalizzato, è necessario crearne uno e associarlo all'istanza writer del cluster Aurora Postgre DB. SQL Per un breve riepilogo dei passaggi, consulta Creazione e applicazione di un gruppo di parametri database personalizzato.
  - Se il cluster Aurora Postgre SQL DB RDS per l'istanza Postgre DB personalizzato, è possibile configurare Trusted Language Extensions. Per informazioni dettagliate, consultare .

Creazione e applicazione di un gruppo di parametri database personalizzato

Utilizza i seguenti passaggi per creare un gruppo di parametri DB personalizzato e configurare il cluster Aurora Postgre DB RDS per l'utilizzo da parte dell'istanza SQL DB

Console

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Gruppi di parametri dal RDS menu Amazon.

3. Scegli Create parameter group (Crea gruppo di parametri).
4. Nella pagina Parameter group details (Dettagli del gruppo di parametri) immetti le seguenti informazioni.
   - Per Parameter group family (Famiglia del gruppo di parametri), scegli aurora-postgresql14.
   - Per Type (Tipo), scegli il gruppo di parametri database.
   - Per Group name (Nome gruppo), assegna al gruppo di parametri un nome significativo nel contesto delle operazioni.
   - In Description (Descrizione), immetti una descrizione utile in modo che gli altri membri del team possano trovarla facilmente.
5. Scegli Create (Crea) . Il tuo gruppo di parametri DB personalizzato viene creato nel tuo Regione AWS. È ora possibile modificare il cluster Aurora Postgre SQL DB per l' i passaggi successivi.
6. Scegli Database dal RDS menu Amazon.
7. Scegli il cluster Aurora Postgre SQL DB RDS con cui desideri utilizzarlo tra quelli elencati, TLE quindi scegli Modifica.
8. Nella pagina Modify DB cluster settings (Modifica le impostazioni del cluster database), trova Database options (Opzioni del database) e usa il selettore per scegliere il gruppo di parametri database personalizzato.
9. Per salvare la modifica seleziona Continua (Continua).
10. Scegli Applica immediatamente in modo da poter continuare a configurare il cluster Aurora Postgre SQL DB RDS per l'utilizzo dell'istanza DB SQL . TLE

Per continuare a impostare il sistema per Trusted Language Extensions, consulta .

Per ulteriori informazioni sull'utilizzo di gruppi di parametri database e cluster database, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

AWS CLI

Puoi evitare di specificare l'`--region`argomento quando usi i comandi configurando CLI AWS CLI con i tuoi valori predefiniti Regione AWS. Per ulteriori informazioni, consulta Nozioni di base sulla configurazione nel AWS Command Line Interface Guida per l'utente.

1. Usa il create-db-parameter-group AWS CLI Regione AWS. Nota che in questo passaggio crei un gruppo di parametri DB da applicare all'istanza writer del tuo cluster Aurora SQL Postgre DB.

In Linux, macOS, oppure Unix:

```
aws rds create-db-parameter-group \
  --region aws-region \
  --db-parameter-group-name custom-params-for-pg-tle \
  --db-parameter-group-family aurora-postgresql14 \
  --description "My custom DB parameter group for Trusted Language Extensions"
```

In Windows:

```
aws rds create-db-parameter-group ^
  --region aws-region ^
  --db-parameter-group-name custom-params-for-pg-tle ^
  --db-parameter-group-family aurora-postgresql14 ^
  --description "My custom DB parameter group for Trusted Language Extensions"
```

Il gruppo di parametri DB personalizzato è disponibile nel Regione AWS, quindi puoi modificare l'istanza writer del tuo cluster Aurora Postgre SQL DB affinché l'istanza Postgre SQL DB RDS utilizzi.

2. Usa il modify-db-instance AWS CLI comando per applicare il gruppo di parametri DB personalizzato all'istanza writer del cluster Aurora SQL Postgre DB. Questo comando riavvia immediatamente l'istanza attiva.

   In Linux, macOS, o Unix:

   ```
   aws rds modify-db-instance \
     --region aws-region \
     --db-instance-identifier your-writer-instance-name \
     --db-parameter-group-name custom-params-for-pg-tle \
     --apply-immediately
   ```

   In Windows:

   ```
   aws rds modify-db-instance ^
     --region aws-region ^
     --db-instance-identifier your-writer-instance-name ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --apply-immediately
   ```

Per continuare a impostare il sistema per Trusted Language Extensions, consulta .

Per ulteriori informazioni, consulta Gruppi di parametri DB per DB Amazon Aurora.

Puoi utilizzare il plugin AWS Management Console o AWS CLI per questi passaggi.

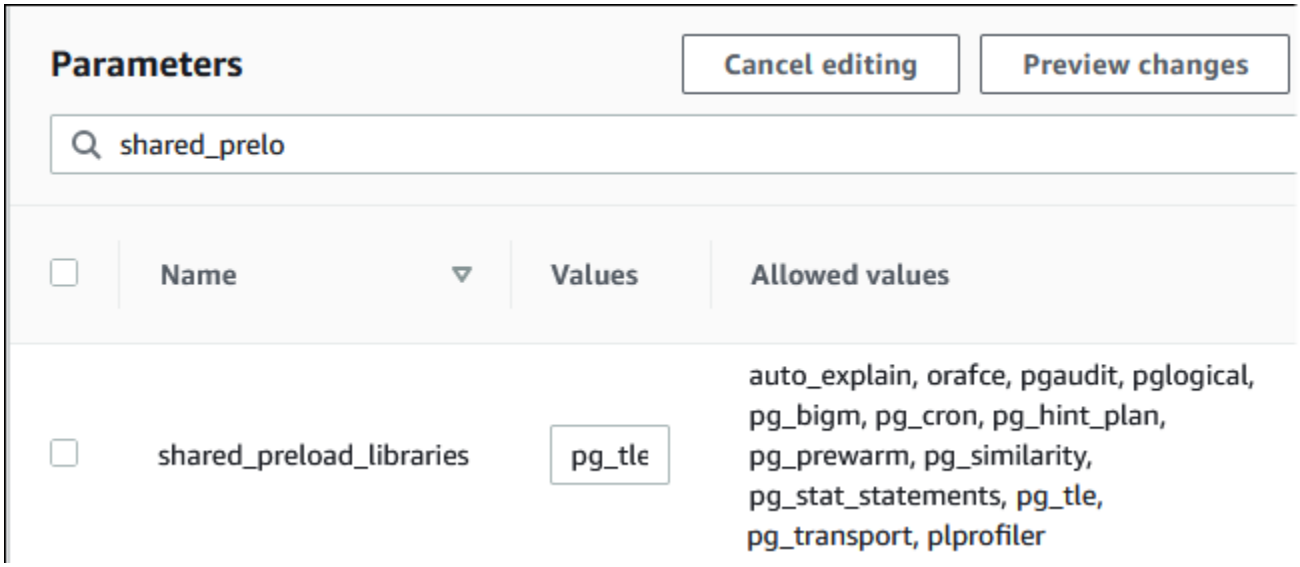
Quando si configurano Trusted Language Extensions nel cluster Aurora Postgre SQL DB RDS , le si installa in un database specifico per l'utilizzo da parte degli utenti del database che dispongono delle autorizzazioni per tale database.

Console

Per impostare Trusted Language Extensions

Esegui i seguenti passaggi utilizzando un account membro del gruppo `rds_superuser` (ruolo).

1. Accedi al AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli l'istanza Writer del cluster Aurora Postgre SQL DB l'istanza DB Postgre. SQL
3. Apri la scheda Configurazione per la tua istanza di masterizzatore di cluster Aurora Postgre SQL DB. Tra i dettagli dell'istanza, individua il collegamento Gruppo di parametri.
4. Scegli il link per aprire i parametri personalizzati associati al tuo cluster Aurora SQL Postgre DB.
5. Nel campo di ricerca Parametri, digita `shared_pre` per trovare il parametro `shared_preload_libraries`.
6. Scegli Edit parameters (Modifica parametri) per accedere ai valori delle proprietà.
7. Aggiungi `pg_tle` all'elenco nel campo Values (Valori). Utilizza una virgola per separare gli elementi nell'elenco di valori.



8. Riavviare l'istanza writer del in modo che la modifica al parametro abbia effetto. `shared_preload_libraries`
9. Quando l'istanza è disponibile, verifica che `pg_tle` sia stato inizializzato. Usalo `psql` per connetterti all'istanza writer del tuo cluster Aurora Postgre SQL DB, RDS per l'istanza Postgre SQL DB, seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. Con `pg_tle` inizializzato, puoi ora creare l'estensione.

```
CREATE EXTENSION pg_tle;
```

Per verificare che l'estensione sia installata, puoi usare il seguente metacomando `psql`.

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+-------------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. Concedi il `pgtle_admin` ruolo al nome utente principale che hai creato per il tuo cluster Aurora Postgre SQL DB RDS per l'istanza SQL DB al momento della configurazione. Se hai accettato l'impostazione predefinita, il valore è `postgres`.

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

Per verificare se la concessione è avvenuta, utilizza il metacomando `psql` come illustrato nell'esempio seguente. Nell'output vengono visualizzati solo i ruoli `pgtle_admin` e `postgres`. Per ulteriori informazioni, consulta Informazioni su ruoli e autorizzazioni di PostgreSQL.

```
labdb=> \du
                          List of roles
    Role name    |           Attributes            |               Member of
-----------------+---------------------------------
+-----------------------------------
pgtle_admin     | Cannot login                     | {}
postgres        | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                | Password valid until infinity    |...
```

12. Chiudi la sessione `psql` usando il metacomando `\q`.

```
\q
```

Per iniziare a creare estensioni, consulta. TLE Esempio: creazione di un'estensione linguistica affidabile utilizzando SQL

AWS CLI

È possibile evitare di specificare l'`--region`argomento quando si utilizzano CLI i comandi configurando AWS CLI con i tuoi valori predefiniti Regione AWS. Per ulteriori informazioni, consulta Nozioni di base sulla configurazione nel AWS Command Line Interface Guida per l'utente.

Per impostare Trusted Language Extensions

1. Usa il modify-db-parameter-group AWS CLI comando da aggiungere `pg_tle` al `shared_preload_libraries` parametro.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
```

```
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Usa il reboot-db-instance AWS CLI comando per riavviare l'istanza writer del RDS cluster Aurora Postgre DB per l'istanza SQL Postgre DB e SQL `pg_tle`

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Quando l'istanza è disponibile, verifica che `pg_tle` sia stato inizializzato. Usalo `psql` per connetterti all'istanza writer del tuo cluster Aurora Postgre SQL DB, RDS per l'istanza Postgre SQL DB, seguente.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

   Con `pg_tle` inizializzato, puoi ora creare l'estensione.

```
CREATE EXTENSION pg_tle;
```

4. Concedi il `pgtle_admin` ruolo al nome utente principale che hai creato per il tuo cluster Aurora Postgre SQL DB RDS per l'istanza SQL DB al momento della configurazione. Se hai accettato l'impostazione predefinita, il valore è `postgres`.

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

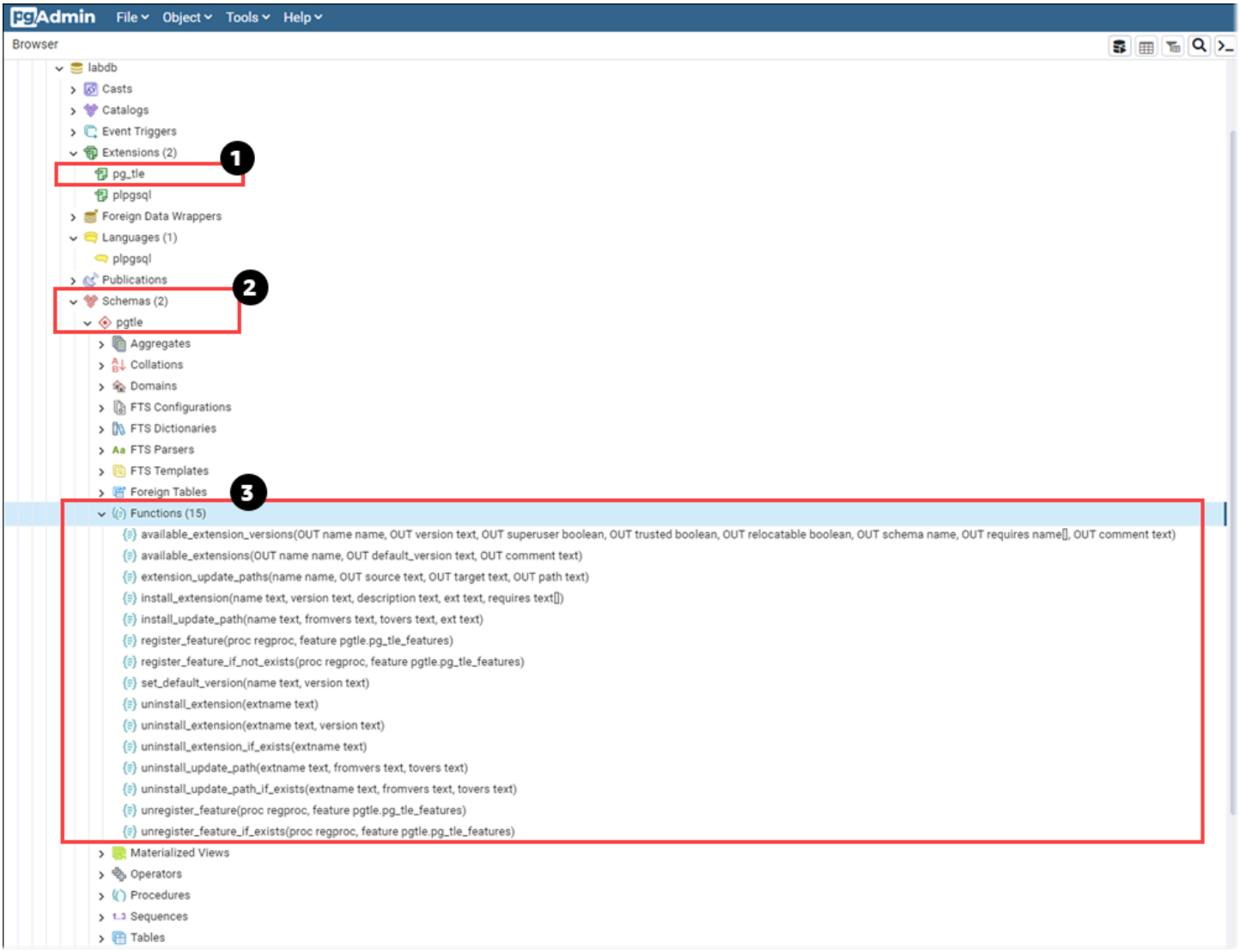
5. Chiudi la sessione `psql` come indicato di seguito.

```
labdb=> \q
```

Per iniziare a creare estensioni, consulta. TLE Esempio: creazione di un'estensione linguistica affidabile utilizzando SQL

Panoramica delle estensioni linguistiche affidabili per Postgre SQL

Trusted Language Extensions for Postgre SQL è un'SQLestensione Postgre che si installa nel cluster Aurora Postgre DB RDS per l'istanza Postgre SQL DB . SQL Nell'immagine seguente di un database di esempio nello strumento pgAdmin client, puoi visualizzare alcuni dei componenti che compongono l'estensione. `pg_tle`



È possibile vedere i dettagli riportati di seguito.

1. Il kit di SQL sviluppo Trusted Language Extensions (TLE) per Postgre viene fornito come estensione. `pg_tle` Pertanto, `pg_tle` viene aggiunto alle estensioni disponibili per il database in cui è installato.
2. TLEha il suo schema,. `pgtle` Questo schema contiene funzioni helper (3) per l'installazione e la gestione delle estensioni create.

3. TLEfornisce oltre una dozzina di funzioni di supporto per l'installazione, la registrazione e la gestione delle estensioni. Per ulteriori informazioni su queste funzioni, consulta Riferimento alla funzione per Trusted Language Extensions for Postgre SQL.

Altri componenti dell'estensione `pg_tle` sono:

- Il ruolo **`pgtle_admin`**: il ruolo `pgtle_admin` viene creato quando viene installata l'estensione `pg_tle`. Questo ruolo include privilegi e deve essere trattato come tale. Ti consigliamo vivamente di seguire il principio del privilegio minimo quando concedi il ruolo `pgtle_admin` agli utenti del database. In altre parole, concedi il `pgtle_admin` ruolo solo agli utenti del database autorizzati a creare, installare e gestire nuove TLE estensioni, come. `postgres`
- La **`pgtle.feature_info`** tabella: la `pgtle.feature_info` tabella è una tabella protetta che contiene informazioni TLEs sugli hook e sulle stored procedure e funzioni personalizzate da essi utilizzate. Se disponi di privilegi `pgtle_admin`, usa le seguenti funzioni Trusted Language Extensions per aggiungere e aggiornare le informazioni nella tabella.
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

È possibile installare tutte le estensioni con cui si crea TLE in qualsiasi cluster Aurora Postgre SQL DB RDS per un'istanza di Postgre DB `pg_tle` L'`pg_tle`estensione è limitata al database SQL Postgre in cui è installata. Le estensioni che crei utilizzando TLE hanno come ambito lo stesso database.

Usa le varie `pgtle` funzioni per installare il codice che compone l'TLEestensione. Le seguenti funzioni di Trusted Language Extensions richiedono tutte il ruolo `pgtle_admin`.

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)

- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

Esempio: creazione di un'estensione linguistica affidabile utilizzando SQL

L'esempio seguente mostra come creare un'TLEestensione denominata `pg_distance` che contiene alcune SQL funzioni per il calcolo delle distanze utilizzando formule diverse. Nell'elenco, puoi trovare la funzione per il calcolo della distanza di Manhattan e la funzione per il calcolo della distanza euclidea. Per ulteriori informazioni sulla differenza tra queste formule, consulta Taxicab geometry (Geometria del taxi) e Euclidean geometry (Geometria euclidea) in Wikipedia.

È possibile utilizzare questo esempio nel proprio cluster Aurora Postgre SQL DB RDS se l'`pg_tle`estensione è configurata come descritto in.

> **Note**
>
> È necessario disporre dei privilegi del ruolo `pgtle_admin` per seguire questa procedura.

Per creare l'estensione di esempio TLE

I passaggi seguenti utilizzano un database di esempio denominato `labdb`. Questo database è di proprietà dell'utente `postgres` principale. Il ruolo `postgres` dispone anche delle autorizzazioni del ruolo `pgtle_admin`.

1. Utilizzalo `psql` per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Crea un'TLEestensione denominata `pg_distance` copiando il codice seguente e incollandolo nella console di sessione. `psql`

   ```
   SELECT pgtle.install_extension
   ```

```
(
 'pg_distance',
 '0.1',
  'Distance functions for two points',
$_pg_tle_$
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;

    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

Viene visualizzato l'output riportato di seguito.

```
install_extension
---------------
 t
(1 row)
```

Gli artefatti che costituiscono l'estensione `pg_distance` sono ora installati nel database. Questi artefatti includono il file di controllo e il codice dell'estensione, che devono essere presenti in modo che l'estensione possa essere creata utilizzando il comando `CREATE EXTENSION`. In altre parole, è comunque necessario creare l'estensione per rendere le funzioni disponibili agli utenti del database.

3. Per creare l'estensione, usa il comando `CREATE EXTENSION` come per qualsiasi altra estensione. Come per altre estensioni, l'utente del database deve disporre delle autorizzazioni `CREATE` nel database.

```
CREATE EXTENSION pg_distance;
```

4. Per testare l'`pg_distance`TLEestensione, puoi usarla per calcolare la distanza da Manhattan tra quattro punti.

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

Per calcolare la distanza euclidea tra lo stesso set di punti, puoi usare quanto segue.

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
5.656854249492381
```

L'estensione `pg_distance` carica le funzioni nel database e le rende disponibili a tutti gli utenti con le autorizzazioni per il database.

Modificare l'estensione TLE

Per migliorare le prestazioni delle query per le funzioni incluse in questa TLE estensione, aggiungi i seguenti due SQL attributi Postgre alle relative specifiche.

- `IMMUTABLE`: l'attributo `IMMUTABLE` garantisce che l'ottimizzatore di query possa utilizzare le ottimizzazioni per migliorare i tempi di risposta delle query. Per ulteriori informazioni, consulta le categorie di volatilità delle funzioni nella documentazione di Postgre. SQL
- `PARALLEL SAFE`— L'`PARALLEL SAFE`attributo è un altro attributo che consente a Postgre SQL di eseguire la funzione in modalità parallela. Per ulteriori informazioni, consulta la documentazione CREATEFUNCTIONdi SQL Postgre.

Nell'esempio seguente, puoi vedere come viene utilizzata la `pgtle.install_update_path` funzione per aggiungere questi attributi a ciascuna funzione per creare una versione `0.2` dell'`pg_distance`TLEestensione. Per ulteriori informazioni su questa funzione, consulta pgtle.install_update_path. È necessario avere il ruolo `pgtle_admin` necessario per eseguire questa operazione.

Per aggiornare un'TLEestensione esistente e specificare la versione predefinita

1. Connect all'istanza writer del cluster Aurora Postgre SQL DB RDS . pgAdmin

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. Modifica l'TLEestensione esistente copiando il codice seguente e incollandolo nella console di sessione. `psql`

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

Viene visualizzata una risposta simile alla seguente.

```
install_update_path
---------------------
 t
(1 row)
```

È possibile impostare questa versione dell'estensione come versione predefinita, in modo che gli utenti del database non debbano specificare una versione quando creano o aggiornano l'estensione nel database.

3. Per specificare che la versione modificata (versione 0.2) dell'TLEestensione è la versione predefinita, utilizzate la `pgtle.set_default_version` funzione come illustrato nell'esempio seguente.

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

Per ulteriori informazioni su questa funzione, consulta pgtle.set_default_version.

4. Una volta installato il codice, è possibile aggiornare l'TLEestensione installata nel modo consueto, utilizzando il `ALTER EXTENSION ... UPDATE` comando, come illustrato di seguito:

```
ALTER EXTENSION pg_distance UPDATE;
```

Eliminare le TLE estensioni da un database

Puoi eliminare TLE le tue estensioni usando il `DROP EXTENSION` comando nello stesso modo in cui lo fai per le altre estensioni SQL Postgre. L'eliminazione dell'estensione non rimuove i file di installazione che costituiscono l'estensione, il che consente agli utenti di ricrearla. Per rimuovere l'estensione e i relativi file di installazione, esegui la seguente procedura in due passaggi.

Per eliminare l'TLEestensione e rimuovere i relativi file di installazione

1. Usa `psql` o un altro strumento client per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=dbname
```

2. Elimina l'estensione come faresti con qualsiasi estensione SQL Postgre.

```
DROP EXTENSION your-TLE-extension
```

Ad esempio, se crei l'estensione `pg_distance` come descritto in Esempio: creazione di un'estensione linguistica affidabile utilizzando SQL, puoi eliminarla come segue.

```
DROP EXTENSION pg_distance;
```

Viene visualizzato l'output che conferma che l'estensione è stata eliminata, come segue.

```
DROP EXTENSION
```

A questo punto, l'estensione non è più attiva nel database. Tuttavia, i file di installazione e il file di controllo sono ancora disponibili nel database, quindi gli utenti del database possono creare nuovamente l'estensione, se lo desiderano.

- Se vuoi lasciare intatti i file di estensione in modo che gli utenti del database possano creare la tua TLE estensione, puoi fermarti qui.
- Se desideri rimuovere tutti i file che costituiscono l'estensione, continua con il passaggio successivo.

3. Per rimuovere tutti i file di installazione per l'estensione, usa la funzione `pgtle.uninstall_extension`. Questa funzione rimuove tutto il codice e i file di controllo dell'estensione.

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

Ad esempio, per rimuovere tutti i file di installazione `pg_distance`, utilizza il comando seguente.

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

Disinstallazione di Trusted Language Extensions per Postgre SQL

Se non desideri più creare TLE le tue estensioni utilizzandoTLE, puoi eliminare l'`pg_tle`estensione e rimuovere tutti gli artefatti. Questa azione include l'eliminazione di eventuali TLE estensioni dal database e l'eliminazione dello schema. `pgtle`

Per eliminare l'estensione **`pg_tle`** e il relativo schema da un database

1. Usa `psql` o un altro strumento client per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password --dbname=dbname
   ```

2. Elimina l'estensione `pg_tle` dal database. Se il database ha TLE le tue estensioni ancora in esecuzione nel database, devi eliminare anche quelle estensioni. A questo scopo, puoi utilizzare la parola chiave `CASCADE`, come illustrato di seguito.

   ```
   DROP EXTENSION pg_tle CASCADE;
   ```

   Se l'estensione `pg_tle` non è ancora attiva nel database, non è necessario utilizzare la parola chiave `CASCADE`.

3. Elimina lo schema `pgtle`. Questa azione rimuove tutte le funzioni di gestione dal database.

   ```
   DROP SCHEMA pgtle CASCADE;
   ```

   Il comando restituisce quanto segue al termine del processo.

   ```
   DROP SCHEMA
   ```

   L'estensione `pg_tle`, il relativo schema, le funzioni e tutti gli artefatti vengono rimossi. Per creare nuove estensioni utilizzandoTLE, ripeti la procedura di configurazione. Per ulteriori informazioni, consulta .

Utilizzo di hook PostgreSQL con le estensioni TLE

Un hook è un meccanismo di callback disponibile in PostgreSQL che consente agli sviluppatori di chiamare funzioni personalizzate o altre routine durante le normali operazioni del database. Il kit di sviluppo TLE supporta gli hook PostgreSQL per poter integrare le funzioni personalizzate con il comportamento di PostgreSQL in fase di esecuzione. Ad esempio, puoi utilizzare un hook per associare il processo di autenticazione al codice personalizzato o per modificare il processo di pianificazione ed esecuzione delle query in base alle tue esigenze specifiche.

Le estensioni TLE possono utilizzare gli hook. Se l'ambito dell'hook è globale, si applica a tutti i database. Pertanto, se l'estensione TLE utilizza un hook globale, è necessario crearla in tutti i database a cui gli utenti possono accedere.

Quando usi `pg_tle` per creare le estensioni Trusted Language Extensions, puoi utilizzare gli hook disponibili da un'API SQL per creare le funzioni della tua estensione. È necessario registrare gli hook con `pg_tle`. Per alcuni hook, potrebbe essere necessario impostare anche vari parametri di configurazione. Ad esempio, l'hook di controllo `passcode` può essere impostato su attivo, disattivo oppure obbligatorio. Per ulteriori informazioni sui requisiti specifici per gli hook `pg_tle` disponibili, consulta Riferimento sugli hook per Trusted Language Extensions per PostgreSQL.

Esempio: creazione di un'estensione che utilizza un hook PostgreSQL

L'esempio discusso in questa sezione utilizza un hook PostgreSQL per controllare la password fornita durante specifiche operazioni SQL e impedisce agli utenti del database di impostare le proprie password su una qualsiasi di quelle contenute nella tabella `password_check.bad_passwords`. La tabella contiene le prime dieci opzioni di password più utilizzate, ma facilmente violabili.

Per configurare questo esempio nel cluster database Aurora PostgreSQL, devi aver già installato Trusted Language Extensions. Per informazioni dettagliate, consultare .

Per configurare l'esempio dell'hook di controllo della password

1. Utilizza `psql` per connetterti all'istanza di scrittura del cluster database Aurora PostgreSQL.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. Copia il codice da Codice di hook di controllo della password e incollalo nel database.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
```

```
   ('123456'),
   ('password'),
   ('12345678'),
   ('qwerty'),
   ('123456789'),
   ('12345'),
   ('1234'),
   ('111111'),
   ('1234567'),
   ('dragon');
 CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

 CREATE FUNCTION password_check.passcheck_hook(username text, password text,
password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
 RETURNS void AS $$
   DECLARE
     invalid bool := false;
   BEGIN
     IF password_type = 'PASSWORD_TYPE_MD5' THEN
       SELECT EXISTS(
         SELECT 1
         FROM password_check.bad_passwords bp
         WHERE ('md5' || md5(bp.plaintext || username)) = password
       ) INTO invalid;
       IF invalid THEN
         RAISE EXCEPTION 'Cannot use passwords from the common password
dictionary';
       END IF;
     ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
       SELECT EXISTS(
         SELECT 1
         FROM password_check.bad_passwords bp
         WHERE bp.plaintext = password
       ) INTO invalid;
       IF invalid THEN
         RAISE EXCEPTION 'Cannot use passwords from the common password
dictionary';
       END IF;
     END IF;
   END
 $$ LANGUAGE plpgsql SECURITY DEFINER;

 GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;
```

```
  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Quando l'estensione è stata caricata nel database, viene visualizzato un output come il seguente.

```
 install_extension
-------------------
 t
(1 row)
```

3. Mentre sei ancora connesso al database, puoi creare l'estensione.

```
CREATE EXTENSION my_password_check_rules;
```

4. È possibile confermare che l'estensione è stata creata nel database utilizzando il seguente metacomando `psql`.

```
\dx
                        List of installed extensions
          Name           | Version |   Schema   |
 Description
------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. Apri un'altra sessione di terminale per lavorare con AWS CLI. È necessario modificare il gruppo di parametri database personalizzato per attivare l'hook di controllo della password. A tale scopo, utilizzate il comando modify-db-parameter-groupCLI come illustrato nell'esempio seguente.

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Per rendere effettive le modifiche all'impostazione del gruppo di parametri possono essere necessari alcuni minuti. Tuttavia, questo parametro è dinamico, quindi non è necessario riavviare l'istanza di scrittura del cluster database Aurora PostgreSQL perché l'impostazione diventi effettiva.

6. Apri la sessione `psql` ed esegui una query sul database per verificare che l'hook di controllo della password sia stato attivato.

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
----------------------------
on
(1 row)
```

L'hook di controllo della password è ora attivo. Puoi testarlo creando un nuovo ruolo e utilizzando una delle password errate, come illustrato nell'esempio seguente.

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

L'output è stato formattato per ragioni di leggibilità.

L'esempio seguente mostra che il comportamento `\password` del metacomando interattivo `pgsql` è influenzato anche dall'hook di controllo della password.

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
```

```
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

Puoi eliminare questa estensione TLE e disinstallare i file di origine, se lo desideri. Per ulteriori informazioni, consulta Eliminare le TLE estensioni da un database.

Codice di hook di controllo della password

Il codice di esempio mostrato qui definisce le specifiche per l'estensione TLE `my_password_check_rules`. Quando copi questo codice e lo incolli nel database, il codice dell'estensione `my_password_check_rules` viene caricato nel database e l'hook `password_check` viene registrato per essere utilizzato dall'estensione.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
```

```
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Riferimento alla funzione per Trusted Language Extensions for Postgre SQL

Visualizza la seguente documentazione di riferimento sulle funzioni disponibili in Trusted Language Extensions for SQL Postgre. Utilizzate queste funzioni per installare, registrare, aggiornare e gestire TLEle estensioni, ovvero le estensioni Postgre sviluppate utilizzando il kit di sviluppo Trusted Language SQL Extensions.

Funzioni

- pgtle.available_extensions
- pgtle.available_extension_versions

- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

La funzione `pgtle.available_extensions` è progettata per restituire un set. Restituisce tutte TLE le estensioni disponibili nel database. Ogni riga restituita contiene informazioni su una singola TLE estensione.

Prototipo di funzione

```
pgtle.available_extensions()
```

Ruolo

Nessuna.

Argomenti

Nessuna.

Output

- `name`— Il nome dell'TLEestensione.

- `default_version`— La versione dell'TLEestensione da utilizzare quando `CREATE EXTENSION` viene chiamata senza specificare una versione.
- `description`— Una descrizione più dettagliata dell'TLEestensione.

Esempio di utilizzo

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

La funzione `available_extension_versions` è progettata per restituire un set. Restituisce un elenco di tutte le TLE estensioni disponibili e delle relative versioni. Ogni riga contiene informazioni su una versione specifica dell'TLEestensione specificata, incluso se richiede un ruolo specifico.

Prototipo di funzione

```
pgtle.available_extension_versions()
```

Ruolo

Nessuna.

Argomenti

Nessuna.

Output

- `name`— Il nome dell'TLEestensione.
- `version`— La versione dell'TLEestensione.
- `superuser`— Questo valore è sempre valido `false` per TLE le tue estensioni. Le autorizzazioni necessarie per creare l'TLEestensione o aggiornarla sono le stesse necessarie per creare altri oggetti nel database specificato.
- `trusted`— Questo valore è sempre `false` per un'TLEestensione.
- `relocatable`— Questo valore è sempre `false` per un'TLEestensione.
- `schema`— Speciifica il nome dello schema in cui è installata l'TLEestensione.
- `requires`— Un array contenente i nomi di altre estensioni necessarie a questa TLE estensione.

- `description`— Una descrizione dettagliata dell'TLEestensione.

Per ulteriori informazioni sui valori di output, vedete Imballaggio di oggetti correlati in un'estensione > File di estensione nella documentazione di Postgre. SQL

Esempio di utilizzo

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

La funzione `extension_update_paths` è progettata per restituire un set. Restituisce un elenco di tutti i possibili percorsi di aggiornamento per un'TLEestensione. Ogni riga include gli aggiornamenti o i downgrade disponibili per quell'estensione. TLE

Prototipo di funzione

```
pgtle.extension_update_paths(name)
```

Ruolo

Nessuna.

Argomenti

`name`— Il nome dell'TLEestensione da cui ottenere i percorsi di aggiornamento.

Output

- `source`: la versione di origine di un aggiornamento.
- `target`: la versione di destinazione di un aggiornamento.
- `path`— Il percorso di aggiornamento utilizzato per aggiornare un'TLEestensione da una `source` `target` versione all'altra, ad esempio`0.1--0.2`.

Esempio di utilizzo

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

La `install_extension` funzione consente di installare gli artefatti che compongono l'TLEestensione nel database, dopodiché può essere creata utilizzando il comando. `CREATE EXTENSION`

Prototipo di funzione

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

Ruolo

Nessuna.

Argomenti

- `name`— Il nome dell'estensione. TLE Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `version`— La versione dell'TLEestensione.
- `description`— Una descrizione dettagliata dell'TLEestensione. Questa descrizione viene visualizzata nel campo `comment` in `pgtle.available_extensions()`.
- `ext`— Il contenuto dell'TLEestensione. Questo valore include gli oggetti, come le funzioni.
- `requires`— Un parametro opzionale che specifica le dipendenze per questa TLE estensione. L'estensione `pg_tle` viene aggiunta automaticamente come dipendenza.

Molti di questi argomenti sono gli stessi inclusi in un file di controllo delle estensioni per l'installazione di un'estensione Postgre SQL sul file system di un'istanza Postgre. SQL Per ulteriori informazioni, consultate la sezione File di estensione nella creazione di pacchetti di oggetti correlati in un'estensione nella documentazione di Postgre. SQL

Output

Questa funzione restituisce `OK` in caso di esito positivo e `NULL` in caso di errore.

- `OK`— L'TLEestensione è stata installata correttamente nel database.
- `NULL`— L'TLEestensione non è stata installata correttamente nel database.

Esempio di utilizzo

```
SELECT pgtle.install_extension(
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

La `install_update_path` funzione fornisce un percorso di aggiornamento tra due diverse versioni di un'TLEestensione. Questa funzione consente agli utenti dell'TLEestensione di aggiornarne la versione utilizzando la `ALTER EXTENSION ... UPDATE` sintassi.

Prototipo di funzione

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

Ruolo

`pgtle_admin`

Argomenti

- `name`— Il nome dell'TLEestensione. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `fromvers`— La versione sorgente dell'TLEestensione per l'aggiornamento.
- `tovers`— La versione di destinazione dell'TLEestensione per l'aggiornamento.
- `ext`: i contenuti dell'aggiornamento. Questo valore include gli oggetti, come le funzioni.

Output

Nessuna.

Esempio di utilizzo

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

La `register_feature` funzione aggiunge la SQL funzionalità interna di Postgre specificata alla tabella. `pgtle.feature_info` Gli SQL hook Postgre sono un esempio di funzionalità interna di Postgre. SQL Il kit di sviluppo Trusted Language Extensions supporta l'uso degli hook Postgre. SQL Attualmente, questa funzione supporta la seguente funzionalità.

- `passcheck`— Registra l'hook per il controllo delle password con la procedura o la funzione che personalizza il comportamento di controllo delle password di Postgre. SQL

Prototipo di funzione

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome di una procedura o funzione memorizzata da utilizzare per la funzionalità.
- `feature`: il nome della funzionalità `pg_tle` (ad esempio `passcheck`) da registrare con la funzione.

Output

Nessuna.

Esempio di utilizzo

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

La `pgtle.register_feature_if_not_exists` funzione aggiunge la funzionalità Postgre specificata alla `pgtle.feature_info` tabella e identifica l'TLEestensione o altra procedura o SQL funzione che utilizza la funzione. Per ulteriori informazioni sugli hook e su Trusted Language Extensions, consulta Utilizzo di hook PostgreSQL con le estensioni TLE.

Prototipo di funzione

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`— Il nome di una procedura o funzione memorizzata che contiene la logica (codice) da utilizzare come funzionalità per l'estensione. TLE Ad esempio, il codice `pw_hook`.
- `feature`— Il nome della funzionalità Postgre da registrare per la SQL TLE funzione. Attualmente, l'unica funzionalità disponibile è l'hook `passcheck`. Per ulteriori informazioni, consulta Hook di verifica della password (passcheck).

Output

Restituisce `true` dopo aver registrato la funzionalità per l'estensione specificata. Restituisce `false` se la funzionalità è già registrata.

Esempio di utilizzo

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

La `set_default_version` funzione consente di specificare una `default_version` per l'TLEestensione. È possibile utilizzare questa funzione per definire un percorso di aggiornamento

e designare la versione come predefinita per l'TLEestensione. Quando gli utenti TLE del database specificano l'estensione nei `ALTER EXTENSION ... UPDATE` comandi `CREATE EXTENSION` and, quella versione dell'TLEestensione viene creata nel database per quell'utente.

Questa funzione restituisce `true` in caso di esito positivo. Se l'TLEestensione specificata nell'`name`argomento non esiste, la funzione restituisce un errore. Analogamente, se `version` l'TLEestensione non esiste, restituisce un errore.

Prototipo di funzione

```
pgtle.set_default_version(name text, version text)
```

Ruolo

`pgtle_admin`

Argomenti

- `name`— Il nome dell'TLEestensione. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `version`— La versione dell'TLEestensione per impostare l'impostazione predefinita.

Output

- `true`: quando l'impostazione della versione predefinita ha esito positivo, la funzione restituisce `true`.
- `ERROR`— Restituisce un messaggio di errore se non esiste un'TLEestensione con il nome o la versione specificati.

Esempio di utilizzo

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension(name)

La `uninstall_extension` funzione rimuove tutte le versioni di un'TLEestensione da un database. Questa funzione impedisce alle future chiamate `CREATE EXTENSION` di installare l'TLEestensione. Se l'TLEestensione non esiste nel database, viene generato un errore.

La `uninstall_extension` funzione non eliminerà un'TLEestensione attualmente attiva nel database. Per rimuovere un'TLEestensione attualmente attiva, devi chiamare esplicitamente `DROP EXTENSION` per rimuoverla.

Prototipo di funzione

```
pgtle.uninstall_extension(extname text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`— Il nome dell'TLEestensione da disinstallare. Questo nome è lo stesso usato `CREATE EXTENSION` per caricare l'TLEestensione da utilizzare in un determinato database.

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension(name, version)

La `uninstall_extension(name, version)` funzione rimuove la versione specificata dell'TLEestensione dal database. Questa funzione impedisce `CREATE EXTENSION` e `ALTER EXTENSION` impedisce l'installazione o l'aggiornamento di un'TLEestensione alla versione specificata. Questa funzione rimuove anche tutti i percorsi di aggiornamento per la versione specificata dell'TLEestensione. Questa funzione non disinstallerà l'TLEestensione se è attualmente attiva nel database. È necessario chiamare in modo esplicito `DROP EXTENSION` per rimuovere l'TLEestensione. Per disinstallare tutte le versioni di un'TLEestensione, consultapgtle.uninstall_extension(name).

Prototipo di funzione

```
pgtle.uninstall_extension(extname text, version text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`— Il nome dell'TLEestensione. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `version`— La versione dell'TLEestensione da disinstallare dal database.

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

La `uninstall_extension_if_exists` funzione rimuove tutte le versioni di un'TLEestensione da un determinato database. Se l'TLEestensione non esiste, la funzione ritorna silenziosamente (non viene generato alcun messaggio di errore). Se l'estensione specificata è attualmente attiva in un database, non viene eliminata dalla funzione. È necessario chiamare esplicitamente `DROP EXTENSION` per rimuovere l'TLEestensione prima di utilizzare questa funzione per disinstallarne gli artefatti.

Prototipo di funzione

```
pgtle.uninstall_extension_if_exists(extname text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`— Il nome dell'estensione. TLE Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.

Output

La funzione `uninstall_extension_if_exists` restituisce `true` dopo aver disinstallato l'estensione specificata. Se l'estensione specificata non esiste, la funzione restituisce `false`.

- `true`— Restituisce `true` dopo aver disinstallato l'TLEestensione.
- `false`— Restituisce `false` quando l'TLEestensione non esiste nel database.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

La `uninstall_update_path` funzione rimuove il percorso di aggiornamento specifico da un'TLEestensione. In tal modo `ALTER EXTENSION ... UPDATE TO` non può utilizzarlo come percorso di aggiornamento.

Se l'TLEestensione è attualmente utilizzata da una delle versioni in questo percorso di aggiornamento, rimane nel database.

Se il percorso di aggiornamento specificato non esiste, la funzione genera un errore.

Prototipo di funzione

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`— Il nome dell'TLEestensione. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `fromvers`— La versione sorgente dell'TLEestensione utilizzata nel percorso di aggiornamento.
- `tovers`— La versione di destinazione dell'TLEestensione utilizzata nel percorso di aggiornamento.

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

La `uninstall_update_path_if_exists` funzione è simile `uninstall_update_path` in quanto rimuove il percorso di aggiornamento specificato da un'TLEestensione. Tuttavia, se il percorso di aggiornamento non esiste, questa funzione non genera un messaggio di errore e restituisce `false`.

Prototipo di funzione

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

Ruolo

`pgtle_admin`

Argomenti

- `extname`— Il nome dell'TLEestensione. Questo valore viene utilizzato per la chiamata `CREATE EXTENSION`.
- `fromvers`— La versione sorgente dell'TLEestensione utilizzata nel percorso di aggiornamento.
- `tovers`— La versione di destinazione dell'TLEestensione utilizzata nel percorso di aggiornamento.

Output

- `true`— La funzione ha aggiornato correttamente il percorso dell'TLEestensione.
- `false`— La funzione non è stata in grado di aggiornare il percorso dell'TLEestensione.

Esempio di utilizzo

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

La funzione `unregister_feature` fornisce un modo per rimuovere le funzioni registrate per utilizzare la funzionalità `pg_tle`, ad esempio gli hook. Per ulteriori informazioni sulla registrazione di una funzionalità, consulta pgtle.register_feature.

Prototipo di funzione

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome di una funzione memorizzata da registrare con una funzionalità `pg_tle`.
- `feature`: il nome della funzionalità `pg_tle` da registrare con la funzione. Ad esempio, `passcheck` è una funzionalità che può essere registrata per essere utilizzata dalle estensioni Trusted Language Extensions sviluppate. Per ulteriori informazioni, consulta Hook di verifica della password (passcheck).

Output

Nessuna.

Esempio di utilizzo

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

La funzione `unregister_feature` fornisce un modo per rimuovere le funzioni registrate per utilizzare la funzionalità `pg_tle`, ad esempio gli hook. Per ulteriori informazioni, consulta Utilizzo di hook PostgreSQL con le estensioni TLE. Restituisce `true` dopo aver completato l'annullamento della registrazione della funzionalità. Restituisce `false` se la funzionalità non è stata registrata.

Per informazioni sulla registrazione delle `pg_tle` funzionalità per le tue TLE estensioni, consulta. pgtle.register_feature

Prototipo di funzione

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

Ruolo

`pgtle_admin`

Argomenti

- `proc`: il nome della funzione memorizzata che è stata registrata per includere una funzionalità `pg_tle`.
- `feature`: il nome della funzionalità `pg_tle` registrata con l'estensione Trusted Language Extensions.

Output

Restituisce `true` o `false`, come indicato di seguito.

- `true`: la funzione ha completato l'annullamento della registrazione della funzionalità dall'estensione.
- `false`— La funzione non è stata in grado di annullare la registrazione della funzionalità dall'TLEestensione.

Esempio di utilizzo

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

Riferimento sugli hook per Trusted Language Extensions per PostgreSQL

Trusted Language Extensions per PostgreSQL supporta gli hook PostgreSQL. Un hook è un meccanismo di callback interno che gli sviluppatori possono usare per estendere le funzionalità di base di PostgreSQL. Utilizzando gli hook, gli sviluppatori possono implementare le proprie funzioni o procedure da utilizzare durante varie operazioni del database, modificando così il comportamento di PostgreSQL in qualche modo. Ad esempio, puoi usare un hook `passcheck` per personalizzare il modo in cui PostgreSQL gestisce le password fornite durante la creazione o la modifica delle password per gli utenti (ruoli).

Visualizza la seguente documentazione per informazioni sull'hook passcheck disponibile per le tue estensioni TLE. Per ulteriori informazioni sugli hook disponibili, incluso l'hook di autenticazione del client, consulta gli hook Trusted Language Extensions.

Hook di verifica della password (passcheck)

L'hook `passcheck` viene utilizzato per personalizzare il comportamento di PostgreSQL durante il processo di verifica della password per i seguenti comandi SQL e il metacomando `psql`.

- `CREATE ROLE username ...PASSWORD`: per ulteriori informazioni, consulta CREATE ROLE nella documentazione di PostgreSQL.
- `ALTER ROLE username...PASSWORD`: per ulteriori informazioni, consulta ALTER ROLE nella documentazione di PostgreSQL.
- `\password username`: questo metacomando `psql` interattivo modifica in modo sicuro la password per l'utente specificato eseguendo l'hashing della password prima di utilizzare in modo trasparente la sintassi `ALTER ROLE ... PASSWORD`. Il metacomando è un wrapper sicuro per il comando `ALTER ROLE ... PASSWORD`, quindi l'hook si applica al comportamento del metacomando `psql`.

Per vedere un esempio, consulta Codice di hook di controllo della password.

Indice

- Prototipo di funzione
- Argomenti
- Configurazione
- Note per l'utilizzo

Prototipo di funzione

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

Argomenti

La funzione dell'hook `passcheck` accetta i seguenti argomenti:

- `username`: il nome (come testo) del ruolo (nome utente) che imposta una password.

- `password`: la password in chiaro o con hash. La password immessa deve corrispondere al tipo specificato in `password_type`.
- `password_type`: specifica il formato `pgtle.password_type` della password, che può essere costituito da una delle seguenti opzioni.
  - `PASSWORD_TYPE_PLAINTEXT`: una password in testo semplice.
  - `PASSWORD_TYPE_MD5`— Una password sottoposta a hash utilizzando l'algoritmo MD5 (message digest 5).
  - `PASSWORD_TYPE_SCRAM_SHA_256`: una password che è stata sottoposta a hash utilizzando l'algoritmo SCRAM-SHA-256.
- `valid_until`: specifica l'ora in cui la password diventa non valida. Questo argomento è facoltativo. Se utilizzi questo argomento, specifica l'ora come valore `timestamptz`.
- `valid_null`: se questo booleano è impostato su `true`, l'opzione `valid_until` è impostata su `NULL`.

Configurazione

La funzione `pgtle.enable_password_check` controlla se l'hook passcheck è attivo. L'hook passcheck ha tre possibili impostazioni.

- `off`: disattiva l'hook `passcheck` di verifica della password. Si tratta del valore di default.
- on: attiva l'hook `passcode` di verifica della password in modo che le password vengano confrontate con la tabella.
- `require`: richiede la definizione di un hook di verifica della password.

Note per l'utilizzo

Per attivare o disattivare l'hook `passcheck`, è necessario modificare il gruppo di parametri database personalizzato per l'istanza di scrittura del cluster database Aurora PostgreSQL.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

In Windows:

```
aws rds modify-db-parameter-group ^
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Riferimento Amazon Aurora PostgreSQL

Nei seguenti argomenti, puoi trovare informazioni su regole di confronto, funzioni, parametri ed eventi di attesa in Amazon Aurora PostgreSQL.

Argomenti

- Fascicolazioni Aurora PostgreSQL per EBCDIC e altre migrazioni del mainframe
-
- Riferimenti relativi alle funzioni Aurora PostgreSQL
- Amazon Aurora PostgreSQL parametri
- Eventi di attesa Amazon Aurora PostgreSQL

Fascicolazioni Aurora PostgreSQL per EBCDIC e altre migrazioni del mainframe

La migrazione delle applicazioni mainframe su nuove piattaforme, ad esempio, preserva idealmente il comportamento delle applicazioni. AWS Per mantenere il comportamento dell'applicazione su una nuova piattaforma esattamente come era sul mainframe, è necessario che i dati migrati vengano fascicolati utilizzando le stesse regole di fascicolazione e ordinamento. Ad esempio, molte soluzioni di migrazione Db2 spostano valori nulli in u0180 (posizione Unicode 0180), pertanto queste fascicolazioni ordinano prima u0180. Questo è un esempio di come le fascicolazioni possono variare rispetto all'origine del mainframe e perché è necessario scegliere una fascicolazione che si meglio mappata alla fascicolazione EBCDIC originale.

Aurora PostgreSQL 14.3 e versioni successive forniscono molte regole di confronto ICU ed EBCDIC per supportare tale migrazione all'utilizzo del servizio di modernizzazione del mainframe. AWS AWS Per ulteriori informazioni su questo servizio, consulta Cos'è la modernizzazione del mainframe? AWS

Nella tabella seguente, sono disponibili le fascicolazioni fornite da Aurora PostgreSQL. Queste regole di confronto seguono le regole EBCDIC e garantiscono che le applicazioni mainframe funzionino allo stesso modo in cui funzionavano nell'ambiente AWS mainframe. Il nome della fascicolazione include la pagina di codice pertinente, (cpnnnn), per poter scegliere la fascicolazione appropriata per l'origine del mainframe. Ad esempio, utilizza `en-US-cp037-x-icu` per ottenere il comportamento di fascicolazione per i dati EBCDIC provenienti da un'applicazione mainframe che utilizza la pagina codice 037.

Fascicolazioni EBCDIC	AWS Collazioni Blue Age	AWS Collazioni Micro Focus
da-DK-cp1142-x-icu	DA-DK-CP1142 b-x-icu	da-DK-CP1142 m-x-icu
da-DK-cp277-x-icu	DA-DK-CP277 b-x-icu	–
de-DE-cp1141-x-icu	de-DE-CP1141 b-x-icu	de-DE-CP1141 m-x-icu
de-DE-cp273-x-icu	De-DE-CP273 b-x-icu	–
en-GB-cp1146-x-icu	it-GB-CP1146 b-x-icu	it-GB-CP1146 m-x-icu
en-GB-cp285-x-icu	it-GB-CP285 b-x-icu	–
en-US-cp037-x-icu	it-IT-CP037 b-x-icu	–
en-US-cp1140-x-icu	it-IT-CP1140 b-x-icu	it-IT-CP1140 m-x-icu
es-ES-cp1145-x-icu	it-IT-CP1145 b-x-icu	fr-ES-CP1145 m-x-icu
es-ES-cp284-x-icu	fr-ES-CP284 b-x-icu	–
fi-FI-cp1143-x-icu	fi-FI-CP1143 b-x-icu	Fi-FI-CP1143 m-x-icu
fi-FI-cp278-x-icu	Fi-FI-CP278 b-x-icu	–
FR-FR-CP1147-x-ICU	fr-IT-CP1147 b-x-icu	fr-FR-CP1147 m-x-icu
fr-FR-cp297-x-icu	fr-IT-CP297 b-x-icu	–
it-IT-cp1144-x-icu	it-IT-CP1144 b-x-icu	IT-IT-CP1144 m-x-icu
it-IT-cp280-x-icu	IT-IT-CP280 b-x-icu	–
nl-BE-cp1148-x-icu	NL-BE-CP1148 b-x-icu	nl-BE-CP1148 m-x-icu
nl-BE-cp500-x-icu	nl-BE-CP500 b-x-icu	–

Per ulteriori informazioni su AWS Blu Age, consulta il Tutorial: Managed Runtime for AWS Blu Age nella Guida per l'utente alla modernizzazione del mainframe AWS .

Per ulteriori informazioni sull'utilizzo di AWS Micro Focus, consultate Tutorial: Managed Runtime for Micro Focus nella AWS Mainframe Modernization User Guide.

Per ulteriori informazioni sulla gestione delle regole di confronto in PostgreSQL, consulta la pagina relativa al supporto delle regole di confronto nella documentazione di PostgreSQL.

Le regole di confronto sono un insieme di regole che determinano il modo in cui le stringhe di caratteri archiviate nel database vengono ordinate e confrontate. Le regole di confronto svolgono un ruolo fondamentale nel sistema del computer e sono incluse come parte del sistema operativo. Le regole di confronto cambiano nel tempo quando vengono aggiunti nuovi caratteri alle lingue o quando vengono modificate le regole di ordinamento.

Le librerie di regole di confronto definiscono regole e algoritmi specifici per una regola di confronto. Le librerie di collazione più popolari utilizzate in Postgre SQL sono GNU C (glibc) e i componenti di internazionalizzazione per Unicode (). ICU Per impostazione predefinita, Aurora Postgre SQL utilizza la collazione glibc che include ordinamenti di caratteri unicode per sequenze di caratteri multibyte.

Quando crei una nuova sistema operativo le regole di confronto disponibili. I SQL parametri Postgre del `CREATE DATABASE` comando `LC_COLLATE` `LC_CTYPE` vengono utilizzati per specificare un confronto, che rappresenta il confronto predefinito in quel database. In alternativa, puoi anche usare il parametro `LOCALE` in `CREATE DATABASE` per impostare questi parametri e determinare la regole di confronto predefinita per le stringhe di caratteri nel database e le regole per classificare i caratteri come lettere, numeri o simboli. Puoi anche scegliere una regola di confronto da utilizzare per una colonna, un indice o una query.

Aurora Postgre SQL dipende dalla libreria glibc del sistema operativo per il supporto delle collazioni. L'SQListanza Aurora Postgre SQL RDS viene aggiornata periodicamente con le ultime versioni del sistema operativo. Questi aggiornamenti a volte includono una nuova versione della libreria glibc. Raramente, le versioni più recenti della libreria glibc modificano l'ordinamento o la regola di confronto di alcuni caratteri e pertanto i dati possono essere ordinati in modo diverso o produrre voci di indice non valide. Se si riscontrano problemi di ordinamento per la regola di confronto durante un aggiornamento, potrebbe essere necessario ricostruire gli indici.

Per ridurre i possibili impatti degli aggiornamenti di glibc, Aurora Postgre SQL RDS per Postgre ora È compatibile con glibc 2.26-59.amzn2 e fornisce stabilità dell'ordinamento per evitare risultati errati delle query.

Riferimenti relativi alle funzioni Aurora PostgreSQL

Di seguito, è possibile trovare un elenco delle funzioni Aurora PostgreSQL disponibili per i cluster Aurora DB che eseguono il motore di database dell'edizione compatibile con Aurora PostgreSQL. Queste funzioni Aurora PostgreSQL sono in aggiunta alle funzioni standard PostgreSQL. Per ulteriori informazioni sulle funzioni PostgreSQL standard, consulta PostgreSQL: funzioni e operatori.

Panoramica

È possibile utilizzare le seguenti funzioni per le istanze database Amazon RDS che eseguono Aurora PostgreSQL:

- aurora_db_instance_identifier
- aurora_ccm_status
- aurora_global_db_instance_status
- aurora_global_db_status
- aurora_list_builtins
- aurora_replica_status
- aurora_stat_activity
- aurora_stat_backend_waits
- aurora_stat_bgwriter
- aurora_stat_database
- aurora_stat_dml_activity
- aurora_stat_get_db_commit_latency
- aurora_stat_logical_wal_cache
- aurora_stat_memctx_usage
- aurora_stat_optimized_reads_cache
- aurora_stat_plans
- aurora_stat_reset_wal_cache
- aurora_stat_statements
- aurora_stat_system_waits
- aurora_stat_wait_event

- aurora_stat_wait_type
- aurora_version
- aurora_volume_logical_start_lsn
- aurora_wait_report

aurora_db_instance_identifier

Comunica il nome dell'istanza database cui si è connessi.

Sintassi

```
aurora_db_instance_identifier()
```

Argomenti

Nessuno

Tipo restituito

Stringa VARCHA

Note per l'utilizzo

Questa funzione visualizza il nome dell'istanza database del cluster Aurora edizione compatibile con PostgreSQL per la connessione al client di database o all'applicazione.

Questa funzione è disponibile a partire dal rilascio di PostgreSQL versioni 13.7, 12.11, 11.16, 10.21 e per tutte le altre versioni successive.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata della funzione `aurora_db_instance_identifier`.

```
=> SELECT aurora_db_instance_identifier();
aurora_db_instance_identifier
------------------------------
```

```
 test-my-instance-name
```

È possibile unire i risultati di questa funzione con la funzione `aurora_replica_status` per ottenere i dettagli relativi all'istanza database per la connessione. Il parametro aurora_replica_status da solo non mostra l'istanza database in uso. Nell'esempio seguente viene illustrato come fare.

```
=> SELECT *
    FROM aurora_replica_status() rt,
         aurora_db_instance_identifier() di
    WHERE rt.server_id = di;
-[ RECORD 1 ]----------------------+-----------------------
server_id                         | test-my-instance-name
session_id                        | MASTER_SESSION_ID
durable_lsn                       | 88492069
highest_lsn_rcvd                  |
current_read_lsn                  |
cur_replay_latency_in_usec        |
active_txns                       |
is_current                        | t
last_transport_error              | 0
last_error_timestamp              |
last_update_timestamp             | 2022-06-03 11:18:25+00
feedback_xmin                     |
feedback_epoch                    |
replica_lag_in_msec               |
log_stream_speed_in_kib_per_second | 0
log_buffer_sequence_number        | 0
oldest_read_view_trx_id           |
oldest_read_view_lsn              |
pending_read_ios                  | 819
```

aurora_ccm_status

Visualizza lo stato del gestore della cache del cluster.

Sintassi

```
aurora_ccm_status()
```

Argomenti

Nessuna.

Tipo restituito

Record SETOF avente le seguenti colonne:

- `buffers_sent_last_minute`: il numero di buffer inviati all'istanza database di lettura designata nell'ultimo minuto.
- `buffers_found_last_minute`: il numero di buffer con accesso frequente identificati durante l'ultimo minuto.
- `buffers_sent_last_scan`: il numero di buffer inviati all'istanza di lettura designata durante l'ultima scansione completa della cache.
- `buffers_found_last_scan`: il numero di buffer ad accesso frequente inviati durante l'ultima scansione completa della cache. I buffer già memorizzati sull'istanza database di lettura designata non vengono inviati.
- `buffers_sent_current_scan`: il numero di buffer inviati durante la scansione corrente.
- `buffers_found_current_scan`: il numero di buffer ad accesso frequente identificati nella scansione corrente.
- `current_scan_progress`: il numero di buffer analizzati finora durante la scansione corrente.

Note per l'utilizzo

È possibile utilizzare questa funzione per controllare e monitorare la funzionalità di gestione della cache del cluster (CCM). Ciò può avvenire solo se la funzionalità di gestione della cache del cluster è attiva sul cluster di database Aurora PostgreSQL. Per utilizzare questa funzione, connettersi all'istanza DB di scrittura sul cluster di database Aurora PostgreSQL.

Per attivare la funzionalità di gestione della cache del cluster per un cluster di database Aurora PostgreSQL, impostare `apg_ccm_enabled` su 1 nel gruppo di parametri del cluster di database personalizzato. Per scoprire come, consulta Configurazione della gestione della cache del cluster.

La gestione della cache del cluster è attiva su un cluster di database Aurora PostgreSQL quando il cluster dispone di un'istanza di lettura Aurora configurata come segue:

- L'istanza di lettura Aurora utilizza lo stesso tipo di classe di istanza database e la stessa dimensione dell'istanza di scrittura del cluster.
- L'istanza di lettura Aurora è configurata come Tier-0 per il cluster. Se il cluster include più di un'istanza di lettura, questa istanza sarà l'unica istanza di lettura di tipo Tier-0.

L'impostazione di più di un'istanza di lettura su Tier-0 disabilita la funzionalità di gestione della cache del cluster. Quando la funzionalità di gestione della cache del cluster è disabilitata, una chiamata a questa funzione restituirà il seguente messaggio di errore:

```
ERROR: Cluster Cache Manager is disabled
```

È inoltre possibile impostare l'estensione PostgreSQL pg_buffercache affinché analizzi la cache del buffer. Per ulteriori informazioni, consulta pg_buffercache nella documentazione di PostgreSQL.

Per ulteriori informazioni, consulta Introduzione alla gestione della cache del cluster Aurora PostgreSQL.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata della funzione `aurora_ccm_status`. Questo primo esempio mostra le statistiche restituite dalla funzionalità di gestione della cache del cluster.

```
=> SELECT * FROM aurora_ccm_status();
 buffers_sent_last_minute | buffers_found_last_minute | buffers_sent_last_scan |
 buffers_found_last_scan | buffers_sent_current_scan | buffers_found_current_scan |
 current_scan_progress
--------------------------+---------------------------+------------------------
+-------------------------+---------------------------+---------------------------
+------------------------
                   2242000 |                   2242003 |               17920442 |
         17923410 |                  14098000 |                  14100964 |
   15877443
```

Per dettagli più completi, è possibile utilizzare la visualizzazione espansa, come illustrato di seguito:

```
\x
Expanded display is on.
SELECT *  FROM aurora_ccm_status();
[ RECORD 1 ]----------------------+---------
buffers_sent_last_minute          | 2242000
buffers_found_last_minute         | 2242003
buffers_sent_last_scan            | 17920442
buffers_found_last_scan           | 17923410
buffers_sent_current_scan         | 14098000
buffers_found_current_scan        | 14100964
```

```
current_scan_progress              | 15877443
```

Questo esempio mostra come controllare la frequenza e la percentuale degli avvii a caldo.

```
=> SELECT buffers_sent_last_minute * 8/60 AS warm_rate_kbps,
100 * (1.0-buffers_sent_last_scan/buffers_found_last_scan) AS warm_percent
FROM aurora_ccm_status ();
 warm_rate_kbps | warm_percent
----------------+--------------
 16523 |        100.0
```

aurora_global_db_instance_status

Visualizza lo stato di tutte le istanze Aurora, incluse le repliche in un cluster di database globale Aurora.

Sintassi

```
aurora_global_db_instance_status()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `server_id`: l'identificatore dell'istanza database.
- `session_id`: un identificatore univoco per la sessione. Il valore di `MASTER_SESSION_ID` identifica l'istanza database di lettura (primaria).
- `aws_region`— Il sistema Regione AWS in cui viene eseguita questa istanza database globale. Per l'elenco delle regioni , consulta Disponibilità nelle regioni.
- `durable_lsn`: il numero di sequenza di log (LSN) reso durevole nell'archiviazione. Un numero di sequenza di registro (LSN) è un numero sequenziale univoco che identifica un record nel registro delle transazioni del database. LSNs sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.

- `highest_lsn_rcvd`: l'LSN più alto che l'istanza database ha ricevuto dall'istanza database di scrittura.
- `feedback_epoch`: l'epoca utilizzata dall'istanza database quando genera informazioni standby a caldo. Uno standby a caldo è un'istanza database che supporta connessioni e query mentre il database primario è in modalità di ripristino o standby. Le informazioni relative allo standby a caldo includono l'epoca (point-in-time) e altri dettagli sull'istanza database utilizzata come standby a caldo. Per ulteriori informazioni, consulta Hot Standby nella documentazione di PostgreSQL.
- `feedback_xmin`: l'ID della transazione attiva più basso utilizzato dall'istanza database.
- `oldest_read_view_lsn`: l'LSN più vecchio utilizzato dall'istanza database per le operazioni di lettura dallo storage.
- `visibility_lag_in_msec`: il ritardo dell'istanza database corrente rispetto all'istanza database di scrittura in millisecondi.

Note per l'utilizzo

Questa funzione mostra le statistiche di replica per un cluster di database Aurora. Per ogni istanza database Aurora PostgreSQL nel cluster, la funzione mostra una riga di dati che include eventuali repliche tra regioni in una configurazione di database globale.

È possibile eseguire questa funzione da qualsiasi istanza in un cluster di database Aurora PostgreSQL o Aurora PostgreSQL. La funzione restituisce dettagli sul ritardo per tutte le istanze di replica.

Per ulteriori informazioni sul monitoraggio del ritardo utilizzando questa funzione (`aurora_global_db_instance_status`) o `aurora_global_db_status`, consulta Monitoraggio dei database globali basati su Aurora SQL Postgre.

Per informazioni sui database globali Aurora, consulta Panoramica del database globale di Amazon Aurora.

Per iniziare a usare i database globali di Aurora, consulta Inizia a usare Amazon Aurora Global Database o consulta Amazon Aurora. FAQs

Esempi

Questo esempio mostra le statistiche delle istanze tra regioni.

```
=> SELECT *
```

```
  FROM aurora_global_db_instance_status();
               server_id                |              session_id              |
  aws_region  | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------+--------------------------------------
+-------------+-------------+------------------+----------------+---------------
+----------------------+------------------------
 db-119-001-instance-01                  | MASTER_SESSION_ID                    | eu-
west-1    |  2534560273 |           [NULL] |         [NULL] |        [NULL] |
     [NULL] |                 [NULL]
 db-119-001-instance-02                  | 4ecff34d-d57c-409c-ba28-278b31d6fc40 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                      6
 db-119-001-instance-03                  | 3e8a20fc-be86-43d5-95e5-bdf19d27ad6b | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                      6
 db-119-001-instance-04                  | fc1b0023-e8b4-4361-bede-2a7e926cead6 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                     23
 db-119-001-instance-05                  | 30319b74-3f08-4e13-9728-e02aa1aa8649 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                     23
 db-119-001-global-instance-1            | a331ffbb-d982-49ba-8973-527c96329c60 | eu-
central-1 |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                    996
 db-119-001-global-instance-1            | e0955367-7082-43c4-b4db-70674064a9da | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                     14
 db-119-001-global-instance-1-eu-west-2a  | 1248dc12-d3a4-46f5-a9e2-85850491a897 | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                      0
```

Questo esempio mostra come controllare il ritardo delle repliche globali in millisecondi.

```
=> SELECT CASE
          WHEN 'MASTER_SESSION_ID' = session_id THEN 'Primary'
         ELSE 'Secondary'
      END AS global_role,
      aws_region,
      server_id,
      visibility_lag_in_msec
  FROM aurora_global_db_instance_status()
  ORDER BY 1, 2, 3;
```

```
   global_role |  aws_region  |               server_id                |
 visibility_lag_in_msec
-------------+--------------+----------------------------------------
+-----------------------
 Primary     | eu-west-1    | db-119-001-instance-01                 |
 [NULL]
 Secondary   | eu-central-1 | db-119-001-global-instance-1           |
     13
 Secondary   | eu-west-1    | db-119-001-instance-02                 |
     10
 Secondary   | eu-west-1    | db-119-001-instance-03                 |
      9
 Secondary   | eu-west-1    | db-119-001-instance-04                 |
      2
 Secondary   | eu-west-1    | db-119-001-instance-05                 |
     18
 Secondary   | eu-west-2    | db-119-001-global-instance-1           |
     14
 Secondary   | eu-west-2    | db-119-001-global-instance-1-eu-west-2a |
     13
```

Questo esempio mostra come controllare il ritardo minimo, massimo e medio rispetto alla configurazione globale Regione AWS del database.

```
=> SELECT 'Secondary' global_role,
       aws_region,
       min(visibility_lag_in_msec) min_lag_in_msec,
       max(visibility_lag_in_msec) max_lag_in_msec,
       round(avg(visibility_lag_in_msec),0) avg_lag_in_msec
  FROM aurora_global_db_instance_status()
 WHERE aws_region NOT IN (SELECT   aws_region
                            FROM aurora_global_db_instance_status()
                           WHERE session_id='MASTER_SESSION_ID')
                         GROUP BY aws_region
UNION ALL
SELECT  'Primary' global_role,
        aws_region,
        NULL,
        NULL,
        NULL
    FROM aurora_global_db_instance_status()
   WHERE session_id='MASTER_SESSION_ID'
ORDER BY 1, 5;
```

```
 global_role |  aws_region  | min_lag_in_msec | max_lag_in_msec | avg_lag_in_msec
-------------+--------------+-----------------+-----------------+-----------------
 Primary     | eu-west-1    |          [NULL] |          [NULL] |          [NULL]
 Secondary   | eu-central-1 |             133 |             133 |             133
 Secondary   | eu-west-2    |               0 |             495 |             248
```

aurora_global_db_status

Visualizza informazioni su vari aspetti del ritardo del database globale Aurora, in particolare il ritardo dell'archiviazione Aurora sottostante (il cosiddetto ritardo di durabilità) e il ritardo rispetto all'obiettivo del punto di ripristino (RPO).

Sintassi

```
aurora_global_db_status()
```

Argomenti

Nessuna.

Tipo restituito

Record SETOF avente le seguenti colonne:

- `aws_region`— In Regione AWS cui si trova questo cluster DB. Per un elenco completo dei modelli Regioni AWS suddivisi per motore, vedereRegioni e zone di disponibilità.
- `highest_lsn_written`: il numero di sequenza di log (LSN) più alto attualmente esistente in questo cluster di database. Un numero di sequenza di registro (LSN) è un numero sequenziale univoco che identifica un record nel registro delle transazioni del database. LSNs sono ordinati in modo tale che un LSN più grande rappresenti una transazione successiva.
- `durability_lag_in_msec`: la differenza tra i valori di timestamp tra `highest_lsn_written` su un cluster di database secondario e `highest_lsn_written` sul cluster di database primario. Un valore -1 identifica il cluster di database globale principale di un database globale Aurora.
- `rpo_lag_in_msec`: il ritardo dell'obiettivo del punto di ripristino (RPO). Il ritardo dell'obiettivo del punto di ripristino (RPO) è il tempo necessario per memorizzare il COMMIT delle transazioni utente più recenti dopo la sua memorizzazione nel cluster di database primario del database globale Aurora. Un valore -1 indica il cluster di database primario e quindi il ritardo non è rilevante.

In sintesi, questo parametro calcola l'obiettivo del punto di ripristino per ciascun cluster di database Aurora PostgreSQL nel database globale Aurora, ovvero quanti dati potrebbero andare perduti in caso di interruzione. Come per il ritardo, l'obiettivo del punto di ripristino (RPO) viene misurato nel tempo.

- `last_lag_calculation_time`: il timestamp che specifica quando sono stati calcolati i valori per `durability_lag_in_msec` e `rpo_lag_in_msec`. Il valore temporale `1970-01-01 00:00:00+00` indica che questo è il cluster di database primario.
- `feedback_epoch`: l'epoca utilizzata dal cluster di database secondario quando genera informazioni sullo standby a caldo. Uno standby a caldo è un'istanza database che supporta connessioni e query mentre il database primario è in modalità di ripristino o standby. Le informazioni relative allo standby a caldo includono l'epoca (point-in-time) e altri dettagli sull'istanza database utilizzata come standby a caldo. Per ulteriori informazioni, consulta Hot Standby nella documentazione di PostgreSQL.
- `feedback_xmin`: l'ID minimo della transazione attiva (meno recente) utilizzato da un cluster di database secondario.

Note per l'utilizzo

Tutte le versioni di Aurora PostgreSQL attualmente disponibili supportano questa funzione. Questa funzione mostra le statistiche di replica per un database globale Aurora. Mostra una riga per ciascun cluster di database in un database globale Aurora PostgreSQL. È possibile eseguire questa funzione da qualsiasi istanza del database globale Aurora PostgreSQL.

Per valutare il ritardo di replica del database globale Aurora, ovvero il ritardo visibile dei dati, consulta aurora_global_db_instance_status.

Per ulteriori informazioni sull'uso di `aurora_global_db_status` e `aurora_global_db_instance_status` per monitorare il ritardo del database globale Aurora, consulta Monitoraggio dei database globali basati su Aurora SQL Postgre. Per informazioni sui database globali Aurora, consulta Panoramica del database globale di Amazon Aurora.

Esempi

Questo esempio mostra come visualizzare le statistiche di archiviazione tra regioni.

```
=> SELECT CASE
```

```
          WHEN '-1' = durability_lag_in_msec THEN 'Primary'
          ELSE 'Secondary'
       END AS global_role,
       *
  FROM aurora_global_db_status();
 global_role | aws_region | highest_lsn_written | durability_lag_in_msec |
 rpo_lag_in_msec | last_lag_calculation_time  | feedback_epoch | feedback_xmin
-------------+------------+---------------------+------------------------
+-----------------+----------------------------+----------------+---------------
 Primary     | eu-west-1  |           131031557 |                     -1 |
 -1 | 1970-01-01 00:00:00+00     |             0 |             0
 Secondary   | eu-west-2  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
 Secondary   | eu-west-3  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
```

aurora_list_builtins

Elenca tutte le funzioni integrate di Aurora PostgreSQL disponibili, insieme a brevi descrizioni e dettagli delle funzioni.

Sintassi

```
aurora_list_builtins()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_list_builtins`.

```
=> SELECT *
FROM aurora_list_builtins();
```

```
              Name                | Result data type |                  Argument data
 types                     | Type | Volatility |  Parallel  | Security |
            Description
-----------------------------------+------------------
+-------------------------------------------------------------+------+------------
+------------+----------
+-----------------------------------------------------------------------
 aurora_version                    | text             |
                           | func | stable     | safe       | invoker  | Amazon Aurora
 PostgreSQL-Compatible Edition version string
 aurora_stat_wait_type             | SETOF record     | OUT type_id smallint, OUT
 type_name text                 | func | volatile   | restricted | invoker  | Lists all
 supported wait types
 aurora_stat_wait_event            | SETOF record     | OUT type_id smallint, OUT
 event_id integer, OUT event_na.| func | volatile   | restricted | invoker  | Lists all
 supported wait events
                                   |                  |.me text
                          |      |           |            |          |
 aurora_list_builtins              | SETOF record     | OUT "Name" text, OUT "Result
 data type" text, OUT "Argum.| func | stable     | safe       | invoker  | Lists all
 Aurora built-in functions
                                   |                  |.ent data types" text, OUT
 "Type" text, OUT "Volatility" .|      |           |            |          |
                                   |                  |.text, OUT "Parallel" text, OUT
 "Security" text, OUT "Des.|      |            |            |          |
                                   |                  |.cription" text
                          |      |           |            |          |
 .
 .
 .
 aurora_stat_file                  | SETOF record     | OUT filename text, OUT
 allocated_bytes bigint, OUT used_.| func | stable     | safe       | invoker  | Lists
 all files present in Aurora storage
                                   |                  |.bytes bigint
                          |      |           |            |          |
 aurora_stat_get_db_commit_latency | bigint           | oid
                           | func | stable     | restricted | invoker  | Per DB commit
 latency in microsecs
```

aurora_replica_status

Visualizza lo stato di tutti i nodi del lettore Aurora PostgreSQL.

Sintassi

```
aurora_replica_status()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `server_id`: ID dell'istanza database (identificatore).
- `session_id`: un identificatore univoco per la sessione corrente, restituito per l'istanza primaria e le istanze di lettura nel modo seguente:
  - Per l'istanza primaria, il `session_id` è sempre `` `MASTER_SESSION_ID' ``.
  - Per le istanze di lettura, il `session_id` è sempre l'`UUID` (universally unique identifier, identificatore univoco universale) dell'istanza di lettura.
- `durable_lsn`: numero di sequenza di log (log sequence number, LSN) archiviato nello storage.
  - Per il volume primario, l'LSN (VDL) durevole del primario attualmente in uso.
  - Per tutti i volumi secondari, il VDL del primario a cui è stato applicato correttamente il secondario.

> **Note**
>
> Un numero di sequenza di registro (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. LSNs sono ordinati in modo tale che un LSN più grande rappresenti una transazione avvenuta più avanti nella sequenza.

- `highest_lsn_rcvd`: l'LSN più alto (più recente) che l'istanza database ha ricevuto dall'istanza database di scrittura.
- `current_read_lsn`: l'LSN della snapshot più recente applicato a tutte le operazioni di lettura.
- `cur_replay_latency_in_usec`: tempo previsto in millisecondi per riprodurre il log sul secondario.
- `active_txns`: numero di transazioni attualmente attive.
- `is_current`: non utilizzato.

- `last_transport_error`: codice di errore dell'ultima replica.
- `last_error_timestamp`: timestamp dell'ultimo errore di replica.
- `last_update_timestamp`: timestamp dell'ultimo aggiornamento allo stato della replica. Da Aurora PostgreSQL versione 13.9, il valore `last_update_timestamp` per l'istanza database a cui si è connessi è impostato su `NULL`.
- `feedback_xmin`: feedback_xmin della replica in standby a caldo. L'ID della transazione attiva più basso (meno recente) utilizzato dall'istanza database.
- `feedback_epoch`: l'epoca utilizzata dall'istanza database quando genera informazioni standby a caldo.
- `replica_lag_in_msec`— Tempo di ritardo dell'istanza reader rispetto all'istanza di Writer, in millisecondi.
- `log_stream_speed_in_kib_per_second`: la velocità effettiva del flusso di log in kilobyte al secondo.
- `log_buffer_sequence_number`: il numero di sequenza del buffer di registro.
- `oldest_read_view_trx_id`: non utilizzato.
- `oldest_read_view_lsn`: l'LSN più vecchio utilizzato dall'istanza database per le operazioni di lettura dallo storage.
- `pending_read_ios`: le letture di pagina in sospeso sulla replica.
- `read_ios`: il numero totale di letture di pagina sulla replica.
- `iops`: non utilizzato.
- `cpu`— Utilizzo della CPU dell'Aurora Storage Daemon per ogni nodo del cluster. Per informazioni sull'utilizzo della CPU da parte dell'istanza, consulta Parametri a livello di istanza per Amazon Aurora.

Note per l'utilizzo

Tutte le versioni di Aurora PostgreSQL attualmente disponibili supportano questa funzione. La funzione `aurora_replica_status` restituisce i valori del gestore di stato della replica di un cluster database Aurora PostgreSQL. Utilizza questa funzione per ottenere informazioni sullo stato della replica sul cluster database Aurora PostgreSQL, inclusi i parametri per tutte le istanze database nel cluster database Aurora. Ad esempio, puoi eseguire le operazioni seguenti:

- Ottenere informazioni sul tipo di istanza (di scrittura, di lettura) nel cluster database Aurora PostgreSQL. Per ottenere queste informazioni, controlla i valori contenuti nelle colonne seguenti:

  - `server_id`: contiene il nome dell'istanza specificato durante la sua creazione. In alcuni casi, ad esempio per l'istanza primaria (di scrittura), il nome viene generato automaticamente aggiungendo -instance-1 al nome creato per il cluster database Aurora PostgreSQL.
  - `session_id`: il campo `session_id` indica se l'istanza è di lettura o di scrittura. Per un'istanza di scrittura, `session_id` è sempre impostato su `''MASTER_SESSION_ID''`. Per un'istanza di lettura, `session_id` è impostato sull'`UUID` del lettore specifico.
- Diagnostica dei problemi comuni correlati alle repliche, come il ritardo di replica. Il ritardo di replica indica il ritardo in millisecondi della cache di pagina di un'istanza di lettura rispetto a quella dell'istanza di scrittura. Questo ritardo si verifica perché i cluster Aurora utilizzano la replica asincrona, come descritto in Replica con Amazon Aurora. Nella colonna `replica_lag_in_msec` sono mostrati i risultati restituiti da questa funzione. Il ritardo può verificarsi anche quando una query viene annullata a causa di conflitti con il ripristino su un server in standby. È possibile verificare se il ritardo di replica è causato o meno da questo conflitto controllando `pg_stat_database_conflicts()`. Per ulteriori informazioni, consulta Raccolta di statistiche nella Documentazione di PostgreSQL. Per ulteriori informazioni sull'alta disponibilità e la replica, consulta Amazon FAQs Aurora.

  Amazon CloudWatch memorizza `replica_lag_in_msec` i risultati nel tempo, come `AuroraReplicaLag` metrica. Per informazioni sull'utilizzo delle CloudWatch metriche per Aurora, consulta Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch

Per ulteriori informazioni sulla risoluzione dei problemi relativi alla lettura e al riavvio di Aurora, consulta Perché la mia replica Amazon Aurora è in ritardo e si è riavviata? nel Centro Supporto AWS.

Esempi

L'esempio seguente mostra come ottenere lo stato di replica di tutte le istanze in un cluster database Aurora PostgreSQL:

```
=> SELECT *
FROM aurora_replica_status();
```

L'esempio seguente mostra l'istanza di lettura nel cluster database `docs-lab-apg-main` Aurora PostgreSQL:

```
=> SELECT server_id,
    CASE
```

```
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id = 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
 db-119-001-instance-01 | writer
```

Nell'esempio seguente sono elencate tutte le istanze di lettura in un cluster:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id <> 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
db-119-001-instance-02  | reader
db-119-001-instance-03  | reader
db-119-001-instance-04  | reader
db-119-001-instance-05  | reader
(4 rows)
```

Nell'esempio seguente sono elencate tutte le istanze, il ritardo di ognuna di esse rispetto all'istanza di scrittura e quanto tempo è trascorso dall'ultimo aggiornamento:

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role,
    replica_lag_in_msec AS replica_lag_ms,
    round(extract (epoch FROM (SELECT age(clock_timestamp(), last_update_timestamp))) *
 1000) AS last_update_age_ms
FROM aurora_replica_status()
ORDER BY replica_lag_in_msec NULLS FIRST;
       server_id        | instance_role | replica_lag_ms | last_update_age_ms
------------------------+---------------+----------------+--------------------
 db-124-001-instance-03 | writer        |         [NULL] |               1756
 db-124-001-instance-01 | reader        |             13 |               1756
```

```
 db-124-001-instance-02 | reader        |             13 |              1756
(3 rows)
```

aurora_stat_activity

Restituisce una riga per processo del server, mostrando le informazioni relative all'attività corrente di quel processo.

Sintassi

```
aurora_stat_activity();
```

Argomenti

Nessuno

Tipo restituito

Restituisce una riga per processo server. Oltre alle `pg_stat_activity` colonne, viene aggiunto il seguente campo:

- planid — identificatore del piano

Note per l'utilizzo

Una visualizzazione supplementare per `pg_stat_activity` restituire le stesse colonne con una `plan_id` colonna aggiuntiva che mostra il piano di esecuzione delle query corrente.

`aurora_compute_plan_id`deve essere abilitato affinché la vista restituisca un plan_id.

Questa funzione è disponibile a partire dalle versioni 14.10, 15.5 di Aurora PostgreSQL e per tutte le altre versioni successive.

Esempi

La query di esempio riportata di seguito aggrega il caricamento massimo per query_id e plan_id.

```
db1=# select count(*), query_id, plan_id
db1-# from aurora_stat_activity() where state = 'active'
db1-# and pid <> pg_backend_pid()
db1-# group by query_id, plan_id
db1-# order by 1 desc;

count |  query_id             |  plan_id
-------+----------------------+-------------
 11    | -5471422286312252535 | -2054628807
 3     | -6907107586630739258 | -815866029
 1     | 5213711845501580017  |  300482084
(3 rows)
```

Se il piano utilizzato per query_id cambia, aurora_stat_activity riporterà un nuovo plan_id.

```
count  |  query_id             |  plan_id
-------+----------------------+-------------
 10    | -5471422286312252535 | 1602979607
 1     | -6907107586630739258 | -1809935983
 1     | -2446282393000597155 | -207532066
(3 rows)
```

aurora_stat_backend_waits

Visualizza le statistiche per l'attività di attesa per un processo di back-end specifico.

Sintassi

```
aurora_stat_backend_waits(pid)
```

Argomenti

`pid`: l'ID per il processo di back-end. È possibile ottenere il processo IDs utilizzando la `pg_stat_activity` visualizzazione.

Tipo restituito

Record SETOF avente le seguenti colonne:

- `type_id`: un numero che indica il tipo di evento di attesa, ad esempio 1 per un blocco leggero (`LWLock`), 3 per un blocco o 6 per una sessione client. Questi valori diventano significativi quando i risultati di questa funzione vengono abbinati alle colonne della funzione `aurora_stat_wait_type`, come descritto nella sezione Esempi.
- `event_id`: un numero identificativo per l'evento di attesa. Abbina questo valore alle colonne di `aurora_stat_wait_event` per ottenere nomi significativi per gli eventi.
- `waits`: conteggio del numero di attese accumulate per l'ID di processo specificato.
- `wait_time`: tempo di attesa in millisecondi.

Note per l'utilizzo

È possibile utilizzare questa funzione per analizzare eventi di attesa di back-end (sessione) specifici che si sono verificati dall'apertura di una connessione. Per ottenere informazioni più significative sui nomi e sui tipi di eventi di attesa, è possibile combinare `aurora_stat_wait_type` e `aurora_stat_wait_event`, utilizzando JOIN come mostrato negli esempi.

Esempi

Questo esempio mostra tutte le attese, tutti i tipi e tutti i nomi di evento per l'ID processo back-end 3027.

```
=> SELECT type_name, event_name, waits, wait_time
        FROM aurora_stat_backend_waits(3027)
NATURAL JOIN aurora_stat_wait_type()
NATURAL JOIN aurora_stat_wait_event();
type_name |       event_name       | waits | wait_time
----------+------------------------+-------+------------
 LWLock    | ProcArrayLock          |     3 |         27
 LWLock    | wal_insert             |   423 |      16336
 LWLock    | buffer_content         | 11840 |    1033634
 LWLock    | lock_manager           | 23821 |    5664506
 Lock      | tuple                  | 10258 |  152280165
 Lock      | transactionid          | 78340 | 1239808783
 Client    | ClientRead             | 34072 |   17616684
 IO        | ControlFileSyncUpdate  |     2 |          0
 IO        | ControlFileWriteUpdate |     4 |         32
```

```
 IO        | RelationMapRead        |     2 |        795
 IO        | WALWrite               | 36666 |      98623
 IO        | XactSync               |  4867 |    7331963
```

Questo esempio mostra i tipi di attesa correnti e cumulativi e gli eventi di attesa per tutte le sessioni attive (`pg_stat_activity state <> 'idle'`) (ma senza la sessione corrente che richiama la funzione (`pid <> pg_backend_pid()`).

```
=> SELECT a.pid,
            a.usename,
            a.app_name,
            a.current_wait_type,
            a.current_wait_event,
            a.current_state,
            wt.type_name AS wait_type,
            we.event_name AS wait_event,
            a.waits,
            a.wait_time
       FROM (SELECT pid,
                    usename,
                    left(application_name,16) AS app_name,
                    coalesce(wait_event_type,'CPU') AS current_wait_type,
                    coalesce(wait_event,'CPU') AS current_wait_event,
                    state AS current_state,
                    (aurora_stat_backend_waits(pid)).*
               FROM pg_stat_activity
              WHERE pid <> pg_backend_pid()
                AND state <> 'idle') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |       wait_event       | waits | wait_time
-------+----------+----------+-------------------+--------------------+---------------
+-----------+------------------------+-------+-----------
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1937 |     29975
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22903 |    760498
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | lock_manager           | 10012 |    223207
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple                  | 20315 |  63081529
 .
```

```
.
.
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
IO        | WALWrite               | 93293 |    237440
30099 | postgres | pgbench  | Lock              | transactionid      | active        |
IO        | XactSync               | 13010 |  19525143
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | ProcArrayLock          |     6 |        53
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | wal_insert             |  1913 |     25450
30100 | postgres | pgbench  | Lock              | transactionid      | active        |
LWLock    | buffer_content         | 22874 |    778005
.
.
.
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | ProcArrayLock          |     3 |        71
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | wal_insert             |  1940 |     27741
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | buffer_content         | 22962 |    776352
30109 | postgres | pgbench  | IO                | XactSync           | active        |
LWLock    | lock_manager           |  9879 |    218826
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | tuple                  | 20401 |  63581306
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Lock      | transactionid          | 50769 | 211645008
30109 | postgres | pgbench  | IO                | XactSync           | active        |
Client    | ClientRead             | 89901 |  44192439
```

Questo esempio mostra il tipo di attesa corrente e i primi tre (3) eventi di attesa cumultativi, nonché gli eventi di attesa per tutte le sessioni attive (pg_stat_activity state <> 'idle') esclusa la sessione corrente (pid <>pg_backend_pid()).

```
=> SELECT top3.*
       FROM (SELECT a.pid,
                    a.usename,
                    a.app_name,
                    a.current_wait_type,
                    a.current_wait_event,
                    a.current_state,
                    wt.type_name AS wait_type,
                    we.event_name AS wait_event,
```

```
                    a.waits,
                    a.wait_time,
                    RANK() OVER (PARTITION BY pid ORDER BY a.wait_time DESC)
                FROM (SELECT pid,
                             usename,
                             left(application_name,16) AS app_name,
                             coalesce(wait_event_type,'CPU') AS current_wait_type,
                             coalesce(wait_event,'CPU') AS current_wait_event,
                             state AS current_state,
                             (aurora_stat_backend_waits(pid)).*
                        FROM pg_stat_activity
                       WHERE pid <> pg_backend_pid()
                         AND state <> 'idle') a
          NATURAL JOIN aurora_stat_wait_type() wt
          NATURAL JOIN aurora_stat_wait_event() we) top3
 WHERE RANK <=3;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |   wait_event    |  waits  | wait_time  | rank
-------+----------+----------+-------------------+--------------------+---------------
+-----------+-----------------+---------+------------+------
 20567 | postgres | psql     | CPU               | CPU                | active        |
 LWLock    | wal_insert      |   25000 |   67512003 |    1
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | WALWrite        | 3071758 |    1016961 |    2
 20567 | postgres | psql     | CPU               | CPU                | active        |
 IO        | BufFileWrite    |   20750 |     184559 |    3
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | transactionid   |  237350 | 1265580011 |    1
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple           |   93641 |  341472318 |    2
 27743 | postgres | pgbench  | Lock              | transactionid      | active        |
 Client    | ClientRead      |  417556 |  204796837 |    3
 .
 .
 .
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | transactionid   |  238068 | 1265816822 |    1
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Lock      | tuple           |   93210 |  338312247 |    2
 27745 | postgres | pgbench  | IO                | XactSync           | active        |
 Client    | ClientRead      |  419157 |  207836533 |    3
 27746 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | transactionid   |  237621 | 1264528811 |    1
```

```
 27746 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple           |   93563 |  339799310 |    2
 27746 | postgres | pgbench  | Lock              | transactionid      | active        |
 Client    | ClientRead      |  417304 |  208372727 |    3
```

aurora_stat_bgwriter

`aurora_stat_bgwriter` è una vista delle statistiche che mostra informazioni sulle scritture nella cache di Letture ottimizzate.

Sintassi

```
aurora_stat_bgwriter()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF con tutte le colonne `pg_stat_bgwriter` e le seguenti colonne aggiuntive. Per ulteriori informazioni sulle colonne `pg_stat_bgwriter`, consulta `pg_stat_bgwriter`.

È possibile ripristinare le statistiche per questa funzione utilizzando `pg_stat_reset_shared("bgwriter")`.

- `orcache_blks_written`: numero totale di blocchi di dati della cache di Letture ottimizzate scritti.
- `orcache_blk_write_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale in millisecondi impiegato per scrivere blocchi di dati di file della cache di Letture ottimizzate. Per ulteriori informazioni, consulta track_io_timing.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive
- 14.9 e versioni successive

Esempi

```
=> select * from aurora_stat_bgwriter();
-[ RECORD 1 ]-----------------+-----------
orcache_blks_written          | 246522
orcache_blk_write_time        | 339276.404
```

aurora_stat_database

Contiene tutte le colonne di pg_stat_database e aggiunge nuove colonne alla fine.

Sintassi

```
aurora_stat_database()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF con tutte le colonne `pg_stat_database` e le seguenti colonne aggiuntive. Per ulteriori informazioni sulle colonne `pg_stat_database`, consulta `pg_stat_database`.

- `storage_blks_read`: numero totale di blocchi condivisi letti dall'archiviazione Aurora in questo database.
- `orcache_blks_hit`: numero totale di accessi alla cache di Letture ottimizzate in questo database.
- `local_blks_read`: numero totale di blocchi locali letti in questo database.
- `storage_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato in millisecondi per leggere i blocchi di file di dati dall'archiviazione Aurora altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `local_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato in millisecondi per leggere i blocchi di file di dati locali, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `orcache_blk_read_time`: se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato in millisecondi per leggere i blocchi di file di dati dalla cache di Letture ottimizzate, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.

> **Note**
>
> Il valore di `blks_read` è la somma di `storage_blks_read`, `orcache_blks_hit` e `local_blks_read`.
> Il valore di `blk_read_time` è la somma di `storage_blk_read_time`, `orcache_blk_read_time` e `local_blk_read_time`.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive
- 14.9 e versioni successive

Esempi

L'esempio seguente mostra come contiene tutte le colonne `pg_stat_database` e aggiunge 6 nuove colonne alla fine:

```
=> select * from aurora_stat_database() where datid=14717;
-[ RECORD 1 ]------------+------------------------------
datid                    | 14717
datname                  | postgres
numbackends              | 1
xact_commit              | 223
xact_rollback            | 4
blks_read                | 1059
blks_hit                 | 11456
tup_returned             | 27746
tup_fetched              | 5220
tup_inserted             | 165
tup_updated              | 42
tup_deleted              | 91
conflicts                | 0
temp_files               | 0
temp_bytes               | 0
deadlocks                | 0
checksum_failures        |
checksum_last_failure    |
blk_read_time            | 3358.689
```

```
blk_write_time           | 0
session_time             | 1076007.997
active_time              | 3684.371
idle_in_transaction_time | 0
sessions                 | 10
sessions_abandoned       | 0
sessions_fatal           | 0
sessions_killed          | 0
stats_reset              | 2023-01-12 20:15:17.370601+00
orcache_blks_hit         | 425
orcache_blk_read_time    | 89.934
storage_blks_read        | 623
storage_blk_read_time    | 3254.914
local_blks_read          | 0
local_blk_read_time      | 0
```

aurora_stat_dml_activity

Riporta l'attività cumulativa per ogni tipo di operazione di linguaggio di manipolazione dei dati (DML) su un database in un cluster Aurora PostgreSQL.

Sintassi

```
aurora_stat_dml_activity(database_oid)
```

Argomenti

database_oid

L'ID oggetto (OID) del database nel cluster Aurora PostgreSQL.

Tipo restituito

Record SETOF

Note per l'utilizzo

La funzione `aurora_stat_dml_activity` è disponibile solo con Aurora PostgreSQL versione 3.1 compatibile con il motore PostgreSQL 11.6 e versioni successive.

Utilizzare questa funzione sui cluster Aurora PostgreSQL con un gran numero di database per identificare quali database hanno un'attività DML più veloce o più lenta, o entrambi.

La funzione `aurora_stat_dml_activity` restituisce il numero di volte in cui le operazioni sono state eseguite e la latenza cumulativa in microsecondi per le operazioni SELECT, INSERT, UPDATE e DELETE. Il report include solo le operazioni DML riuscite.

È possibile reimpostare questa statistica utilizzando la funzione di accesso alle statistiche PostgreSQL `pg_stat_reset`. È possibile controllare l'ultima volta che questa statistica è stata reimpostata utilizzando la funzione `pg_stat_get_db_stat_reset_time`. Per ulteriori informazioni sulle funzioni di accesso alle statistiche PostgreSQL, consulta Raccolta di statistiche nella documentazione di PostgreSQL.

Esempi

L'esempio seguente mostra come eseguire il report delle statistiche di attività DML per il database connesso.

```
--Define the oid variable from connected database by using \gset
=> SELECT oid,
          datname
     FROM pg_database
    WHERE datname=(select current_database()) \gset
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
select_count | select_latency_microsecs | insert_count | insert_latency_microsecs |
 update_count | update_latency_microsecs | delete_count | delete_latency_microsecs
--------------+--------------------------+--------------+--------------------------
+--------------+--------------------------+--------------+--------------------------
       178957 |                 66684115 |       171065 |                 28876649 |
    519538 |            1454579206167 |            1 |                    53027


-- Showing the same results with expanded display on
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
-[ RECORD 1 ]------------+-------------
select_count             | 178957
select_latency_microsecs | 66684115
insert_count             | 171065
insert_latency_microsecs | 28876649
update_count             | 519538
update_latency_microsecs | 1454579206167
delete_count             | 1
delete_latency_microsecs | 53027
```

L'esempio seguente mostra le statistiche di attività DML per tutti i database nel cluster Aurora PostgreSQL. Questo cluster dispone di due database, `postgres` e `mydb`. L'elenco separato da virgole corrisponde ai campi `select_count`, `select_latency_microsecs`, `insert_count`, `insert_latency_microsecs`, `update_count`, `update_latency_microsecs`, `delete_count` e `delete_latency_microsecs`.

Aurora PostgreSQL crea e utilizza un database di sistema denominato `rdsadmin` per supportare operazioni amministrative quali backup, ripristini, controlli di integrità, replica e così via. Queste operazioni DML non hanno alcun impatto sul cluster Aurora PostgreSQL.

```
=> SELECT oid,
    datname,
    aurora_stat_dml_activity(oid)
    FROM pg_database;
oid  |    datname     |                   aurora_stat_dml_activity
-------+---------------
+-----------------------------------------------------------------
 14006 | template0      | (,,,,,,,)
 16384 | rdsadmin       | (2346623,1211703821,4297518,817184554,0,0,0,0)
     1 | template1      | (,,,,,,,)
 14007 | postgres       |
 (178961,66716329,171065,28876649,519538,1454579206167,1,53027)
 16401 | mydb           | (200246,64302436,200036,107101855,600000,83659417514,0,0)
```

Nell'esempio seguente vengono illustrate le statistiche di attività DML per tutti i database, organizzate in colonne per una migliore leggibilità.

```
SELECT db.datname,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 1), '()') AS select_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 2), '()') AS select_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 3), '()') AS insert_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 4), '()') AS insert_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 5), '()') AS update_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 6), '()') AS update_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 7), '()') AS delete_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 8), '()') AS delete_latency_microsecs
FROM  (SELECT datname,
              aurora_stat_dml_activity(oid) AS asdmla
         FROM pg_database
      ) AS db;
```

```
    datname     | select_count | select_latency_microsecs | insert_count |
 insert_latency_microsecs | update_count | update_latency_microsecs | delete_count |
 delete_latency_microsecs
---------------+--------------+--------------------------+--------------
+--------------------------+--------------+--------------------------+--------------
+--------------------------
 template0      |              |                          |              |
             |              |                          |              |
 rdsadmin       | 4206523      | 2478812333               | 7009414      | 1338482258
             | 0            | 0                        | 0            | 0
 template1      |              |                          |              |
             |              |                          |              |
 fault_test     | 66           | 452099                   | 0            | 0
             | 0            | 0                        | 0            | 0
 db_access_test | 1            | 5982                     | 0            | 0
             | 0            | 0                        | 0            | 0
 postgres       | 42035        | 95179203                 | 5752         | 2678832898
             | 21157        | 441883182488             | 2            | 1520
 mydb           | 71           | 453514                   | 0            | 0
             | 1            | 190                      | 1            | 152
```

Nell'esempio seguente viene illustrata la latenza cumulativa media (latenza cumulativa divisa per conteggio) per ogni operazione DML per il database con OID 16401.

```
=> SELECT select_count,
          select_latency_microsecs,
          select_latency_microsecs/NULLIF(select_count,0) select_latency_per_exec,
          insert_count,
          insert_latency_microsecs,
          insert_latency_microsecs/NULLIF(insert_count,0) insert_latency_per_exec,
          update_count,
          update_latency_microsecs,
          update_latency_microsecs/NULLIF(update_count,0) update_latency_per_exec,
          delete_count,
          delete_latency_microsecs,
          delete_latency_microsecs/NULLIF(delete_count,0) delete_latency_per_exec
     FROM aurora_stat_dml_activity(16401);
-[ RECORD 1 ]------------+-------------
select_count             | 451312
select_latency_microsecs | 80205857
select_latency_per_exec  | 177
insert_count             | 451001
insert_latency_microsecs | 123667646
```

```
insert_latency_per_exec  | 274
update_count             | 1353067
update_latency_microsecs | 200900695615
update_latency_per_exec  | 148478
delete_count             | 12
delete_latency_microsecs | 448
delete_latency_per_exec  | 37
```

aurora_stat_get_db_commit_latency

Ottiene la latenza di commit cumulativa in microsecondi per i database Aurora PostgreSQL. La Latenza commit viene misurata come il tempo che intercorre tra il momento in cui un client invia una richiesta di commit e il momento in cui riceve la conferma di commit.

Sintassi

```
aurora_stat_get_db_commit_latency(database_oid)
```

Argomenti

database_oid

L'ID oggetto (OID) del database Aurora PostgreSQL.

Tipo restituito

Record SETOF

Note per l'utilizzo

Amazon CloudWatch utilizza questa funzione per calcolare la latenza media di commit. Per ulteriori informazioni sui CloudWatch parametri di Amazon e su come risolvere i problemi relativi alla latenza di commit elevata, consulta Visualizzazione delle metriche nella console Amazon RDS Prendere decisioni migliori su Amazon RDS con i parametri di Amazon. CloudWatch

È possibile reimpostare questa statistica utilizzando la funzione di accesso alle statistiche PostgreSQL `pg_stat_reset`. È possibile controllare l'ultima volta che questa statistica è stata reimpostata utilizzando la funzione `pg_stat_get_db_stat_reset_time`. Per ulteriori informazioni sulle funzioni di accesso alle statistiche PostgreSQL, consulta Raccolta di statistiche nella documentazione di PostgreSQL.

Esempi

Nell'esempio seguente viene ottenuta la latenza di commit cumulativa per ogni database nel cluster `pg_database`.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database;

  oid  |    datname     | aurora_stat_get_db_commit_latency
-------+----------------+-----------------------------------
 14006 | template0      |                                 0
 16384 | rdsadmin       |                         654387789
     1 | template1      |                                 0
 16401 | mydb           |                            229556
 69768 | postgres       |                             22011
```

Nell'esempio seguente viene ottenuta la latenza di commit cumulativa per il database attualmente connesso. Prima di chiamare la funzione `aurora_stat_get_db_commit_latency`, l'esempio utilizza `\gset` per definire una variabile per `oid` e imposta il suo valore dal database connesso.

```
--Get the oid value from the connected database before calling
 aurora_stat_get_db_commit_latency
=> SELECT oid
     FROM pg_database
    WHERE datname=(SELECT current_database()) \gset
=> SELECT *
     FROM aurora_stat_get_db_commit_latency(:oid);

 aurora_stat_get_db_commit_latency
-----------------------------------
                        1424279160
```

Nell'esempio seguente viene ottenuta la latenza di commit cumulativa per il database `mydb` nel cluster `pg_database`. Quindi, reimposta questa statistica utilizzando la funzione `pg_stat_reset` e mostra i risultati. Infine, utilizza la funzione `pg_stat_get_db_stat_reset_time` per controllare l'ultima volta che questa statistica è stata reimpostata.

```
=> SELECT oid,
    datname,
```

```
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database
    WHERE datname = 'mydb';

  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                           3320370


=> SELECT pg_stat_reset();
 pg_stat_reset
---------------


=> SELECT oid,
          datname,
          aurora_stat_get_db_commit_latency(oid)
     FROM pg_database
    WHERE datname = 'mydb';
  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                                 6


=> SELECT *
     FROM pg_stat_get_db_stat_reset_time(16427);

 pg_stat_get_db_stat_reset_time
--------------------------------
 2021-04-29 21:36:15.707399+00
```

aurora_stat_logical_wal_cache

Mostra l'utilizzo della cache WAL (write-ahead log) logica per slot.

Sintassi

```
SELECT * FROM aurora_stat_logical_wal_cache()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `name`: il nome dello slot di replica.
- `active_pid`: ID del processo walsender.
- `cache_hit`: il numero totale di riscontri della cache wal dall'ultimo ripristino.
- `cache_miss`: il numero totale di riscontri mancati della cache wal dall'ultimo ripristino.
- `blks_read`: il numero totale di richieste di lettura della cache wal.
- `hit_rate`: la percentuale di riscontri nella cache WAL (cache_hit/blks_read).
- `last_reset_timestamp`: data e ora dell'ultimo ripristino del contatore.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL.

- 15.2 e tutte le versioni successive
- 14.7 e versioni successive
- 13.8 e versioni successive
- 12.12 e versioni successive
- 11.17 e versioni successive

Esempi

L'esempio seguente mostra due slot di replica con una sola funzione `aurora_stat_logical_wal_cache`attiva.

```
=> SELECT *
     FROM aurora_stat_logical_wal_cache();
    name    | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
------------+------------+-----------+------------+-----------+----------
+-------------------------------
 test_slot1 |      79183 |        24 |          0 |        24 | 100.00%  | 2022-08-05
 17:39:56.830635+00
 test_slot2 |            |         1 |          0 |         1 | 100.00%  | 2022-08-05
 17:34:04.036795+00
```

```
(2 rows)
```

aurora_stat_memctx_usage

Segnala l'utilizzo del contesto della memoria per ciascun processo PostgreSQL.

Sintassi

```
aurora_stat_memctx_usage()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `pid`: l'ID del processo.
- `name`: il nome del contesto della memoria.
- `allocated`: il numero di byte ottenuti dal sottosistema della memoria sottostante mediante il contesto della memoria.
- `used`: il numero di byte vincolati per i client del contesto della memoria.
- `instances`: il numero di contesti di questo tipo attualmente esistenti.

Note per l'utilizzo

Questa funzione visualizza l'utilizzo del contesto della memoria per ciascun processo PostgreSQL. Alcuni processi sono etichettati `anonymous`. I processi non sono esposti perché contengono parole chiave limitate.

Questa funzione è disponibile a partire dalle seguenti versioni di Aurora PostgreSQL:

- 15.3 o versioni successive alla 15
- 14.8 o versioni successive alla 14
- 13.11 o versioni successive alla 13
- 12.15 e versioni successive alla 12
- 11.20 e versioni successive alla 11

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata della funzione `aurora_stat_memctx_usage`.

```
=> SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                              | allocated |   used  | instances
-------+-----------------------------------+-----------+---------+-----------
123864 | Miscellaneous                     |     19520 |   15064 |         3
123864 | Aurora File Context               |      8192 |     616 |         1
123864 | Aurora WAL Context                |      8192 |     296 |         1
123864 | CacheMemoryContext                |    524288 |  422600 |         1
123864 | Catalog tuple context             |     16384 |   13736 |         1
123864 | ExecutorState                     |     32832 |   28304 |         1
123864 | ExprContext                       |      8192 |    1720 |         1
123864 | GWAL record construction          |      1024 |     832 |         1
123864 | MdSmgr                            |      8192 |     296 |         1
123864 | MessageContext                    |    532480 |  353832 |         1
123864 | PortalHeapMemory                  |      1024 |     488 |         1
123864 | PortalMemory                      |      8192 |     576 |         1
123864 | printtup                          |      8192 |     296 |         1
123864 | RelCache hash table entries       |      8192 |    8152 |         1
123864 | RowDescriptionContext             |      8192 |    1344 |         1
123864 | smgr relation context             |      8192 |     296 |         1
123864 | Table function arguments          |      8192 |     352 |         1
123864 | TopTransactionContext             |      8192 |     632 |         1
123864 | TransactionAbortContext           |     32768 |     296 |         1
123864 | WAL record construction           |     50216 |   43904 |         1
123864 | hash table                        |     65536 |   52744 |         6
123864 | Relation metadata                 |    191488 |  124240 |        87
104992 | Miscellaneous                     |      9280 |    7728 |         3
104992 | Aurora File Context               |      8192 |     376 |         1
104992 | Aurora WAL Context                |      8192 |     296 |         1
104992 ||Autovacuum Launcher               |      8192 |     296 |         1
104992 | Autovacuum database list          |     16384 |     744 |         2
104992 | CacheMemoryContext                |    262144 |  140288 |         1
104992 | Catalog tuple context             |      8192 |     296 |         1
104992 | GWAL record construction          |      1024 |     832 |         1
104992 | MdSmgr                            |      8192 |     296 |         1
104992 | PortalMemory                      |      8192 |     296 |         1
104992 | RelCache hash table entries       |      8192 |     296 |         1
104992 | smgr relation context             |      8192 |     296 |         1
```

```
104992 | Autovacuum start worker (tmp)   |      8192 |     296 |         1
104992 | TopTransactionContext           |     16384 |     592 |         2
104992 | TransactionAbortContext         |     32768 |     296 |         1
104992 | WAL record construction         |     50216 |   43904 |         1
104992 | hash table                      |     49152 |   34024 |         4
(39 rows)
```

Alcune parole chiave limitate verranno nascoste e l'aspetto dell'output sarà il seguente:

```
postgres=>SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                            | allocated |   used  | instances
-------+---------------------------------+-----------+---------+-----------
  5482 | anonymous                       |      8192 |     456 |         1
  5482 | anonymous                       |      8192 |     296 |         1
```

aurora_stat_optimized_reads_cache

Questa funzione mostra le statistiche relative alla cache a più livelli.

Sintassi

```
aurora_stat_optimized_reads_cache()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- `total_size`: dimensione totale della cache di Letture ottimizzate.
- `used_size`: dimensione della pagina utilizzata nella cache di Letture ottimizzate.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive

- 14.9 e versioni successive

Esempi

L'esempio seguente mostra l'output di un'istanza r6gd.8xlarge:

```
=> select pg_size_pretty(total_size) as total_size, pg_size_pretty(used_size)
                as used_size from aurora_stat_optimized_reads_cache();
total_size | used_size
-----------+-----------
1054 GB    | 975 GB
```

aurora_stat_plans

Restituisce una riga per ogni piano di esecuzione monitorato.

Sintassi

```
aurora_stat_plans(
    showtext
)
```

Argomenti

- showtext — Mostra il testo della query e del piano. I valori validi sono NULL, true o false. True mostrerà il testo della query e del piano.

Tipo restituito

Restituisce una riga per ogni piano tracciato che contiene tutte le colonne di `aurora_stat_statements` e le seguenti colonne aggiuntive.

- planid — identificatore del piano
- explain_plan — spiega il testo del piano
- tipo_piano:
  - `no plan`- nessun piano è stato acquisito
  - `estimate`- piano elaborato con costi stimati
  - `actual`- piano acquisito con EXPLAIN ANALYZE

- plan_captured_time — l'ultima volta che è stato acquisito un piano

Note per l'utilizzo

`aurora_compute_plan_id`deve essere abilitato e `pg_stat_statements` deve essere attivo affinché i piani possano essere `shared_preload_libraries` tracciati.

Il numero di piani disponibili è controllato dal valore impostato nel `pg_stat_statements.max` parametro. Quando `aurora_compute_plan_id` è abilitato, è possibile tenere traccia dei piani fino al valore specificato in`aurora_stat_plans`.

Questa funzione è disponibile a partire dalle versioni 14.10, 15.5 di Aurora PostgreSQL e per tutte le altre versioni successive.

Esempi

Nell'esempio seguente, i due piani relativi all'identificatore di query -5471422286312252535 vengono acquisiti e le statistiche delle istruzioni vengono tracciate dal planid.

```
db1=# select calls, total_exec_time, planid, plan_captured_time, explain_plan
db1-# from aurora_stat_plans(true)
db1-# where queryid = '-5471422286312252535'

calls    |  total_exec_time   |   planid    |      plan_captured_time       |
                 explain_plan
---------+--------------------+-------------+-------------------------------
+-----------------------------------------------------------------
 1532632 |  3209846.097107853 |  1602979607 | 2023-10-31 03:27:16.925497+00 | Update on
 pgbench_branches                                        +
         |                    |             |                               |   ->
 Bitmap Heap Scan on pgbench_branches                       +
         |                    |             |                               |
 Recheck Cond: (bid = 76)                                 +
         |                    |             |                               |         -
>  Bitmap Index Scan on pgbench_branches_pkey          +
         |                    |             |                               |
     Index Cond: (bid = 76)
   61365 | 124078.18012200127 | -2054628807 | 2023-10-31 03:20:09.85429+00  | Update on
 pgbench_branches                                        +
         |                    |             |                               |   ->
 Index Scan using pgbench_branches_pkey on pgbench_branches+
```

```
        |                    |            |                                |
Index Cond: (bid = 17)
```

aurora_stat_reset_wal_cache

Reimposta il contatore per la cache wal logica.

Sintassi

Per reimpostare uno slot specifico

```
SELECT * FROM aurora_stat_reset_wal_cache('slot_name')
```

Per ripristinare tutti gli slot

```
SELECT * FROM aurora_stat_reset_wal_cache(NULL)
```

Argomenti

`NULL` o `slot_name`

Tipo restituito

Messaggio di stato, stringa di testo

- Reimposta il contatore logico della cache wal. Messaggio di esito positivo. Questo testo viene restituito quando la funzione ha esito positivo.
- Lo slot di replica non è stato trovato Riprova. Messaggio di errore Questo testo viene restituito quando la funzione non riesce.

Note per l'utilizzo

Questa funzione è disponibile per le seguenti versioni.

- Aurora PostgreSQL 14.5 e versioni successive
- Aurora PostgreSQL 13.8 e versioni successive
- Aurora PostgreSQL 12.12 e versioni successive
- Aurora SQLversion Postgre 11.17 e versioni successive

Esempi

L'esempio seguente utilizza la funzione `aurora_stat_reset_wal_cache` per reimpostare uno slot denominato `test_results` e quindi tenta di reimpostare uno slot che non esiste.

```
=> SELECT *
     FROM aurora_stat_reset_wal_cache('test_slot');
aurora_stat_reset_wal_cache
-----------------------------------
 Reset the logical wal cache counter.
(1 row)
=> SELECT *
     FROM aurora_stat_reset_wal_cache('slot-not-exist');
aurora_stat_reset_wal_cache
-----------------------------------------------
 Replication slot not found. Please try again.
(1 row)
```

aurora_stat_statements

Mostra tutte le colonne `pg_stat_statements` e ne aggiunge altre alla fine.

Sintassi

```
aurora_stat_statements(showtext boolean)
```

Argomenti

mostra testo booleano

Tipo restituito

Record SETOF con tutte le colonne `pg_stat_statements` e le seguenti colonne aggiuntive. Per ulteriori informazioni sulle colonne `pg_stat_statements`, consulta `pg_stat_statements`.

È possibile ripristinare le statistiche per questa funzione utilizzando `pg_stat_statements_reset()`.

- `storage_blks_read`: numero totale di blocchi condivisi letti dall'archiviazione Aurora con questa istruzione.

- `orcache_blks_hit`: numero totale di accessi alla cache di Letture ottimizzate con questa istruzione.
- `storage_blk_read_time`— Se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato dall'istruzione per leggere i blocchi condivisi da aurora storage, in millisecondi, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `local_blk_read_time`— Se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato dall'istruzione per leggere i blocchi locali, in millisecondi, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `orcache_blk_read_time`— Se `track_io_timing` è abilitato, tiene traccia del tempo totale impiegato dall'istruzione per leggere i blocchi condivisi dalla cache di lettura ottimizzata, in millisecondi, altrimenti il valore è zero. Per ulteriori informazioni, consulta track_io_timing.
- `total_plan_peakmem`— Somma totale dei valori di memoria di picco durante la fase di pianificazione per tutte le chiamate a questa istruzione. Per visualizzare la memoria di picco media durante la pianificazione dell'istruzione, dividi questo valore per il numero di chiamate.
- `min_plan_peakmem`— Il valore di memoria di picco più basso rilevato durante la pianificazione di tutte le chiamate a questa istruzione.
- `max_plan_peakmem`— Il massimo valore di memoria di picco durante la pianificazione registrato in tutte le chiamate a questa dichiarazione.
- `total_exec_peakmem`— Somma totale dei valori di memoria di picco durante la fase di esecuzione per tutte le chiamate a questa istruzione. Per visualizzare la memoria di picco media durante l'esecuzione dell'istruzione, dividi questo valore per il numero di chiamate.
- `min_exec_peakmem`—Il più piccolo valore di memoria di picco, in byte, rilevato durante l'esecuzione in tutte le chiamate a questa istruzione.
- `max_exec_peakmem`— Valore massimo di memoria di picco, in byte, rilevato durante l'esecuzione in tutte le chiamate a questa istruzione.

> **Note**
>
> `total_plan_peakmenmin_plan_peakmem`, e `max_plan_peakmem` vengono monitorati solo quando l'impostazione `pg_stat_statements.track_planning` è attivata.

Note per l'utilizzo

Per utilizzare la funzione aurora_stat_statements (), dovete includere `pg_stat_statements` l'estensione nel parametro. `shared_preload_libraries`

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.4 e versioni successive
- 14.9 e versioni successive

Le colonne che mostrano i picchi di memoria sono disponibili nelle seguenti versioni:

- 16.3 e versioni successive
- 15.7 e versioni successive
- 14.12 e versioni successive

Esempi

L'esempio seguente mostra come trasporta tutte le colonne pg_stat_statements e alla fine aggiunge 11 nuove colonne:

```
=> select * from aurora_stat_statements(true) where query like 'with window_max%';
-[ RECORD 1 ]----------
+-----------------------------------------------------------------------------------------------
userid                 | 16409
dbid                   | 5
toplevel               | t
queryid                | -8347523682669847482
query                  | with window_max as (select custid, max(scratch) over (order by
 scratch rows between $1 preceding
and $2 following) wmax from ts) select sum(wmax), max(custid) from window_max
plans                  | 0
total_plan_time        | 0
min_plan_time          | 0
max_plan_time          | 0
mean_plan_time         | 0
stddev_plan_time       | 0
calls                  | 4
total_exec_time        | 254.105121
min_exec_time          | 57.503164000000005
max_exec_time          | 68.687418
```

```
mean_exec_time          | 63.52628025
stddev_exec_time        | 5.150765359979643
rows                    | 4
shared_blks_hit         | 200192
shared_blks_read        | 0
shared_blks_dirtied     | 0
shared_blks_written     | 0
local_blks_hit          | 0
local_blks_read         | 0
local_blks_dirtied      | 0
local_blks_written      | 0
temp_blks_read          | 0
temp_blks_written       | 0
blk_read_time           | 0
blk_write_time          | 0
temp_blk_read_time      | 0
temp_blk_write_time     | 0
wal_records             | 0
wal_fpi                 | 0
wal_bytes               | 0
jit_functions           | 0
jit_generation_time     | 0
jit_inlining_count      | 0
jit_inlining_time       | 0
jit_optimization_count  | 0
jit_optimization_time   | 0
jit_emission_count      | 0
jit_emission_time       | 0
storage_blks_read       | 0
orcache_blks_hit        | 0
storage_blk_read_time   | 0
local_blk_read_time     | 0
orcache_blk_read_time   | 0
total_plan_peakmem      | 0
min_plan_peakmem        | 0
max_plan_peakmem        | 0
total_exec_peakmem      | 6356224
min_exec_peakmem        | 1589056
max_exec_peakmem        | 1589056
```

aurora_stat_system_waits

Segnala le informazioni sugli eventi di attesa per l'istanza database di Aurora PostgreSQL.

Sintassi

```
aurora_stat_system_waits()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF

Note per l'utilizzo

Questa funzione restituisce il numero cumulativo di attese e tempo di attesa cumulativo per ogni evento di attesa generato dall'istanza database a cui si è attualmente connessi.

Il recordset restituito include i seguenti campi:

- `type_id`: ID del tipo di evento di attesa.
- `event_id`: ID dell'evento di attesa.
- `waits`: numero di volte in cui si è verificato l'evento di attesa.
- `wait_time`: tempo di attesa totale in microsecondi per questo evento.

Le statistiche restituite da questa funzione vengono reimpostate al riavvio di un'istanza database.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_stat_system_waits`.

```
=> SELECT *
    FROM aurora_stat_system_waits();
 type_id | event_id  |   waits   |  wait_time
---------+-----------+-----------+-------------
       1 |  16777219 |        11 |        12864
       1 |  16777220 |       501 |       174473
       1 |  16777270 |     53171 |     23641847
       1 |  16777271 |        23 |       319668
       1 |  16777274 |        60 |        12759
.
```

```
.
.
      10 | 167772231 |    204596 |    790945212
      10 | 167772232 |         2 |        47729
      10 | 167772234 |         1 |          888
      10 | 167772235 |         2 |           64
```

L'esempio seguente mostra come puoi utilizzare questa funzione insieme a `aurora_stat_wait_event` e `aurora_stat_wait_type` per produrre risultati più leggibili.

```
=> SELECT type_name,
             event_name,
             waits,
             wait_time
        FROM aurora_stat_system_waits()
NATURAL JOIN aurora_stat_wait_event()
NATURAL JOIN aurora_stat_wait_type();

 type_name |       event_name       |  waits   |  wait_time
-----------+------------------------+----------+--------------
 LWLock    | XidGenLock             |       11 |        12864
 LWLock    | ProcArrayLock          |      501 |       174473
 LWLock    | buffer_content         |    53171 |     23641847
 LWLock    | rdsutils               |        2 |        12764
 Lock      | tuple                  |    75686 |   2033956052
 Lock      | transactionid          |  1765147 |  47267583409
 Activity  | AutoVacuumMain         |   136868 |  56305604538
 Activity  | BgWriterHibernate      |     7486 |  55266949471
 Activity  | BgWriterMain           |     7487 |   1508909964
.
.
.
 IO        | SLRURead               |        3 |        11756
 IO        | WALWrite               | 52544463 |    388850428
 IO        | XactSync               |   187073 |    597041642
 IO        | ClogRead               |        2 |        47729
 IO        | OutboundCtrlRead       |        1 |          888
 IO        | OutboundCtrlWrite      |        2 |           64
```

aurora_stat_wait_event

Elenca tutti gli eventi di attesa supportati per Aurora PostgreSQL. Per ulteriori informazioni sulla sicurezza con Aurora PostgreSQL, consulta Eventi di attesa Amazon Aurora PostgreSQL.

Sintassi

```
aurora_stat_wait_event()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- type_id: ID del tipo di evento di attesa.
- event_id: ID dell'evento di attesa.
- type_name: nome del tipo di attesa
- event_name: nome dell’evento di attesa

Note per l'utilizzo

Per visualizzare i nomi degli eventi con tipi di eventi anziché IDs, utilizzate questa funzione insieme ad altre funzioni come `aurora_stat_wait_type` e`aurora_stat_system_waits`. I nomi degli eventi di attesa restituiti da questa funzione sono gli stessi di quelli restituiti dalla funzione `aurora_wait_report`.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_stat_wait_event`.

```
=>  SELECT *
    FROM aurora_stat_wait_event();

 type_id | event_id  |               event_name
---------+-----------+-----------------------------------------
       1 |  16777216 | <unassigned:0>
       1 |  16777217 | ShmemIndexLock
       1 |  16777218 | OidGenLock
       1 |  16777219 | XidGenLock
.
```

```
.
.
       9 | 150994945 | PgSleep
       9 | 150994946 | RecoveryApplyDelay
      10 | 167772160 | BufFileRead
      10 | 167772161 | BufFileWrite
      10 | 167772162 | ControlFileRead
.
.
.
      10 | 167772226 | WALInitWrite
      10 | 167772227 | WALRead
      10 | 167772228 | WALSync
      10 | 167772229 | WALSyncMethodAssign
      10 | 167772230 | WALWrite
      10 | 167772231 | XactSync
.
.
.
      11 | 184549377 | LsnAllocate
```

L'esempio seguente unisce `aurora_stat_wait_type` e `aurora_stat_wait_event` per restituire nomi di tipo e nomi di eventi per una migliore leggibilità.

```
=> SELECT *
    FROM aurora_stat_wait_type() t
    JOIN aurora_stat_wait_event() e
      ON t.type_id = e.type_id;

 type_id | type_name | type_id | event_id  |               event_name
---------+-----------+---------+-----------+-----------------------------------------
       1 | LWLock    |       1 |  16777216 | <unassigned:0>
       1 | LWLock    |       1 |  16777217 | ShmemIndexLock
       1 | LWLock    |       1 |  16777218 | OidGenLock
       1 | LWLock    |       1 |  16777219 | XidGenLock
       1 | LWLock    |       1 |  16777220 | ProcArrayLock
.
.
.
       3 | Lock      |       3 |  50331648 | relation
       3 | Lock      |       3 |  50331649 | extend
       3 | Lock      |       3 |  50331650 | page
       3 | Lock      |       3 |  50331651 | tuple
```

```
.
.
.
      10 | IO         |      10 | 167772214 | TimelineHistorySync
      10 | IO         |      10 | 167772215 | TimelineHistoryWrite
      10 | IO         |      10 | 167772216 | TwophaseFileRead
      10 | IO         |      10 | 167772217 | TwophaseFileSync
.
.
.
      11 | LSN        |      11 | 184549376 | LsnDurable
```

aurora_stat_wait_type

Elenca tutti i tipi di attesa supportati per Aurora PostgreSQL.

Sintassi

```
aurora_stat_wait_type()
```

Argomenti

Nessuno

Tipo restituito

Record SETOF avente le seguenti colonne:

- type_id: ID del tipo di evento di attesa.
- type_name: nome del tipo di attesa

Note per l'utilizzo

Per visualizzare i nomi degli eventi di attesa con tipi di eventi di attesa anziché IDs, usa questa funzione insieme ad altre funzioni come `aurora_stat_wait_event` e`aurora_stat_system_waits`. I nomi dei tipi di attesa restituiti da questa funzione sono gli stessi di quelli restituiti dalla funzione `aurora_wait_report`.

Esempi

Nell'esempio seguente vengono mostrati i risultati della chiamata alla funzione `aurora_stat_wait_type`.

```
=> SELECT *
     FROM aurora_stat_wait_type();
 type_id | type_name
---------+-----------
       1 | LWLock
       3 | Lock
       4 | BufferPin
       5 | Activity
       6 | Client
       7 | Extension
       8 | IPC
       9 | Timeout
      10 | IO
      11 | LSN
```

aurora_version

Restituisce il valore di stringa del numero di versione di Amazon Aurora PostgreSQL-Compatible Edition.

Sintassi

```
aurora_version()
```

Argomenti

Nessuno

Tipo restituito

Stringa CHAR o VARCHAR

Note per l'utilizzo

Questa funzione visualizza la versione del motore di database di Amazon Aurora PostgreSQL-Compatible Edition. Il numero di versione viene restituito come stringa formattata come*major*.

minor.*patch*. Per ulteriori informazioni sui numeri di versione del motore Aurora PostgreSQL, consulta Numero versione Aurora

Puoi scegliere quando applicare aggiornamenti di versione secondaria impostando la finestra di manutenzione per il cluster database Aurora PostgreSQL. Per scoprire come fare, consulta Manutenzione di un cluster database Amazon Aurora.

A partire dal rilascio di PostgreSQL versioni 13.3, 12.8, 11.13, 10.18 e per tutte le altre versioni successive, i numeri di versione di Aurora seguono i numeri di versione di PostgreSQL. Per ulteriori informazioni su questa versione, consulta Aggiornamenti di Amazon Aurora PostgreSQL nelle Note di rilascio di Aurora PostgreSQL.

Esempi

L'esempio seguente mostra i risultati della chiamata della funzione `aurora_version` su un cluster database Aurora PostgreSQL in esecuzione prima su PostgreSQL 12.7, Aurora PostgreSQL versione 4.2 e poi su Aurora PostgreSQL versione 13.3.

```
=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
```

Questo esempio mostra varie opzioni di utilizzo della funzione per ottenere maggiori dettagli sulla versione di Aurora PostgreSQL. In questo esempio Aurora presenta un numero di versione distinto rispetto a PostgreSQL.

```
=> SHOW SERVER_VERSION;
 server_version
----------------
 12.7
(1 row)

=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
```

```
(1 row)

=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 12.7
(1 row)

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------
 PostgreSQL 12.7 on aarch64-unknown-linux-gnu, compiled by aarch64-unknown-linux-gnu-
gcc (GCC) 7.4.0, 64-bit
(1 row)

=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------
 Aurora: 4.2.2 Compatible with PostgreSQL: 12.7
(1 row)
```

Nell'esempio successivo, la funzione viene utilizzata con le stesse opzioni dell'esempio precedente. In questo esempio il numero di versione di Aurora non è distinto rispetto al numero di versione di PostgreSQL.

```
=> SHOW SERVER_VERSION;
server_version
----------------
 13.3

=> SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 13.3
```

```
=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------------
 Aurora: 13.3.0 Compatible with PostgreSQL: 13.3
```

aurora_volume_logical_start_lsn

Restituisce il numero di sequenza di log (LSN) utilizzato per identificare l'inizio di un record nel flusso WAL (write-ahead log) logico del volume del cluster Aurora.

Sintassi

```
aurora_volume_logical_start_lsn()
```

Argomenti

Nessuno

Tipo restituito

`pg_lsn`

Note per l'utilizzo

Questa funzione identifica l'inizio del record nel flusso WAL logico per un determinato volume del cluster Aurora. È possibile utilizzare questa funzione durante l'aggiornamento della versione principale utilizzando la replica logica e la clonazione rapida Aurora per determinare il numero di sequenza di log in corrispondenza del quale viene eseguito uno snapshot o un clone del database. È quindi possibile utilizzare la replica logica per trasmettere continuamente i record di dati più recenti dopo il numero di sequenza di log e sincronizzare le modifiche da editore a sottoscrittore.

Per ulteriori informazioni sull'utilizzo della replica logica per l'aggiornamento di una versione principale, consulta Utilizzo della replica logica per eseguire l'aggiornamento a una versione principale per Aurora PostgreSQL.

Questa funzione è disponibile per le seguenti versioni di Aurora PostgreSQL:

- 15.2 o versioni successive alla 15
- 14.3 o versioni successive alla 14
- 13.6 o versioni successive alla 13
- 12.10 e versioni successive alla 12
- 11.15 e versioni successive alla 11
- 10.20 e versioni successive alla 10

Esempi

È possibile ottenere il numero di sequenza di log utilizzando la seguente query:

```
postgres=> SELECT aurora_volume_logical_start_lsn();

aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

aurora_wait_report

Questa funzione mostra l'attività dell'evento di attesa durante un periodo di tempo.

Sintassi

```
aurora_wait_report([time])
```

Argomenti

tempo (facoltativo)

Il tempo in secondi. Il valore predefinito è 10 secondi.

Tipo restituito

Record SETOF avente le seguenti colonne:

- type_name: nome del tipo di attesa
- event_name: nome dell'evento di attesa
- wait: numero di attese
- wait_time: tempo di attesa in millisecondi
- ms_per_wait: durata media di un'attesa in millisecondi
- waits_per_xact: media delle attese per ogni transazione
- ms_per_xact: durata media di una transizione in millisecondi

Note per l'utilizzo

Questa funzione è disponibile a partire da Aurora PostgreSQL versione 1.1 compatibile con PostgreSQL 9.6.6 e versioni successive.

Per utilizzare questa funzione è necessario innanzitutto creare l'estensione `aurora_stat_utils` Aurora PostgreSQL, nel modo seguente:

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Per ulteriori informazioni sulle versioni dell'estensione Aurora PostgreSQL disponibili, consulta Versioni delle estensioni per Amazon Aurora PostgreSQL in Note di rilascio di Aurora PostgreSQL.

Questa funzione calcola gli eventi di attesa a livello di istanza confrontando i dati statistici di due snapshot della funzione aurora_stat_system_waits () e di pg_stat_database, acquisite dalla vista delle statistiche di PostgreSQL.

Per ulteriori informazioni su `aurora_stat_system_waits()` e `pg_stat_database`, consulta Raccolta di statistiche nella documentazione di PostgreSQL.

Quando è in esecuzione, questa funzione acquisisce una snapshot iniziale, attende il numero di secondi specificato e acquisisce una seconda snapshot. La funzione confronta le due snapshot e restituisce la differenza. Questa differenza rappresenta l'attività dell'istanza in quell'intervallo di tempo.

Sull'istanza di scrittura, la funzione visualizza anche il numero di transazioni di cui è stato eseguito il commit e il TPS (transazioni al secondo). Questa funzione restituisce informazioni a livello di istanza e include tutti i database sull'istanza.

Esempi

Questo esempio mostra come creare l'estensione aurora_stat_utils per poter utilizzare la funzione aurora_wait_report.

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Questo esempio mostra come controllare il report delle attese per 10 secondi.

```
=> SELECT *
     FROM aurora_wait_report();
NOTICE:  committed 34 transactions in 10 seconds (tps 3)
 type_name |    event_name     | waits | wait_time | ms_per_wait | waits_per_xact |
 ms_per_xact
-----------+-------------------+-------+-----------+-------------+----------------
+-------------
 Client    | ClientRead        |    26 |  30003.00 |    1153.961 |           0.76 |
 882.441
 Activity  | WalWriterMain     |    50 |  10051.32 |     201.026 |           1.47 |
 295.627
 Timeout   | PgSleep           |     1 |  10049.52 |   10049.516 |           0.03 |
 295.574
 Activity  | BgWriterHibernate |     1 |  10048.15 |   10048.153 |           0.03 |
 295.534
 Activity  | AutoVacuumMain    |    18 |   9941.66 |     552.314 |           0.53 |
 292.402
 Activity  | BgWriterMain      |     1 |    201.09 |     201.085 |           0.03 |
   5.914
 IO        | XactSync          |    15 |     25.34 |       1.690 |           0.44 |
   0.745
 IO        | RelationMapRead   |    12 |      0.54 |       0.045 |           0.35 |
   0.016
 IO        | WALWrite          |    84 |      0.21 |       0.002 |           2.47 |
   0.006
 IO        | DataFileExtend    |     1 |      0.02 |       0.018 |           0.03 |
   0.001
```

Questo esempio mostra come controllare il report delle attese per 60 secondi.

```
=> SELECT *
     FROM aurora_wait_report(60);
NOTICE:  committed 1544 transactions in 60 seconds (tps 25)
```

```
 type_name |       event_name       |  waits  | wait_time | ms_per_wait |
 waits_per_xact | ms_per_xact
-----------+------------------------+---------+-----------+-------------
+---------------+-------------
 Lock      | transactionid          |    6422 | 477000.53 |      74.276 |
 4.16 |     308.938
 Client    | ClientRead             |    8265 | 270752.99 |      32.759 |
 5.35 |     175.358
 Activity  | CheckpointerMain       |       1 |  60100.25 |   60100.246 |
 0.00 |      38.925
 Timeout   | PgSleep                |       1 |  60098.49 |   60098.493 |
 0.00 |      38.924
 Activity  | WalWriterMain          |     296 |  60010.99 |     202.740 |
 0.19 |      38.867
 Activity  | AutoVacuumMain         |     107 |  59827.84 |     559.139 |
 0.07 |      38.749
 Activity  | BgWriterMain           |     290 |  58821.83 |     202.834 |
 0.19 |      38.097
 IO        | XactSync               |    1295 |  55220.13 |      42.641 |
 0.84 |      35.764
 IO        | WALWrite               | 6602259 |  47810.94 |       0.007 |
 4276.07 |      30.966
 Lock      | tuple                  |     473 |  29880.67 |      63.173 |
 0.31 |      19.353
 LWLock    | buffer_mapping         |     142 |   3540.13 |      24.930 |
 0.09 |       2.293
 Activity  | BgWriterHibernate      |     290 |   1124.15 |       3.876 |
 0.19 |       0.728
 IO        | BufFileRead            |    7615 |    618.45 |       0.081 |
 4.93 |       0.401
 LWLock    | buffer_content         |      73 |    345.93 |       4.739 |
 0.05 |       0.224
 LWLock    | lock_manager           |      62 |    191.44 |       3.088 |
 0.04 |       0.124
 IO        | RelationMapRead        |      72 |      5.16 |       0.072 |
 0.05 |       0.003
 LWLock    | ProcArrayLock          |       1 |      2.01 |       2.008 |
 0.00 |       0.001
 IO        | ControlFileWriteUpdate |       2 |      0.03 |       0.013 |
 0.00 |       0.000
 IO        | DataFileExtend         |       1 |      0.02 |       0.018 |
 0.00 |       0.000
```

```
IO        | ControlFileSyncUpdate  |       1 |      0.00 |       0.000 |
0.00 |       0.000
```

Amazon Aurora PostgreSQL parametri

La gestione del cluster di database di Amazon Aurora è uguale a quella delle istanze database Amazon RDS, ovvero utilizza i parametri di un gruppo parametri del database. Tuttavia, Amazon Aurora si differenzia da Amazon RDS in quanto un cluster DB Aurora ha più istanze DB. Alcuni parametri che utilizzi per gestire il cluster DB Amazon Aurora DB si applicano a tutto il cluster, mentre altri si applicano soltanto a una specifica istanza database nel cluster DB, come segue.

- Gruppo di parametri del cluster DB: un gruppo di parametri del cluster DB contiene il set di parametri di configurazione del motore applicabili in tutto il cluster DB Aurora. Ad esempio, la gestione della cache del cluster è una caratteristica di un cluster DB Aurora controllato dal parametro `apg_ccm_enabled` che fa parte del gruppo di parametri del cluster di database. Il gruppo di parametri del cluster DB contiene anche le impostazioni predefinite per il gruppo parametri del database per le istanze database che costituiscono il cluster.
- Gruppo parametri del database: un gruppo di parametri del database è l'insieme di valori di configurazione del motore che si applicano a un'istanza database specifica di quel tipo di motore. I gruppi parametri del database per il motore PostgreSQL DB vengono utilizzati da un'istanza database RDS per PostgreSQL e da un cluster database Aurora PostgreSQL. Queste impostazioni di configurazione si applicano alle proprietà che possono variare tra le istanze database all'interno di un cluster Aurora, ad esempio le dimensioni dei buffer di memoria.

I parametri a livello di cluster sono gestiti nei gruppi di parametri del cluster database. I parametri a livello di istanza sono gestiti nei gruppi parametri del database. Puoi gestire i parametri utilizzando la console Amazon RDS AWS CLI, o l'API Amazon RDS. Sono disponibili comandi separati per la gestione dei parametri a livello di cluster e a livello di istanza.

- Per gestire i parametri a livello di cluster in un gruppo di parametri del cluster DB, usa il comando - group. modify-db-cluster-parameter AWS CLI
- Per gestire i parametri a livello di istanza in un gruppo di parametri DB per un'istanza DB in un cluster DB, usa il comando. modify-db-parameter-group AWS CLI

Per ulteriori informazioni AWS CLI, consulta Using the AWS CLI nella Guida per l'AWS Command Line Interface utente.

Per ulteriori informazioni sui gruppi di parametri, consultare .

Visualizzazione del cluster di database Aurora PostgreSQL e dei parametri del database

È possibile visualizzare tutti i gruppi di parametri di default disponibili per le istanze DB Amazon RDS for PostgreSQL e per i cluster DB Aurora PostgreSQL nella AWS Management Console. I gruppi di parametri predefiniti per tutti i motori DB e i tipi e le versioni di cluster DB sono elencati per ogni AWS regione. Vengono elencati anche tutti i gruppi di parametri personalizzati.

Invece di visualizzarli in AWS Management Console, puoi anche elencare i parametri contenuti nei gruppi di parametri del cluster DB e nei gruppi di parametri DB utilizzando AWS CLI o l'API Amazon RDS. Ad esempio, per elencare i parametri in un gruppo di parametri del cluster DB, usa il describe-db-cluster-parameters AWS CLI comando seguente:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12
```

Il comando restituisce descrizioni JSON dettagliate di ciascun parametro. Per ridurre la quantità di informazioni restituite, è possibile specificare le preferenze utilizzando l'opzione `--query`. Ad esempio, puoi ottenere il nome del parametro, la relativa descrizione e i valori consentiti per il gruppo di parametri del cluster di database Aurora PostgreSQL 12 di default come segue:

In Linux, macOS, oppure Unix:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

In Windows:

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Un gruppo di parametri del cluster di DB Aurora include il gruppo di parametri dell'istanza database e i valori di default per un determinato motore di Aurora DB. È possibile ottenere l'elenco dei parametri

DB dallo stesso gruppo di parametri predefinito Aurora PostgreSQL utilizzando il comando illustrato di seguito. describe-db-parameters AWS CLI

In Linux, macOS, oppure Unix:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

In Windows:

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

I comandi precedenti restituiscono elenchi di parametri dal cluster di database o dal gruppo parametri del database con descrizioni e altri dettagli specificati nella query. Di seguito è riportata una risposta di esempio:

```
[
    [
        {
            "ParameterName": "apg_enable_batch_mode_function_execution",
            "ApplyType": "dynamic",
            "Description": "Enables batch-mode functions to process sets of rows at a
 time.",
            "AllowedValues": "0,1"
        }
    ],
    [
        {
            "ParameterName": "apg_enable_correlated_any_transform",
            "ApplyType": "dynamic",
            "Description": "Enables the planner to transform correlated ANY Sublink
 (IN/NOT IN subquery) to JOIN when possible.",
            "AllowedValues": "0,1"
        }
    ],...
```

Di seguito sono riportate le tabelle contenenti i valori per il parametro del cluster database di default e il parametro database per Aurora PostgreSQL versione 14.

Parametri a livello di cluster Aurora PostgreSQL

Puoi visualizzare i parametri a livello di cluster disponibili per una versione specifica di Aurora PostgreSQL utilizzando la AWS console di gestione, la CLI o l'API Amazon RDS. AWS Per informazioni sulla visualizzazione dei parametri in un gruppo di parametri del cluster database Aurora PostgreSQL nella console RDS, consultare Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster DB in Amazon Aurora.

Alcuni parametri a livello di cluster non sono disponibili in tutte le versioni e alcuni sono obsoleti. Per informazioni sulla visualizzazione dei parametri di una versione specifica di Aurora PostgreSQL, consultare Visualizzazione del cluster di database Aurora PostgreSQL e dei parametri del database.

Ad esempio, nella tabella seguente sono elencati i parametri disponibili nel gruppo di parametri del cluster database di default per Aurora PostgreSQL versione 14. Se si crea un cluster di database Aurora PostgreSQL senza specificare il proprio gruppo parametri del database personalizzato, il cluster di database viene creato utilizzando il gruppo di parametri cluster di database Aurora predefinito per la versione scelta, ad esempio `default.aurora-postgresql14`, `default.aurora-postgresql13` e così via.

Per un elenco dei parametri dell'istanza database per lo stesso gruppo di parametri del cluster database, consulta Parametri a livello di istanza Aurora PostgreSQL.

Nome del parametro	Descrizione	Default
ansi_constraint_trigger_ordering	Modifica l'ordine di attivazione dei trigger di vincolo in modo che siano compatibili con lo standard ANSI SQL.	–
ansi_force_foreign_key_checks	Assicurati che le operazioni referenziali come l'eliminazione a cascata o l'aggiornamento a cascata si verifichino sempre indipendentemente dai vari contesti di attivazione esistenti per l'operazione.	–
ansi_qualified_update_set_target	Tabella di supporto e qualificatori dello schema nelle istruzioni UPDATE ... SET.	–
apg_ccm_enabled	Abilita o disabilita la gestione della cache del cluster per il cluster.	–

Nome del parametro	Descrizione	Default
apg_enable_batch_mode_function_execution	Consente alle funzioni in modalità batch di elaborare serie di righe in una volta.	–
apg_enable_correlated_any_transform	Consente al pianificatore di trasformare il link secondario ANY (sottoquery IN/NOT IN) in JOIN quando possibile.	–
apg_enable_function_migration	Consente al pianificatore di migrare le funzioni scalari idonee alla clausola FROM.	–
apg_enable_not_in_transform	Consente al pianificatore di trasformare la query secondaria NOT IN in ANTI JOIN quando possibile.	–
apg_enable_remove_redundant_inner_joins	Consente al pianificatore di rimuovere i join interni ridondanti.	–
apg_enable_semijoin_push_down	Consente l'utilizzo di filtri semijoin per gli hash join.	–
apg_plan_mgmt.capture_plan_baselines	Modalità baseline di acquisizione del piano. manual (manuale): abilita l'acquisizione del piano per qualsiasi istruzione SQL, off: disabilita l'acquisizione del piano, automatic (automatica): abilita l'acquisizione del piano per le istruzioni in pg_stat_statements che soddisfano i criteri di idoneità.	off
apg_plan_mgmt.max_databases	Imposta il numero massimo di database che possono gestire le query utilizzando apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Imposta il numero massimo di piani che possono essere memorizzati nella cache da apg_plan_mgmt.	10000

Nome del parametro	Descrizione	Default
apg_plan_mgmt.plan_retention_period	Numero massimo di giorni dall'ultimo utilizzo di un piano prima che venga eliminato automaticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Costo totale stimato del piano al di sotto del quale verrà eseguito un piano non approvato.	0
apg_plan_mgmt.use_plan_baselines	Utilizzare solo piani approvati o fissi per le istruzioni gestite.	false
application_name	Imposta il nome dell'applicazione da riportare nelle statistiche e nei registri.	–
array_nulls	Abilita l'inserimento di elementi NULL negli array.	–
aurora_compute_plan_id	Monitora i piani di esecuzione delle query per rilevare i piani di esecuzione che contribuiscono al carico corrente del database e per tenere traccia delle statistiche sulle prestazioni dei piani di esecuzione nel tempo. Per ulteriori informazioni, consulta Monitoraggio dei piani di esecuzione delle query per Aurora PostgreSQL.	on
authentication_timeout	(s) Imposta il tempo massimo concesso per completare l'autenticazione del client.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE per la registrazione del piano.	–
auto_explain.log_buffers	Registra l'utilizzo dei buffer.	–
auto_explain.log_format	Formato EXPLAIN da utilizzare per la registrazione del piano.	–

Nome del parametro	Descrizione	Default
auto_explain.log_min_duration	Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrati i piani.	–
auto_explain.log_nested_statements	Registra le istruzioni nidificate.	–
auto_explain.log_timing	Raccogli i dati temporali, non solo il numero di righe.	–
auto_explain.log_triggers	Includi statistiche di attivazione nei piani.	–
auto_explain.log_verbose	Usa EXPLAIN VERBOSE per la registrazione del piano.	–
auto_explain.sample_rate	Frazione di query da elaborare.	–
autovacuum	Avvia il sottoprocesso autovacuum.	–
autovacuum_analyze_scale_factor	Numero di inserimenti, aggiornamenti o eliminazioni di tupla prima dell'analisi come una frazione di reltuple.	0.05
autovacuum_analyze_threshold	Numero minimo di inserti, aggiornamenti o eliminazioni di tupla prima dell'analisi.	–
autovacuum_freeze_max_age	Età nella quale eseguire l'autovacuum in una tabella per impedire il wraparound ID della transazione.	–
autovacuum_max_workers	Imposta il numero massimo di processi dipendenti di autovacuum in esecuzione simultanea	MASSIMO (DBInstanceClassMemory/64371566592 ,3)

Nome del parametro	Descrizione	Default
autovacuum_multixact_freeze_max_age	Età Multixact a cui eseguire l'autovacuum in una tabella per impedire il wraparound multixact.	–
autovacuum_naptime	(s) Periodo di inattività tra le esecuzioni di autovacuum.	5
autovacuum_vacuum_cost_delay	(ms) Ritardo del costo del vacuum, in millisecondi, per l'autovacuum.	5
autovacuum_vacuum_cost_limit	Quantità del costo del vacuum disponibile prima del napping, per l'autovacuum.	MASSIMO (log (/21474836480) *600,200) DBInstance ClassMemory
autovacuum_vacuum_insert_scale_factor	Numero di inserti, aggiornamenti o eliminazioni di tupla prima di eseguire il vacuum come una frazione di reltuple.	–
autovacuum_vacuum_insert_threshold	Numero minimo di inserimenti tupla prima del vacuum o -1 per disabilitare i vacuum.	–
autovacuum_vacuum_scale_factor	Numero di aggiornamenti o eliminazioni di tupla prima del vacuum come una frazione di reltuple.	0.1
autovacuum_vacuum_threshold	Numero minimo di aggiornamenti o eliminazioni di tupla prima del vacuum.	–
autovacuum_work_mem	(kB) Imposta la memoria massima da utilizzare per ogni processo di dipendente autovacuum.	MASSIMO DBInstance ClassMemory (/32768 ,131072)
babelfishpg_tds.default_server_name	Il nome del server Babelfish predefinito	Microsoft SQL Server

Nome del parametro	Descrizione	Default
babelfishpg_tds.listen_address	Imposta il nome host o l'indirizzo o gli indirizzi IP su cui ascoltare TDS.	*
babelfishpg_tds.port	Imposta la porta TDS TCP su cui il server è in ascolto.	1433
babelfishpg_tds.tds_debug_log_level	Imposta il livello di registrazione in TDS, 0 disabilita la registrazione	1
babelfishpg_tds.tds_default_numeric_precision	Imposta la precisione predefinita del tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno.	38
babelfishpg_tds.tds_default_numeric_scale	Imposta la scala predefinita di tipo numerico da inviare nei metadati della colonna TDS se il motore non ne specifica uno.	8
babelfishpg_tds.tds_default_packet_size	Imposta la dimensione predefinita del pacchetto per tutti i client SQL Server in fase di connessione	4096
babelfishpg_tds.tds_default_protocol_version	Imposta una versione del protocollo TDS predefinita per tutti i client in fase di connessione	DEFAULT
babelfishpg_tds.tds_ssl_encrypt	Imposta l'opzione di crittografia SSL	0
babelfishpg_tds.tds_ssl_max_protocol_version	Imposta la versione massima del protocollo SSL/TLS da utilizzare per la sessione TDS.	TLSv12.

Nome del parametro	Descrizione	Default
babelfishpg_tds.tds_ssl_min_protocol_version	Imposta la versione minima del protocollo SSL/TLS da utilizzare per la sessione TDS.	TLSv1.2 da Aurora PostgreSQL versione 16, TLSv1 per versioni precedenti a Aurora PostgreSQL versione 16
babelfishpg_tsql.default_locale	Localizzazione predefinita da utilizzare per le regole di confronto create tramite CREATE COLLATION.	it-IT
babelfishpg_tsql.migration_mode	Definisce se sono supportati più database utente	multi-db da Aurora PostgreSQL versione 16, single-db per versioni precedenti a Aurora PostgreSQL versione 16
babelfishpg_tsql.server_collation_name	Nome delle regole di confronto del server predefinite	SQL_Latin1_General_CP1_CI_AS (default)
babelfishpg_tsql.versione	Imposta l'output della variabile @ @VERSION	default
backend_flush_after	(8Kb) Numero di pagine dopo le quali le scritture eseguite in precedenza vengono riportate su disco.	–
backslash_quote	Imposta se una doppia barra rovesciata \\ è consentita nelle stringhe letterali.	–
backtrace_functions	Registra backtrace per errori in queste funzioni.	–
bytea_output	Imposta il formato di output per byte.	–
check_function_bodies	Controlla i corpi delle funzioni durante CREATE FUNCTION.	–

Nome del parametro	Descrizione	Default
client_connection_check_interval	Imposta l'intervallo di tempo tra i controlli di disconnessione durante l'esecuzione di query.	–
client_encoding	Imposta la codifica del set di caratteri del client.	UTF8
client_min_messages	Imposta i livelli dei messaggi che vengono inviati al client.	–
compute_query_id	Calcola gli identificatori delle query.	auto
config_file	Imposta il file di configurazione principale del server.	/.conf rdsdbdata/config/postgresql
constraint_exclusion	Consente al pianificatore di utilizzare i vincoli per ottimizzare le query.	–
cpu_index_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna voce di indice durante una scansione dell'indice.	–
cpu_operator_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna chiamata dell'operatore o della funzione.	–
cpu_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna tupla (riga).	–
cron.database_name	Imposta il database per archiviare le tabelle di metadati pg_cron	postgres
cron.log_run	Registra tutti i processi eseguiti nella tabella job_run_details	on
cron.log_statement	Registra tutte le istruzioni cron prima dell'esecuzione.	off
cron.max_running_jobs	Numero massimo di processi che possono essere eseguiti contemporaneamente.	5

Nome del parametro	Descrizione	Default
cron.use_background_workers	Abilita i dipendente in background per pg_cron	on
cursor_tuple_fraction	Imposta la stima del pianificatore della frazione delle righe del cursore che verranno recuperate.	–
data_directory	Imposta la directory dei dati dei server.	/rdsdbdata/db
datestyle	Imposta il formato del display per i valori di data e ora.	–
db_user_namespace	Abilita i nomi utente per database.	–
deadlock_timeout	(ms) Imposta il tempo di attesa su un lock prima di verificare il deadlock.	–
debug_pretty_print	I trattini analizzano e visualizzano le visualizzazioni dell'albero.	–
debug_print_parse	Registra ogni albero di analisi della query.	–
debug_print_plan	Registra ogni programma di esecuzione della query.	–
debug_print_rewritten	Registra ogni albero di analisi riscritto della query.	–
default_statistics_target	Imposta la destinazione della statistica predefinita.	–
default_tablespace	Imposta il tablespace predefinito per la creazione di tabelle e indici.	–
default_toast_compression	Imposta il metodo di compressione predefinito per i valori comprimibili.	–

Nome del parametro	Descrizione	Default
default_transaction_deferrable	Imposta lo stato differibile predefinito delle nuove transazioni.	–
default_transaction_isolation	Imposta il livello di isolamento della transazione di ogni nuova transazione.	–
default_transaction_read_only	Imposta lo stato di sola lettura predefinito delle nuove transazioni.	–
effective_cache_size	(8 kB) Imposta l'ipotesi del pianificatore sulla dimensione della cache del disco.	SOMMA (DBInstanceClassMemory/12038, -50003)
effective_io_concurrency	Numero di richieste simultanee che possono essere gestite in modo efficace dal sottosistema del disco.	–
enable_async_append	Consente ai pianificatori di utilizzare piani di aggiunta asincroni.	–
enable_bitmapscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione bitmap.	–
enable_gathermerge	Abilita l'utilizzo da parte del pianificatore di piani di unione di raccolta.	–
enable_hashagg	Abilita l'utilizzo da parte del pianificatore di piani di aggregazione hash.	–
enable_hashjoin	Abilita l'utilizzo da parte del pianificatore di piani di unione hash.	–
enable_incremental_sort	Consente ai pianificatori di utilizzare fasi incrementali di ordinamento.	–
enable_indexonlyscan	Consente ai pianificatori di utilizzare i piani. index-only-scan	–

Nome del parametro	Descrizione	Default
enable_indexscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione dell'indice.	–
enable_material	Abilita l'utilizzo da parte del pianificatore della materializzazione.	–
enable_memoize	Abilita l'uso della memorizzazione da parte dei pianificatori	–
enable_mergejoin	Abilita l'utilizzo da parte del pianificatore di piani di unione.	–
enable_nestloop	Abilita l'utilizzo da parte del pianificatore di piani di unione a ciclo nested.	–
enable_parallel_append	Consente ai pianificatori di utilizzare piani di aggiunta paralleli.	–
enable_parallel_hash	Consente ai pianificatori di utilizzare piani di hash paralleli.	–
enable_partition_pruning	Abilita l'eliminazione delle partizioni del tempo di pianificazione e del runtime.	–
enable_partitionwise_aggregate	Consente l'aggregazione e il raggruppamento a livello di partizione.	–
enable_partitionwise_join	Attiva il join a livello di partizione.	–
enable_seqscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione sequenziali.	–
enable_sort	Abilita l'utilizzo da parte del pianificatore di passaggi di ordinamento espliciti.	–
enable_tidscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione TID.	–

Nome del parametro	Descrizione	Default
escape_string_warning	Avvisa circa la perdita di barre rovesciate nelle stringhe letterali ordinarie.	–
exit_on_error	Termina la sessione in caso di errore.	–
extra_float_digits	Imposta il numero di cifre visualizzate per i valori del punto variabile.	–
force_parallel_mode	Forza l'uso di strutture di query parallele.	–
from_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale le sottoquery non vengono compresse.	–
geqo	Abilita l'ottimizzazione genetica delle query.	–
geqo_effort	GEQO: lo sforzo viene utilizzato per impostare il valore predefinito per altri parametri GEQO.	–
geqo_generations	GEQO: numero di iterazioni dell'algoritmo.	–
geqo_pool_size	GEQO: numero di individui nella popolazione.	–
geqo_seed	GEQO: seme per la selezione casuale del percorso.	–
geqo_selection_bias	GEQO: pressione selettiva all'interno della popolazione.	–
geqo_threshold	Imposta la soglia degli elementi FROM oltre i quali viene utilizzato GEQO.	–
gin_fuzzy_search_limit	Imposta il risultato massimo consentito per la ricerca esatta da GIN.	–
gin_pending_list_limit	(kB) Imposta la dimensione massima dell'elenco in sospeso per l'indice GIN.	–

Nome del parametro	Descrizione	Default
hash_mem_multiplier	Multipli di work_mem da utilizzare per le tabelle hash.	–
hba_file	Imposta il file di configurazione hba dei server.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Consente il feedback da uno standby a caldo all'elemento primario che eviterà conflitti di query.	on
huge_pages	Riduce il sovraccarico quando un'istanza database lavora con grandi blocchi di memoria contigui, come quelli utilizzati dai buffer condivisi. È attivato per impostazione predefinita per tutte le classi di istanza database diverse dalle classi di istanza t3.medium, db.t3.large, db.t4g.medium, db.t4g.large	on
ident_file	Imposta il file di configurazione ident dei server.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Imposta la durata massima concessa di ogni transazione inattiva.	86400000
idle_session_timeout	Termina qualsiasi sessione che è rimasta inattiva (ovvero in attesa di una query del client) per un periodo di tempo specificato, ma che non è all'interno di una transazione aperta	–
intervalstyle	Imposta il formato del display per i valori dell'intervallo.	–
join_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale i costrutti JOIN non vengono appiattiti.	–

Nome del parametro	Descrizione	Default
krb_caseins_users	Indica se i nomi utente GSSAPI (Generic Security Service API) devono essere trattati senza distinzione tra maiuscole e minuscole (true) o meno. Per impostazione predefinita, questo parametro è impostato su false, quindi Kerberos suppone che i nomi utente applichin o la distinzione tra maiuscole e minuscole. Per ulteriori informazioni, consulta la pagina GSSAPI Authentication (Autenticazione GSSAPI) nella documentazione di PostgreSQL.	false
lc_messages	Imposta la lingua nella quale vengono visualizz ati i messaggi.	–
lc_monetary	Imposta l'ambientazione per la formattazione degli importi monetari.	–
lc_numeric	Imposta l'ambientazione per la formattazione degli numeri.	–
lc_time	Imposta l'ambientazione per la formattazione dei valori di data e ora.	–
listen_addresses	Imposta il nome host o l'indirizzo o gli indirizzi IP da ascoltare.	*
lo_compat_privileges	Abilita la modalità di compatibilità con le versioni precedenti per i controlli dei privilegi su oggetti di grandi dimensioni.	0
log_autovacuum_min_duration	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrate le azioni di autovacuum.	10000
log_connections	Registra ogni connessione riuscita.	–

Nome del parametro	Descrizione	Default
log_destination	Imposta la destinazione per l'output del registro del server.	stderr
log_directory	Imposta la directory di destinazione per i file di file di log.	/rdsdbdata/log/error
log_disconnections	Registra la fine di una sessione, compresa la durata.	–
log_duration	Registra la durata di ogni istruzione SQL completata.	–
log_error_verbosity	Imposta la verbosità dei messaggi registrati.	–
log_executor_stats	Scrive le statistiche sulla prestazione degli esecutori nel registro del server.	–
log_file_mode	Imposta le autorizzazioni del file per i file di file di log.	0644
log_filename	Imposta il modello del nome del file per i file di registro.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Avvia un processo secondario per acquisire l'output stderr e/o csvlogs nei file di log.	1
log_hostname	Registra il nome host nei registri delle connessioni.	0
logical_decoding_work_mem	(kB) Questa quantità di memoria può essere utilizzata da ogni buffer interno di riordino prima del versamento su disco.	–
log_line_prefix	Controlla le informazioni con prefisso su ciascuna riga di registro.	%t:%r:%u@%d:%p]:
log_lock_waits	Registra lunghe attese di lock.	–

Nome del parametro	Descrizione	Default
log_duration_sample	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verrà registrato un campione di istruzioni. Il campionamento è determinato da log_statement_sample_rate.	–
log_min_duration_statement	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrate le istruzioni.	–
log_min_error_statement	Fa sì che tutte le istruzioni che generano un errore pari o superiore a questo livello vengano registrate.	–
log_min_messages	Imposta i livelli dei messaggi che vengono registrati.	–
log_parameter_max_length	(B) Quando si registrano le istruzioni, limita i valori dei parametri registrati ai primi N byte.	–
log_parameter_max_length_on_error	(B) Quando si segnala un errore, limita i valori dei parametri registrati ai primi N byte.	–
log_parser_stats	Scrive le statistiche sulla prestazione del decodificatore nel registro del server.	–
log_planner_stats	Scrive le statistiche sulla prestazione del programmatore nel registro del server.	–
log_replication_commands	Registra ogni comando di replica.	–
log_rotation_age	(min) La rotazione del file di log automatico avverrà dopo N minuti.	60
log_rotation_size	(kB) La rotazione del file di log automatico avverrà dopo N kilobyte.	100000

Nome del parametro	Descrizione	Default
log_statement	Imposta il tipo di istruzioni registrate.	–
log_statement_sample_rate	Frazione di istruzioni che superano log_min_duration_sample da registrare.	–
log_statement_stats	Scrive le statistiche cumulative sulla prestazione nel registro del server.	–
log_temp_files	(kB) Registra l'uso di file temporanei più grandi di questo numero di kilobyte.	–
log_timezone	Imposta il fuso orario da utilizzare nei messaggi di registro.	UTC
log_transaction_sample_rate	Imposta la frazione di transazioni da registrare per le nuove transazioni.	–
log_truncate_on_rotation	Tronca i file di log esistenti con lo stesso nome durante la rotazione del registro.	0
maintenance_io_concurrency	Una variante di effective_io_concurrency utilizzata per i lavori di manutenzione.	1
maintenance_work_mem	(kB) Imposta la memoria massima da utilizzare per le operazioni di manutenzione.	MASSIMO (DBInstanceClassMemory/63963136 *1024,65536)
max_connections	Imposta il numero massimo di connessioni simultanee.	MINIMO (DBInstanceClassMemory/9531392 5.000)
max_files_per_process	Imposta il numero massimo di file aperti in modo simultaneo per ogni processo del server.	–
max_locks_per_transaction	Imposta il numero massimo di lock per transazione.	64

Nome del parametro	Descrizione	Default
max_logical_replication_workers	Numero massimo di processi di dipendente di replica logica.	–
max_parallel_maintenance_workers	Imposta il numero massimo di processi paralleli per operazione di manutenzione.	–
max_parallel_workers	Imposta il numero massimo di worker paralleli che possono essere attivi in una sola volta.	MASSIMO ($VCPU/2,8) DBInstance
max_parallel_workers_per_gather	Imposta il numero massimo di processi paralleli per nodo executor.	–
max_pred_locks_per_page	Imposta il numero massimo di tuple bloccate dal predicato per pagina.	–
max_pred_locks_per_relation	Imposta il numero massimo di pagine e tuple bloccate dal predicato per relazione.	–
max_pred_locks_per_transaction	Imposta il numero massimo di lock del predicato per transazione.	–
max_prepared_transactions	Imposta il numero massimo di transazioni preparati in modo simultaneo.	0
max_replication_slots	Imposta il numero massimo di slot di replica che il server può supportare.	20
max_slot_wal_keep_size	(MB) Gli slot di replica saranno contrassegnati come non riusciti e i segmenti rilasciati per l'eliminazione o il riciclaggio se questo spazio è occupato dai WAL su disco.	–
max_stack_depth	(kB) Imposta la profondità massima della pila in kilobyte.	6144

Nome del parametro	Descrizione	Default
max_standby_streaming_delay	(ms) Imposta il ritardo massimo prima di annullare le query quando un server con standby a caldo elabora i dati WAL in streaming .	14000
max_sync_workers_per_subscription	Numero massimo di worker di sincronizzazione per abbonamento	2
max_wal_senders	Imposta il numero massimo di processi del mittente WAL in esecuzione simultanea	10
max_worker_processes	Imposta il numero massimo di processi dipendente simultanei.	MASSIMO ($VCPU*2,8) DBInstance
min_dynamic_shared_memory	(MB) Quantità di memoria condivisa dinamica riservata all'avvio.	–
min_parallel_index_scan_size	(8 kB) Imposta la quantità minima di dati di indicizzazione per una scansione parallela.	–
min_parallel_table_scan_size	(8 kB) Imposta la quantità minima di dati della tabella per una scansione parallela.	–
old_snapshot_threshold	(min) Tempo prima che uno snapshot sia troppo vecchio per leggere le pagine modificate dopo l'acquisizione dello snapshot.	–
orafce.nls_date_format	Emula il comportamento di output oracles date.	–
orafce.timezone	Specifica il fuso orario utilizzato per la funzione sysdate.	–
parallel_leader_participation	Controlla se Gather e Gather Merge eseguono anche piani secondari.	–

Nome del parametro	Descrizione	Default
parallel_setup_cost	Imposta la stima dei pianificatori del costo di avvio dei processi dei dipendenti per la query parallela.	–
parallel_tuple_cost	Imposta la stima del pianificatore del costo del passaggio di ciascuna tupla (riga) dal back-end dipendente a quello principale.	–
password_encryption	Crittografa le password.	–
pgaudit.log	Specifica quali classi di istruzioni verranno registrate per registrazione di verifica della sessione.	–
pgaudit.log_catalog	Specifica che la registrazione delle sessioni deve essere abilitata nel caso in cui tutte le relazioni in un'istruzione siano in pg_catalog.	–
pgaudit.log_level	Specifica il livello di log che verrà utilizzato per le voci di registro.	–
pgaudit.log_parameter	Specifica che la registrazione di verifica deve includere i parametri passati con l'istruzione.	–
pgaudit.log_relation	Specifica se la registrazione di verifica della sessione deve creare una voce di registro separata per ogni relazione (TABLE, VIEW, ecc.) a cui fa riferimento in un'istruzione SELECT o DML.	–
pgaudit.log_statement_once	Specifica se la registrazione includerà il testo dell'istruzione e i parametri con la prima voce di registro per una combinazione istruzione/istruzione secondaria o con ogni voce.	–
pgaudit.role	Specifica il ruolo principale da utilizzare per la registrazione di verifica degli oggetti.	–

Nome del parametro	Descrizione	Default
pg_bigm.enable_recheck	Specifica se eseguire Recheck, che è un processo interno di ricerca full-text.	on
pg_bigm.gin_key_limit	Specifica il numero massimo di 2-grammi della parola chiave di ricerca da utilizzare per la ricerca full-text.	0
pg_bigm.last_update	Riporta l'ultima data aggiornata del modulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Specifica la soglia minima utilizzata dalla ricerca di somiglianza.	0.3
pg_hint_plan.debug_print	Registra i risultati dell'analisi dei suggerimenti.	–
pg_hint_plan.enable_hint	Forza il pianificatore a utilizzare i piani specificati nel commento del suggerimento precedente alla query.	–
pg_hint_plan.enable_hint_table	Forza il pianificatore a non ottenere indicazioni utilizzando le ricerche nella tabella.	–
pg_hint_plan.message_level	Livello di messaggio dei messaggi di debug.	–
pg_hint_plan.parse_messages	Livello di messaggio degli errori di analisi.	–
pglogical.batch_inserts	Inserimenti batch, se possibile	–
pglogical.conflict_log_level	Imposta il livello di registro utilizzato per la registrazione dei conflitti risolti.	–
pglogical.conflict_resolution	Imposta il metodo utilizzato per la risoluzione dei conflitti per i conflitti risolvibili.	–

Nome del parametro	Descrizione	Default
pglogical.extra_connection_options	opzioni di connessione da aggiungere a tutte le connessioni dei nodi peer	–
pglogical.synchronous_commit	Valore di commit sincrono specifico pglogical	–
pglogical.use_spi	Usa SPI invece dell'API di basso livello per applicare le modifiche	–
pgtle.clientauth_databases_to_skip	Elenco di database da saltare per la funzionalità clientauth.	–
pgtle.clientauth_db_name	Controlla quale database viene utilizzato per la funzionalità clientauth.	–
pgtle.clientauth_num_parallel_workers	Numero di lavoratori in background utilizzati per la funzionalità clientauth.	–
pgtle.clientauth_users_to_skip	Elenco di utenti da saltare per la funzionalità clientauth.	–
pgtle.enable_clientauth	Abilita la funzionalità clientauth.	–
pgtle.passcheck_db_name	Imposta quale database viene utilizzato per la funzionalità passcheck a livello di cluster.	–
pg_prewarm.autoprewarm	Avvia il dipendente di pre-riscaldamento automatico.	–
pg_prewarm.autoprewarm_interval	Imposta l'intervallo tra i dump dei buffer condivisi	–
pg_similarity.block_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.block_threshold	Imposta la soglia utilizzata dalla funzione di similarità dei blocchi.	–

Nome del parametro	Descrizione	Default
pg_similarity.block_tokenizer	Imposta il tokenizzatore per la funzione di similarità dei blocchi.	–
pg_similarity.cosine_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.cosine_threshold	Imposta la soglia utilizzata dalla funzione di similarità del coseno.	–
pg_similarity.cosine_tokenizer	Imposta il tokenizzatore per la funzione di similarità del coseno.	–
pg_similarity.dice_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.dice_threshold	Imposta la soglia utilizzata dalla misura di similarità di Dice.	–
pg_similarity.dice_tokenizer	Imposta il tokenizzatore per la misura di similarità di Dice.	–
pg_similarity.euclidean_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.euclidean_threshold	Imposta la soglia utilizzata dalla misura di similarità euclidea.	–
pg_similarity.euclidean_tokenizer	Imposta il tokenizzatore per la misura di similarità euclidea.	–
pg_similarity.hamming_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.hamming_threshold	Imposta la soglia utilizzata dal parametro di similarità dei blocchi.	–
pg_similarity.jaccard_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.jaccard_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaccard.	–
pg_similarity.jaccard_tokenizer	Imposta il tokenizzatore per la misura di similarità di Jaccard.	–
pg_similarity.jaro_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaro_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaro.	–
pg_similarity.jarowinkler_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jarowinkler_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.levenshtein_threshold	Imposta la soglia utilizzata dalla misura di similarità di Levenshtein.	–
pg_similarity.matching_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.matching_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.matching_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.mongeelkan_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.mongeelkan_threshold	Imposta la soglia utilizzata dalla misura di similarità di Monge-Elkan.	–

Nome del parametro	Descrizione	Default
pg_similarity.mongeelkan_tokenizer	Imposta il tokenizzatore per la misura di similarità di Monge-Elkan.	–
pg_similarity.nw_gap_penalty	Imposta la penalità del gap utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.nw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.overlap_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.overlap_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.qgram_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.qgram_threshold	Imposta la soglia utilizzata dalla misura di similitudine Q-Gram.	–
pg_similarity.qgram_tokenizer	Imposta il tokenizzatore per la misura Q-Gram.	–
pg_similarity.swg_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarità.swg_threshold	Imposta la soglia utilizzata dalla misura di somiglianza Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.sw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Smith-Waterman.	–
pg_stat_statements.max	Imposta il numero massimo di istruzioni monitorate da pg_stat_statements.	–
pg_stat_statements.save	Salva le statistiche pg_stat_statements attraverso le interruzioni del server.	–
pg_stat_statements.track	Seleziona le istruzioni che vengono monitorate da pg_stat_statements.	–
pg_stat_statements.track_planning	Seleziona se la durata della pianificazione è monitorata da pg_stat_statements.	–
pg_stat_statements.track_utility	Seleziona se i comandi di utilità sono monitorati da pg_stat_statements.	–
plan_cache_mode	Controlla la selezione da parte del pianificatore di un piano personalizzato o generico.	–
port	Imposta la porta TCP su cui il server è in ascolto.	EndPointPort
postgis.gdal_enabled_drivers	Abilita o disabilita i driver GDAL utilizzati con PostGIS in Postgres 9.3.5 e versioni successive.	ENABLE_ALL
quote_all_identifiers	Aggiungi le virgolette a tutti gli identificatori quando si generano i frammenti SQL.	–
random_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo non sequenziale.	–

Nome del parametro	Descrizione	Default
rdkit.dice_threshold	Soglia inferiore di somiglianza Dice. Le molecole con somiglianza inferiore alla soglia non sono simili per operazione #.	–
rdkit.do_chiral_sss	È bene prendere in considerazione la stereochimica nella corrispondenza delle sottostrutture. Se false, non vengono utilizzate informazioni stereochimiche nelle corrispondenze della sottostruttura.	–
rdkit.tanimoto_threshold	Soglia inferiore di somiglianza Tanimoto. Le molecole con somiglianza inferiore alla soglia non sono simili per operazione %.	–
rds.accepted_password_auth_method	Forza l'autenticazione per le connessioni con password archiviata localmente.	md5+scram
rds.adaptive_autovacuum	Parametro RDS per abilitare/disabilitare l'autovacuum adattivo.	1
rds.babelfish_status	Parametro RDS per abilitare/disabilitare Babelfish per Aurora PostgreSQL.	off
rds.enable_plan_management	Abilita o disabilita l'estensione apg_plan_mgmt.	0

Nome del parametro	Descrizione	Default
rds.extensions	Elenco di estensioni fornite da RDS	address_standardizer, address_standardizer_data_us, apg_plan_mgmt, aurora_stat_utils, amcheck, autoinc, aws_commons, aws_ml, aws_s3, aws_lambda, bool_plperl, bloom, btree_gin, btree_gist, citext, cube, dblink, dict_int, dict_xsyn, earthdistance, fuzzystrmatch, hll, hstore, hstore_plperl, insert_username, intagg, intarray, ip4r, isn, jsonb_plperl, lo, log_fdw, ltree, moddatetime, old_snapshot, oracle_fdw, orafce, pgaudit, pgcrypto, pglogical, pgrouting, pgrowlocks, pgstattuple, pgtap, pg_bigm, pg_buffercache, pg_cron, pg_freespacemap, pg_hint_plan, pg_partman, pg_prewarm, pg_proctab, pg_repack, pg_simila

Nome del parametro	Descrizione	Default
		rity, pg_stat_s tatements, pg_trgm, pg_visibility, plcoffee, plls, plperl, plpgsql, plprofiler, pltcl, plv8, postgis, postgis_t iger_geocoder, postgis_raster, postgis_topology, postgres_fdw, prefix, rdkit, rds_tools, refint, sslinfo, tablefunc, tds_fdw, test_parser, tsm_system_rows, tsm_system_time, unaccent, uuid-ossp
rds.force_admin_lo gging_level	Visualizza i messaggi di registro per le azioni dell'utente amministratore di RDS nei database dei clienti.	–
rds.force_autovacu um_logging_level	Visualizza i messaggi dei registri relativi alle operazioni di autovacuum.	WARNING
rds.force_ssl	Forza le connessioni SSL.	0

Nome del parametro	Descrizione	Default
rds.global_db_rpo	(s) Soglia dell'obiettivo del punto di ripristino, in secondi, che blocca i commit dell'utente quando viene violata. **Important** Questo parametro è destinato ai database globali Aurora basati su PostgreSQL. Per un database non globale, lasciare il valore predefinito. Per ulteriori informazioni su questo parametro, consulta the section called "Gestione RPOs (Aurora PostgreSQL)".	–
rds.logical_replication	Abilita la decodifica logica.	0
rds.logically_replicate_unlogged_tables	Le tabelle non registrate vengono replicate logicamente.	1
rds.log_retention_period	Amazon RDS elimina i registri PostgreSQL più vecchi di N minuti.	4320
rds.pg_stat_ramdisk_size	Dimensioni del ramdisk delle statistiche in MB. Un valore diverso da zero configurerà il ramdisk. Questo parametro è disponibile solo in Aurora PostgreSQL 14 e versioni precedenti.	0
rds.rds_superuser_reserved_connections	Imposta il numero di slot di connessione riservati a rds_superusers. Questo parametro è disponibile solo nelle versioni 15 e precedenti. Per ulteriori informazioni, consulta la documentazione di PostgreSQL sulle connessioni riservate.	2
rds.restrict_password_commands	limita i comandi relativi alla password ai membri di rds_password	–

Nome del parametro	Descrizione	Default
rds.superuser_variables	Elenco delle variabili del solo utente con privilegi avanzati per le quali eleviamo le istruzioni di modifica rds_superuser.	session_replication_role
recovery_init_sync_method	Imposta il metodo per sincronizzare la directory dei dati prima del ripristino da arresto anomalo.	syncfs
remove_temp_files_after_crash	Rimuove i file temporanei dopo l'arresto anomalo del back-end	0
restart_after_crash	Reinizializza il server dopo l'arresto anomalo del back-end.	–
row_security	Abilita la sicurezza delle righe.	–
search_path	Imposta l'ordine di ricerca dello schema per i nomi che non sono qualificati come schema.	–
seq_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo sequenziale.	–
session_replication_role	Imposta il comportamento delle sessioni per i trigger e le regole di riscrittura.	–
shared_buffers	(8 kB) Imposta il numero di buffer di memoria condivisa utilizzati dal server.	SUM (DBInstanceClassMemory/12038, -50003)
shared_preload_libraries	Elenca le librerie condivise da precaricare nel server.	pg_stat_statements
ssl	Abilita le connessioni SSL.	1
ssl_ca_file	Posizione del file dell'autorità del server SSL.	/-cert.pem rdsdbdata/rds-metadata/ca

Nome del parametro	Descrizione	Default
ssl_cert_file	Posizione del file del certificato del server SSL.	/rdsdbdata/rds-metadata/server-cert.pem
ssl_ciphers	Imposta l'elenco delle crittografie TLS consentite da utilizzare su connessioni sicure.	–
ssl_crl_dir	Posizione della directory dell'elenco di revoche di certificati SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Posizione del file della chiave privata del server SSL	/rdsdbdata/rds-metadata/server-chiave.pem
ssl_max_protocol_version	Imposta la versione del protocollo SSL/TLS massima consentita	–
ssl_min_protocol_version	Imposta la versione del protocollo SSL/TLS minima consentita	TLSv12.
standard_conforming_strings	Fa sì che le stringhe ... trattino letteralmente le barre rovesciate.	–
statement_timeout	(ms) Imposta la durata massima concessa di ogni istruzione.	–
stats_temp_directory	Scrive i file di statistiche temporanei nella directory specificata.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Imposta il numero di slot di connessione riservati agli utenti con privilegi avanzati.	3
synchronize_seqscans	Abilita le scansioni sequenziali sincronizzate.	–
synchronous_commit	Imposta il livello di sincronizzazione delle transazioni correnti.	on

Nome del parametro	Descrizione	Default
tcp_keepalives_count	Numero massimo di ritrasmissioni keepalive TCP.	–
tcp_keepalives_idle	(s) Tempo tra l'emissione di keepalive TCP.	–
tcp_keepalives_interval	(s) Tempo tra la ritrasmissione di keepalive TCP.	–
temp_buffers	(8 kB) Imposta il numero massimo di buffer temporanei utilizzati da ogni sessione.	–
temp_file_limit	Vincola la quantità totale di spazio su disco in kilobyte che un determinato processo PostgreSQL può utilizzare per i file temporanei, escludendo lo spazio utilizzato per tabelle temporanee esplicite	-1
temp_tablespaces	Imposta i tablespace da usare per le tabelle temporanee e i file di ordinamento.	–
timezone	Imposta il fuso orario per la visualizzazione e l'interpretazione dei timestamp.	UTC
track_activities	Raccoglie informazioni sui comandi di esecuzione.	–
track_activity_query_size	Imposta la dimensione riservata per pg_stat_activity.current_query, in byte.	4096
track_commit_timestamp	Raccoglie l'ora di commit della transazione.	–
track_counts	Raccoglie statistiche sull'attività del database.	–
track_functions	Raccoglie statistiche a livello di funzione sull'attività del database.	pl

Nome del parametro	Descrizione	Default
track_io_timing	Raccoglie statistiche di temporizzazione sull'attività di I/O del database.	1
track_wal_io_timing	Raccoglie statistiche di temporizzazione per l'attività di I/O WAL.	–
transform_null_equals	Tratta expr=NULL come expr È NULL.	–
update_process_title	Aggiorna il titolo del processo per mostrare il comando SQL attivo.	–
vacuum_cost_delay	(ms) Ritardo del costo del vacuum, in millisecondi.	–
vacuum_cost_limit	Quantità del costo del vacuum disponibile prima del napping.	–
vacuum_cost_page_dirty	Costo del vacuum per una pagina sporcata dal vacuum.	–
vacuum_cost_page_hit	Costo del vacuum per una pagina trovata nella cache del buffer.	–
vacuum_cost_page_miss	Costo del vacuum per una non pagina trovata nella cache del buffer.	0
vacuum_defer_cleanup_age	Numero di transazioni in base alle quali il VACUUM e la pulizia A CALDO devono essere posticipati, se presenti.	–
vacuum_failsafe_age	Età in cui il VACUUM deve attivare una misura di sicurezza per evitare un'interruzione del wraparound.	1200000000
vacuum_freeze_min_age	Età minima in cui il VACUUM dovrebbe congelare una riga della tabella.	–

Nome del parametro	Descrizione	Default
vacuum_freeze_table_age	Età in cui il VACUUM dovrebbe eseguire la scansione di un'intera tabella per congelare le tuple.	–
vacuum_multixact_failsafe_age	Età multixact in cui il VACUUM deve attivare una misura di sicurezza per evitare un'interruzione del wraparound.	1200000000
vacuum_multixact_freeze_min_age	Età minima alla quale VACUUM deve congelare un file MultiXactId in una riga della tabella.	–
vacuum_multixact_freeze_table_age	Età del Multixact in cui il VACUUM dovrebbe eseguire la scansione di un'intera tabella per congelare le tuple.	–
wal_buffer	(8 kB) Imposta il numero di buffer della pagina del disco nella memoria condivisa per WAL.	–
wal_receiver_create_temp_slot	Imposta se un ricevitore WAL deve creare uno slot di replica temporaneo se non è configurato uno slot permanente.	0
wal_receiver_status_interval	(s) Imposta l'intervallo massimo tra i report di stato del ricevitore WAL al nodo primario.	–
wal_receiver_timeout	(ms) Imposta il tempo di attesa massimo per la ricezione dei dati dal nodo primario.	30000
wal_sender_timeout	(ms) Imposta il tempo massimo di attesa della replica WAL.	–
work_mem	(kB) Imposta la memoria massima da utilizzare per gli spazi di lavoro delle query.	–
xmlbinary	Imposta come i valori binari devono essere codificati in XML.	–

Nome del parametro	Descrizione	Default
xmloption	Imposta se i dati XML nelle operazioni di analisi e serializzazione implicite devono essere considerati come documenti o frammenti di contenuto.	–

Parametri a livello di istanza Aurora PostgreSQL

Puoi visualizzare i parametri a livello di istanza disponibili per una versione specifica di Aurora PostgreSQL utilizzando la console di AWS gestione, la CLI o l'API Amazon RDS. AWS Per informazioni sulla visualizzazione dei parametri in un gruppo di parametri del database Aurora PostgreSQL nella console RDS, consultare Visualizzazione dei valori dei parametri per un gruppo di parametri DB in Aurora.

Alcuni parametri a livello di istanza non sono disponibili in tutte le versioni e alcuni sono obsoleti. Per informazioni sulla visualizzazione dei parametri di una versione specifica di Aurora PostgreSQL, consultare Visualizzazione del cluster di database Aurora PostgreSQL e dei parametri del database.

Ad esempio, nella tabella seguente sono elencati i parametri applicabili a una specifica istanza database di un cluster database Aurora PostgreSQL. Questo elenco è stato generato eseguendo il comando con for the value. describe-db-parameters AWS CLI `default.aurora-postgresql14--db-parameter-group-name`

Per un elenco dei parametri del cluster database per lo stesso gruppo di parametri DB, consulta Parametri a livello di cluster Aurora PostgreSQL.

Nome del parametro	Descrizione	Default
apg_enable_batch_mode_function_execution	Consente alle funzioni in modalità batch di elaborare serie di righe in una volta.	–
apg_enable_correlated_any_transform	Consente al pianificatore di trasformare il link secondario ANY (sottoquery IN/NOT IN) in JOIN quando possibile.	–
apg_enable_function_migration	Consente al pianificatore di migrare le funzioni scalari idonee alla clausola FROM.	–
apg_enable_not_in_transform	Consente al pianificatore di trasformare la query secondaria NOT IN in ANTI JOIN quando possibile.	–
apg_enable_remove_redundant_inner_joins	Consente al pianificatore di rimuovere i join interni ridondanti.	–

Nome del parametro	Descrizione	Default
apg_enable_semijoin_push_down	Consente l'utilizzo di filtri semijoin per gli hash join.	–
apg_plan_mgmt.capture_plan_baselines	Modalità baseline di acquisizione del piano. manual (manuale): abilita l'acquisizione del piano per qualsiasi istruzione SQL, off: disabilita l'acquisizione del piano, automatic (automatica): abilita l'acquisizione del piano per le istruzioni in pg_stat_statements che soddisfano i criteri di idoneità.	off
apg_plan_mgmt.max_databases	Imposta il numero massimo di database che possono gestire le query utilizzando apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Imposta il numero massimo di piani che possono essere memorizzati nella cache da apg_plan_mgmt.	10000
apg_plan_mgmt.plan_retention_period	Numero massimo di giorni dall'ultimo utilizzo di un piano prima che venga eliminato automaticamente.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Costo totale stimato del piano al di sotto del quale verrà eseguito un piano non approvato.	0
apg_plan_mgmt.use_plan_baselines	Utilizzare solo piani approvati o fissi per le istruzioni gestite.	false
application_name	Imposta il nome dell'applicazione da riportare nelle statistiche e nei registri.	–

Nome del parametro	Descrizione	Default
aurora_compute_plan_id	Monitora i piani di esecuzione delle query per rilevare i piani di esecuzione che contribuiscono al carico corrente del database e per tenere traccia delle statistiche sulle prestazioni dei piani di esecuzione nel tempo. Per ulteriori informazioni, consulta Monitoraggio dei piani di esecuzione delle query per Aurora PostgreSQL.	on
aurora_temp_space_size	(MB) Imposta la dimensione dello spazio allocato per gli oggetti temporanei abilitati alle letture ottimizzate su cluster Aurora ottimizzati per I/O con classi di istanze supportate.	DBInstanceClassMemory/524288
authentication_timeout	(s) Imposta il tempo massimo concesso per completare l'autenticazione del client.	–
auto_explain.log_analyze	Usa EXPLAIN ANALYZE per la registrazione del piano.	–
auto_explain.log_buffers	Registra l'utilizzo dei buffer.	–
auto_explain.log_format	Formato EXPLAIN da utilizzare per la registrazione del piano.	–
auto_explain.log_min_duration	Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrati i piani.	–
auto_explain.log_nested_statements	Registra le istruzioni nidificate.	–
auto_explain.log_timing	Raccogli i dati temporali, non solo il numero di righe.	–
auto_explain.log_triggers	Includi statistiche di attivazione nei piani.	–

Nome del parametro	Descrizione	Default
auto_explain.log_verbose	Usa EXPLAIN VERBOSE per la registrazione del piano.	–
auto_explain.sample_rate	Frazione di query da elaborare.	–
babelfishpg_tds.listen_address	Imposta il nome host o l'indirizzo/gli indirizzi IP su cui ascoltare TDS.	*
babelfishpg_tds.tds_debug_log_level	Imposta il livello di registrazione in TDS, 0 disabilita la registrazione	1
backend_flush_after	(8Kb Numero di pagine dopo le quali le scritture eseguite in precedenza vengono riportate su disco.	–
bytea_output	Imposta il formato di output per byte.	–
check_function_bodies	Controlla i corpi delle funzioni durante CREATE FUNCTION.	–
client_connection_check_interval	Imposta l'intervallo di tempo tra i controlli di disconnessione durante l'esecuzione di query.	–
client_min_messages	Imposta i livelli dei messaggi che vengono inviati al client.	–
config_file	Imposta il file di configurazione principale del server.	/.conf rdsdbdata/config/postgresql
constraint_exclusion	Consente al pianificatore di utilizzare i vincoli per ottimizzare le query.	–
cpu_index_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna voce di indice durante una scansione dell'indice.	–

Nome del parametro	Descrizione	Default
cpu_operator_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna chiamata dell'operatore o della funzione.	–
cpu_tuple_cost	Imposta la stima del pianificatore del costo di elaborazione di ciascuna tupla (riga).	–
cron.database_name	Imposta il database per archiviare le tabelle di metadati pg_cron	postgres
cron.log_run	Registra tutti i processi eseguiti nella tabella job_run_details	on
cron.log_statement	Registra tutte le istruzioni cron prima dell'esecuzione.	off
cron.max_running_jobs	Numero massimo di processi che possono essere eseguiti contemporaneamente.	5
cron.use_background_workers	Abilita i dipendente in background per pg_cron	on
cursor_tuple_fraction	Imposta la stima del pianificatore della frazione delle righe del cursore che verranno recuperate.	–
db_user_namespace	Abilita i nomi utente per database.	–
deadlock_timeout	(ms) Imposta il tempo di attesa su un lock prima di verificare il deadlock.	–
debug_pretty_print	I trattini analizzano e visualizzano le visualizzazioni dell'albero.	–
debug_print_parse	Registra ogni albero di analisi della query.	–
debug_print_plan	Registra ogni programma di esecuzione della query.	–

Nome del parametro	Descrizione	Default
debug_print_rewritten	Registra ogni albero di analisi riscritto della query.	–
default_statistics_target	Imposta la destinazione della statistica predefinita.	–
default_transaction_deferrable	Imposta lo stato differibile predefinito delle nuove transazioni.	–
default_transaction_isolation	Imposta il livello di isolamento della transazione di ogni nuova transazione.	–
default_transaction_read_only	Imposta lo stato di sola lettura predefinito delle nuove transazioni.	–
effective_cache_size	(8 kB) Imposta l'ipotesi dei pianificatori sulla dimensione della cache del disco.	SOMMA (DBInstanceClassMemory/12038, -50003
effective_io_concurrency	Numero di richieste simultanee che possono essere gestite in modo efficace dal sottosistema del disco.	–
enable_async_append	Consente ai pianificatori di utilizzare piani di aggiunta asincroni.	–
enable_bitmapscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione bitmap.	–
enable_gathermerge	Abilita l'utilizzo da parte del pianificatore di piani di unione di raccolta.	–
enable_hashagg	Abilita l'utilizzo da parte del pianificatore di piani di aggregazione hash.	–
enable_hashjoin	Abilita l'utilizzo da parte del pianificatore di piani di unione hash.	–

Nome del parametro	Descrizione	Default
enable_incremental_sort	Consente ai pianificatori di utilizzare fasi incrementali di ordinamento.	–
enable_indexonlyscan	Consente ai pianificatori di utilizzare i piani. index-only-scan	–
enable_indexscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione dell'indice.	–
enable_material	Abilita l'utilizzo da parte del pianificatore della materializzazione.	–
enable_memoize	Abilita l'uso della memorizzazione da parte dei pianificatori	–
enable_mergejoin	Abilita l'utilizzo da parte del pianificatore di piani di unione.	–
enable_nestloop	Abilita l'utilizzo da parte del pianificatore di piani di unione a ciclo nested.	–
enable_parallel_append	Consente ai pianificatori di utilizzare piani di aggiunta paralleli.	–
enable_parallel_hash	Consente ai pianificatori di utilizzare piani di hash paralleli.	–
enable_partition_pruning	Abilita l'eliminazione delle partizioni del tempo di pianificazione e del runtime.	–
enable_partitionwise_aggregate	Consente l'aggregazione e il raggruppamento a livello di partizione.	–
enable_partitionwise_join	Attiva il join a livello di partizione.	–
enable_seqscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione sequenziali.	–

Nome del parametro	Descrizione	Default
enable_sort	Abilita l'utilizzo da parte del pianificatore di passaggi di ordinamento espliciti.	–
enable_tidscan	Abilita l'utilizzo da parte del pianificatore di piani di scansione TID.	–
escape_string_warning	Avvisa circa la perdita di barre rovesciate nelle stringhe letterali ordinarie.	–
exit_on_error	Termina la sessione in caso di errore.	–
force_parallel_mode	Forza l'uso di strutture di query parallele.	–
from_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale le sottoquery non vengono compresse.	–
geqo	Abilita l'ottimizzazione genetica delle query.	–
geqo_effort	GEQO: lo sforzo viene utilizzato per impostare il valore predefinito per altri parametri GEQO.	–
geqo_generations	GEQO: numero di iterazioni dell'algoritmo.	–
geqo_pool_size	GEQO: numero di individui nella popolazione.	–
geqo_seed	GEQO: seme per la selezione casuale del percorso.	–
geqo_selection_bias	GEQO: pressione selettiva all'interno della popolazione.	–
geqo_threshold	Imposta la soglia degli elementi FROM oltre i quali viene utilizzato GEQO.	–
gin_fuzzy_search_limit	Imposta il risultato massimo consentito per la ricerca esatta da GIN.	–

Nome del parametro	Descrizione	Default
gin_pending_list_limit	(kB) Imposta la dimensione massima dell'elenco in sospeso per l'indice GIN.	–
hash_mem_multiplier	Multipli di work_mem da utilizzare per le tabelle hash.	–
hba_file	Imposta il file di configurazione hba dei server.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Consente il feedback da uno standby a caldo all'elemento primario che eviterà conflitti di query.	on
ident_file	Imposta il file di configurazione ident dei server.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Imposta la durata massima concessa di ogni transazione inattiva.	86400000
idle_session_timeout	Termina qualsiasi sessione che è rimasta inattiva (ovvero, in attesa di una richiesta da parte del client), ma che non rientra in una transazione aperta, per un periodo di tempo specificato	–
join_collapse_limit	Imposta la dimensione dell'elenco FROM oltre la quale i costrutti JOIN non vengono appiattiti.	–
lc_messages	Imposta la lingua nella quale vengono visualizzati i messaggi.	–
listen_addresses	Imposta il nome host o l'indirizzo/gli indirizzi IP da ascoltare.	*
lo_compat_privileges	Abilita la modalità di compatibilità con le versioni precedenti per i controlli dei privilegi su oggetti di grandi dimensioni.	0

Nome del parametro	Descrizione	Default
log_connections	Registra ogni connessione riuscita.	–
log_destination	Imposta la destinazione per l'output del registro del server.	stderr
log_directory	Imposta la directory di destinazione per i file di file di log.	/rdsdbdata/log/error
log_disconnections	Registra la fine di una sessione, compresa la durata.	–
log_duration	Registra la durata di ogni istruzione SQL completata.	–
log_error_verbosity	Imposta la verbosità dei messaggi registrati.	–
log_executor_stats	Scrive le statistiche sulla prestazione degli esecutori nel registro del server.	–
log_file_mode	Imposta le autorizzazioni del file per i file di file di log.	0644
log_filename	Imposta il modello del nome del file per i file di registro.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Avvia un processo secondario per acquisire l'output stderr e/o csvlogs nei file di log.	1
log_hostname	Registra il nome host nei registri delle connessioni.	0
logical_decoding_work_mem	(kB) Questa quantità di memoria può essere utilizzata da ogni buffer interno di riordino prima del versamento su disco.	–
log_line_prefix	Controlla le informazioni con prefisso su ciascuna riga di registro.	%t:%r:%u@%d:%p]:

Nome del parametro	Descrizione	Default
log_lock_waits	Registra lunghe attese di lock.	–
log_duration_sample	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verrà registrato un campione di istruzioni. Il campionamento è determinato da log_statement_sample_rate.	–
log_min_duration_statement	(ms) Imposta il tempo minimo di esecuzione al di sopra del quale verranno registrate le istruzioni.	–
log_min_error_statement	Fa sì che tutte le istruzioni che generano un errore pari o superiore a questo livello vengano registrate.	–
log_min_messages	Imposta i livelli dei messaggi che vengono registrati.	–
log_parameter_max_length	(B) Quando si registrano le istruzioni, limita i valori dei parametri registrati ai primi N byte.	–
log_parameter_max_length_on_error	(B) Quando si segnala un errore, limita i valori dei parametri registrati ai primi N byte.	–
log_parser_stats	Scrive le statistiche sulla prestazione del decodificatore nel registro del server.	–
log_planner_stats	Scrive le statistiche sulla prestazione del programmatore nel registro del server.	–
log_replication_commands	Registra ogni comando di replica.	–
log_rotation_age	(min) La rotazione del file di log automatico avverrà dopo N minuti.	60

Nome del parametro	Descrizione	Default
log_rotation_size	(kB) La rotazione del file di log automatico avverrà dopo N kilobyte.	100000
log_statement	Imposta il tipo di istruzioni registrate.	–
log_statement_sample_rate	Frazione di istruzioni che superano log_min_duration_sample da registrare.	–
log_statement_stats	Scrive le statistiche cumulative sulla prestazione nel registro del server.	–
log_temp_files	(kB) Registra l'uso di file temporanei più grandi di questo numero di kilobyte.	–
log_timezone	Imposta il fuso orario da utilizzare nei messaggi di registro.	UTC
log_truncate_on_rotation	Tronca i file di log esistenti con lo stesso nome durante la rotazione del registro.	0
maintenance_io_concurrency	Una variante di effective_io_concurrency utilizzata per i lavori di manutenzione.	1
maintenance_work_mem	(kB) Imposta la memoria massima da utilizzare per le operazioni di manutenzione.	MASSIMO (/6396313 6) *1024,655 36 DBInstance ClassMemory
max_connections	Imposta il numero massimo di connessioni simultanee.	MINIMO (DBInstanceClassMemory/9531392) 5.000
max_files_per_process	Imposta il numero massimo di file aperti in modo simultaneo per ogni processo del server.	–
max_locks_per_transaction	Imposta il numero massimo di lock per transazione.	64

Nome del parametro	Descrizione	Default
max_parallel_maintenance_workers	Imposta il numero massimo di processi paralleli per operazione di manutenzione.	–
max_parallel_workers	Imposta il numero massimo di worker paralleli che possono essere attivi in una sola volta.	MASSIMO ($ VCPU/2,8 DBInstance
max_parallel_workers_per_gather	Imposta il numero massimo di processi paralleli per nodo executor.	–
max_pred_locks_per_page	Imposta il numero massimo di tuple bloccate dal predicato per pagina.	–
max_pred_locks_per_relation	Imposta il numero massimo di pagine e tuple bloccate dal predicato per relazione.	–
max_pred_locks_per_transaction	Imposta il numero massimo di lock del predicato per transazione.	–
max_slot_wal_keep_size	(MB) Gli slot di replica saranno contrassegnati come non riusciti e i segmenti rilasciati per l'eliminazione o il riciclaggio, se questo spazio è occupato dai WAL su disco.	–
max_stack_depth	(kB) Imposta la profondità massima della pila in kilobyte.	6144
max_standby_streaming_delay	(ms) Imposta il ritardo massimo prima di annullare le query quando un server con standby a caldo elabora i dati WAL in streaming .	14000
max_worker_processes	Imposta il numero massimo di processi dipendente simultanei.	MASSIMO ($ DBInstance VCPU*2,8
min_dynamic_shared_memory	(MB) Quantità di memoria condivisa dinamica riservata all'avvio.	–

Nome del parametro	Descrizione	Default
min_parallel_index_scan_size	(8 kB) Imposta la quantità minima di dati di indicizzazione per una scansione parallela.	–
min_parallel_table_scan_size	(8 kB) Imposta la quantità minima di dati della tabella per una scansione parallela.	–
old_snapshot_threshold	(min) Tempo prima che uno snapshot sia troppo vecchio per leggere le pagine modificate dopo l'acquisizione dello snapshot.	–
parallel_leader_participation	Controlla se Gather e Gather Merge eseguono anche piani secondari.	–
parallel_setup_cost	Imposta la stima dei pianificatori del costo di avvio dei processi dei dipendenti per la query parallela.	–
parallel_tuple_cost	Imposta la stima del pianificatore del costo del passaggio di ciascuna tupla (riga) dal back-end dipendente a quello principale.	–
pgaudit.log	Specifica quali classi di istruzioni verranno registrate per registrazione di verifica della sessione.	–
pgaudit.log_catalog	Specifica che la registrazione delle sessioni deve essere abilitata nel caso in cui tutte le relazioni in un'istruzione siano in pg_catalog.	–
pgaudit.log_level	Specifica il livello di log che verrà utilizzato per le voci di registro.	–
pgaudit.log_parameter	Specifica che la registrazione di verifica deve includere i parametri passati con l'istruzione.	–

Nome del parametro	Descrizione	Default
pgaudit.log_relation	Specifica se la registrazione di verifica della sessione deve creare una voce di registro separata per ogni relazione (TABLE, VIEW, ecc.) a cui fa riferimento in un'istruzione SELECT o DML.	–
pgaudit.log_statement_once	Specifica se la registrazione includerà il testo dell'istruzione e i parametri con la prima voce di registro per una combinazione istruzione/istruzione secondaria o con ogni voce.	–
pgaudit.role	Specifica il ruolo principale da utilizzare per la registrazione di verifica degli oggetti.	–
pg_bigm.enable_recheck	Specifica se eseguire Recheck, che è un processo interno di ricerca full-text.	on
pg_bigm.gin_key_limit	Specifica il numero massimo di 2-grammi della parola chiave di ricerca da utilizzare per la ricerca full-text.	0
pg_bigm.last_update	Riporta l'ultima data aggiornata del modulo pg_bigm.	2013.11.22
pg_bigm.similarity_limit	Specifica la soglia minima utilizzata dalla ricerca di somiglianza.	0.3
pg_hint_plan.debug_print	Registra i risultati dell'analisi dei suggerimenti.	–
pg_hint_plan.enable_hint	Forza il pianificatore a utilizzare i piani specificati nel commento del suggerimento precedente alla query.	–
pg_hint_plan.enable_hint_table	Forza il pianificatore a non ottenere indicazioni utilizzando le ricerche nella tabella.	–

Nome del parametro	Descrizione	Default
pg_hint_plan.message_level	Livello di messaggio dei messaggi di debug.	–
pg_hint_plan.parse_messages	Livello di messaggio degli errori di analisi.	–
pglogical.batch_inserts	Inserimenti batch, se possibile	–
pglogical.conflict_log_level	Imposta il livello di registro utilizzato per la registrazione dei conflitti risolti.	–
pglogical.conflict_resolution	Imposta il metodo utilizzato per la risoluzione dei conflitti per i conflitti risolvibili.	–
pglogical.extra_connection_options	opzioni di connessione da aggiungere a tutte le connessioni dei nodi peer	–
pglogical.synchronous_commit	Valore di commit sincrono specifico pglogical	–
pglogical.use_spi	Usa SPI invece dell'API di basso livello per applicare le modifiche	–
pg_similarity.block_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.block_threshold	Imposta la soglia utilizzata dalla funzione di similarità dei blocchi.	–
pg_similarity.block_tokenizer	Imposta il tokenizzatore per la funzione di similarità dei blocchi.	–
pg_similarity.cosine_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.cosine_threshold	Imposta la soglia utilizzata dalla funzione di similarità del coseno.	–

Nome del parametro	Descrizione	Default
pg_similarity.cosine_tokenizer	Imposta il tokenizzatore per la funzione di similarità del coseno.	–
pg_similarity.dice_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.dice_threshold	Imposta la soglia utilizzata dalla misura di similarità di Dice.	–
pg_similarity.dice_tokenizer	Imposta il tokenizzatore per la misura di similarità di Dice.	–
pg_similarity.euclidean_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.euclidean_threshold	Imposta la soglia utilizzata dalla misura di similarità euclidea.	–
pg_similarity.euclidean_tokenizer	Imposta il tokenizzatore per la misura di similarità euclidea.	–
pg_similarity.hamming_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.hamming_threshold	Imposta la soglia utilizzata dal parametro di similarità dei blocchi.	–
pg_similarity.jaccard_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaccard_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaccard.	–
pg_similarity.jaccard_tokenizer	Imposta il tokenizzatore per la misura di similarità di Jaccard.	–
pg_similarity.jaro_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.jaro_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jaro.	–
pg_similarity.jaro winkler_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.jaro winkler_threshold	Imposta la soglia utilizzata dalla misura di similarità di Jarowinkler.	–
pg_similarity.leve nshtein_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.leve nshtein_threshold	Imposta la soglia utilizzata dalla misura di similarità di Levenshtein.	–
pg_similarity.matc hing_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.matc hing_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.matc hing_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di corrispondenza.	–
pg_similarity.mong eelkan_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.mong eelkan_threshold	Imposta la soglia utilizzata dalla misura di similarità di Monge-Elkan.	–
pg_similarity.mong eelkan_tokenizer	Imposta il tokenizzatore per la misura di similarità di Monge-Elkan.	–
pg_similarity.nw_g ap_penalty	Imposta la penalità del gap utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.nw_i s_normalized	Imposta se il valore del risultato è normalizzato o meno.	–

Nome del parametro	Descrizione	Default
pg_similarity.nw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.overlap_threshold	Imposta la soglia utilizzata dalla misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.overlap_tokenizer	Imposta il tokenizzatore per la misura di similarità del coefficiente di sovrapposizione.	–
pg_similarity.qgram_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.qgram_threshold	Imposta la soglia utilizzata dalla misura di similitudine Q-Gram.	–
pg_similarity.qgram_tokenizer	Imposta il tokenizzatore per la misura Q-Gram.	–
pg_similarity.swg_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarità.swg_threshold	Imposta la soglia utilizzata dalla misura di somiglianza Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Imposta se il valore del risultato è normalizzato o meno.	–
pg_similarity.sw_threshold	Imposta la soglia utilizzata dalla misura di similitudine di Smith-Waterman.	–
pg_stat_statements.max	Imposta il numero massimo di istruzioni monitorate da pg_stat_statements.	–
pg_stat_statements.save	Salva le statistiche pg_stat_statements attraverso le interruzioni del server.	–

Nome del parametro	Descrizione	Default
pg_stat_statements.track	Seleziona le istruzioni che vengono monitorate da pg_stat_statements.	–
pg_stat_statements.track_planning	Seleziona se la durata della pianificazione è monitorata da pg_stat_statements.	–
pg_stat_statements.track_utility	Seleziona se i comandi di utilità sono monitorati da pg_stat_statements.	–
postgis.gdal_enabled_drivers	Abilita o disabilita i driver GDAL utilizzati con PostGIS in Postgres 9.3.5 e versioni successive.	ENABLE_ALL
quote_all_identifiers	Aggiungi le virgolette a tutti gli identificatori quando si generano i frammenti SQL.	–
random_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo non sequenziale.	–
rds.enable_memory_management	Migliora le funzionalità di gestione della memoria in Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 e versioni successive, prevenendo problemi di stabilità e riavvii del database causati da insufficiente memoria libera. Per ulteriori informazioni, consulta Migliore gestione della memoria in Aurora PostgreSQL.	True
rds.force_admin_logging_level	Visualizza i messaggi di registro per le azioni dell'utente amministratore di RDS nei database dei clienti.	–
rds.log_retention_period	Amazon RDS elimina i registri PostgreSQL più vecchi di N minuti.	4320

Nome del parametro	Descrizione	Default
rds.memory_allocation_guard	Migliora le funzionalità di gestione della memoria in Aurora PostgreSQL 11.21, 12.16, 13.12, 14.9, 15.4 e versioni precedenti, prevenendo problemi di stabilità e riavvii del database causati da insufficiente memoria libera. Per ulteriori informazioni, consulta Migliore gestione della memoria in Aurora PostgreSQL.	False
rds.pg_stat_ramdisk_size	Dimensioni del ramdisk delle statistiche in MB. Un valore diverso da zero configurerà il ramdisk.	0
rds.rds_superuser_reserved_connections	Imposta il numero di slot di connessione riservati a rds_superusers. Questo parametro è disponibile solo nelle versioni 15 e precedenti. Per ulteriori informazioni, consulta la documentazione di PostgreSQL sulle connessioni riservate.	2
rds.superuser_variables	Elenco delle variabili del solo utente con privilegi avanzati per le quali eleviamo le istruzioni di modifica rds_superuser.	session_replication_role
remove_temp_files_after_crash	Rimuove i file temporanei dopo l'arresto anomalo del back-end	0
restart_after_crash	Reinizializza il server dopo l'arresto anomalo del back-end.	–
row_security	Abilita la sicurezza delle righe.	–
search_path	Imposta l'ordine di ricerca dello schema per i nomi che non sono qualificati come schema.	–

Nome del parametro	Descrizione	Default
seq_page_cost	Imposta la stima del pianificatore del costo di una pagina del disco recuperata in modo sequenziale.	–
session_replicatio n_role	Imposta il comportamento delle sessioni per i trigger e le regole di riscrittura.	–
shared_buffers	(8 kB) Imposta il numero di buffer di memoria condivisa utilizzati dal server.	SUM (DBInstan ceClassMemory/1203 8, -50003)
shared_preload_lib raries	Elenca le librerie condivise da precaricare nel server.	pg_stat_statements
ssl_ca_file	Posizione del file dell'autorità del server SSL.	/-cert.pem rdsdbdata/ rds-metadata/ca
ssl_cert_file	Posizione del file del certificato del server SSL.	/rdsdbdata/rds-met adata/server-cert.pem
ssl_crl_dir	Posizione della directory dell'elenco di revoche di certificati SSL.	/rdsdbdata/rds-met adata/ssl_crl_dir/
ssl_key_file	Posizione del file della chiave privata del server SSL	/rdsdbdata/rds-met adata/server-chiav e.pem
standard_conformin g_strings	Fa sì che le stringhe ... trattino letteralmente le barre rovesciate.	–
statement_timeout	(ms) Imposta la durata massima concessa di ogni istruzione.	–
stats_temp_directory	Scrive i file di statistiche temporanei nella directory specificata.	/rdsdbdata/db/pg_s tat_tmp

Nome del parametro	Descrizione	Default
superuser_reserved_connections	Imposta il numero di slot di connessione riservati agli utenti con privilegi avanzati.	3
synchronize_seqscans	Abilita le scansioni sequenziali sincronizzate.	–
tcp_keepalives_count	Numero massimo di ritrasmissioni keepalive TCP.	–
tcp_keepalives_idle	(s) Tempo tra l'emissione di keepalive TCP.	–
tcp_keepalives_interval	(s) Tempo tra la ritrasmissione di keepalive TCP.	–
temp_buffers	(8 kB) Imposta il numero massimo di buffer temporanei utilizzati da ogni sessione.	–
temp_file_limit	Vincola la quantità totale di spazio su disco in kilobyte che un determinato processo PostgreSQL può utilizzare per i file temporanei, escludendo lo spazio utilizzato per tabelle temporanee esplicite	-1
temp_tablespaces	Imposta i tablespace da usare per le tabelle temporanee e i file di ordinamento.	–
track_activities	Raccoglie informazioni sui comandi di esecuzione.	–
track_activity_query_size	Imposta la dimensione riservata per pg_stat_activity.current_query, in byte.	4096
track_counts	Raccoglie statistiche sull'attività del database.	–
track_functions	Raccoglie statistiche a livello di funzione sull'attività del database.	pl

Nome del parametro	Descrizione	Default
track_io_timing	Raccoglie statistiche di temporizzazione sull'attività di I/O del database.	1
transform_–_equals	Treats expr=– as expr IS –.	–
update_process_title	Aggiorna il titolo del processo per mostrare il comando SQL attivo.	–
wal_receiver_status_interval	(s) Imposta l'intervallo massimo tra i report di stato del ricevitore WAL al nodo primario.	–
work_mem	(kB) Imposta la memoria massima da utilizzare per gli spazi di lavoro delle query.	–
xmlbinary	Imposta come i valori binari devono essere codificati in XML.	–
xmloption	Imposta se i dati XML nelle operazioni di analisi e serializzazione implicite devono essere considerati come documenti o frammenti di contenuto.	–

Eventi di attesa Amazon Aurora PostgreSQL

Di seguito sono riportati alcuni tra gli eventi di attesa più frequenti di Aurora PostgreSQL. Per ulteriori informazioni sugli eventi di attesa e l'ottimizzazione del cluster database Aurora PostgreSQL, consulta Sintonizzazione degli eventi di attesa per Aurora PostgreSQL.

Attività: ArchiverMain

Il processo di archiviazione è in attesa di attività.

Attività: AutoVacuumMain

Il processo del programma di avvio di autovacuum è in attesa di attività.

Attività: BgWriterHibernate

Il processo di scrittura in background è in ibernazione in attesa dell'attività.

Attività: BgWriterMain

Il processo di scrittura in background è in attesa di attività.

Attività: CheckpointerMain

Il processo di checkpointer è in attesa di attività.

Attività: LogicalApplyMain

Il processo di applicazione della replica logica è in attesa di attività.

Attività: LogicalLauncherMain

Il processo di avvio della replica logica è in attesa di attività.

Attività: PgStatMain

Il processo di raccolta statistiche è in attesa di attività.

Attività: RecoveryWalAll

Un processo è in attesa del WAL (write-ahead log) da un flusso al momento del ripristino.

Attività: RecoveryWalStream

Il processo di avvio è in attesa che il WAL (write-ahead log) arrivi durante il ripristino dello streaming.

Attività: SysLoggerMain

Il processo syslogger è in attesa di attività.

Attività: WalReceiverMain

Il processo WAL (write-ahead log) del destinatario è in attesa di attività.

Attività: WalSenderMain

Il processo WAL (write-ahead log) del mittente è in attesa di attività.

Attività: WalWriterMain

Il processo di scrittura WAL (write-ahead log) è in attesa di attività.

BufferPin:BufferPin

Un processo è in attesa di acquisire un pin esclusivo su un buffer.

Client: GSSOpen Server

Un processo è in attesa di leggere i dati dal client nel tentativo di stabilire una sessione GSSAPI (Generic Security Service Application Program Interface).

Cliente: ClientRead

Un processo di backend è in attesa di ricevere dati da un client PostgreSQL. Per ulteriori informazioni, consulta Cliente: ClientRead.

Cliente: ClientWrite

Un processo di backend è in attesa di inviare più dati a un client PostgreSQL. Per ulteriori informazioni, consulta Cliente: ClientWrite.

Cliente: Lib PQWal ReceiverConnect

Un processo è in attesa nel destinatario WAL (write-ahead log) per stabilire la connessione al server remoto.

Cliente: Lib PQWal ReceiverReceive

Un processo è in attesa nel destinatario WAL (write-ahead log) per ricevere i dati dal server remoto.

Cliente: Server SSLOpen

Un processo è in attesa di Secure Sockets Layer (SSL) durante il tentativo di connessione.

Cliente: WalReceiverWaitStart

Un processo è in attesa che il processo di avvio invii i dati iniziali per la replica in streaming.

Cliente: WalSenderWaitFor WAL

Un processo è in attesa che il WAL (write-ahead log) venga svuotato nel processo mittente WAL.

Cliente: WalSenderWriteData

Un processo è in attesa di qualsiasi attività durante l'elaborazione delle risposte dal destinatario WAL (write-ahead log) nel processo mittente WAL.

CPU

Un processo di backend è attivo o è in attesa della CPU. Per ulteriori informazioni, consulta CPU.

Extension:extension

Un processo di backend è in attesa di una condizione definita da un'estensione o da un modulo.

IO: AuroraEnhancedLogical WALRead

Un processo di backend sta recuperando i record di registro dal volume Change Data Capture (CDC).

IO: AuroraOptimizedReadsCacheRead

Un processo è in attesa di una lettura dalla cache a più livelli di Letture ottimizzate perché la pagina non è disponibile nella memoria condivisa.

IO: AuroraOptimizedReads CacheSegmentTronca

Un processo è in attesa che un file di segmento della cache a più livelli di Letture ottimizzate venga troncato.

IO: AuroraOptimizedReadsCacheWrite

Il processo di scrittura in background è in attesa di scrivere nella cache a più livelli di Letture ottimizzate.

IO: AuroraStorageLogAllocate

Una sessione sta allocando i metadati e si sta preparando per la scrittura nel registro delle transazioni.

IO: BufFileRead

Quando le operazioni richiedono più memoria della quantità definita dai parametri di memoria di lavoro, il motore crea file temporanei su disco. Questo evento di attesa si verifica quando le operazioni vengono lette dai file temporanei. Per ulteriori informazioni, consulta IO:buffileRead e IO:buffileWrite.

IO: BufFileWrite

Quando le operazioni richiedono più memoria della quantità definita dai parametri di memoria di lavoro, il motore crea file temporanei su disco. Questo evento di attesa si verifica quando le operazioni scrivono nei file temporanei. Per ulteriori informazioni, consulta IO:buffileRead e IO:buffileWrite.

IO: ControlFileRead

Un processo è in attesa di una lettura dal file `pg_control`.

IO: ControlFileSync

Un processo è in attesa del file `pg_control` per raggiungere un'archiviazione durevole.

IO: ControlFileSyncUpdate

Un processo è in attesa di un aggiornamento del file `pg_control` per raggiungere un’archiviazione durevole.

IO: ControlFileWrite

Un processo è in attesa di una scrittura sul file `pg_control`.

IO: ControlFileWriteUpdate

Un processo è in attesa di una scrittura per aggiornare il file `pg_control`.

IO: CopyFileRead

Un processo è in attesa di una lettura durante un'operazione di copia dei file.

IO: CopyFileWrite

Un processo è in attesa di una scrittura durante un'operazione di copia dei file.

IO: DataFileExtend

Un processo è in attesa dell'estensione di un file di dati di relazione.

IO: DataFileFlush

Un processo è in attesa che un file di dati di relazione raggiunga un’archiviazione durevole.

IO: DataFileImmediateSync

Un processo è in attesa di una sincronizzazione immediata di un file di dati di relazione a un’archiviazione durevole.

IO: DataFilePrefetch

Un processo è in attesa di un prefetch asincrono da un file di dati di relazione.

IO: DataFileSync

Un processo è in attesa di modifiche a un file di dati di relazione per raggiungere un’archiviazione durevole.

IO: DataFileRead

Un processo di back-end ha cercato di trovare una pagina nei buffer condivisi, non l'ha trovata e quindi l'ha letta dall’archiviazione. Per ulteriori informazioni, consulta IO: DataFileRead.

IO: DataFileTruncate

Un processo è in attesa che un file di dati di relazione venga troncato.

IO: DataFileWrite

Un processo è in attesa di una scrittura su un file di dati di relazione.

IO: DSMFill ZeroWrite

Un processo è in attesa di scrivere zero byte su un file dinamico di backup della memoria condivisa.

IO: LockFileAddToDataDirRead

Un processo è in attesa di una lettura durante l'aggiunta di una riga al file da bloccare della directory dati.

IO: LockFileAddToDataDirSync

Un processo è in attesa che i dati raggiungano un'archiviazione durevole durante l'aggiunta di una riga al file da bloccare della directory dati.

IO: LockFileAddToDataDirWrite

Un processo è in attesa di scrittura durante l'aggiunta di una riga al file da bloccare della directory dati.

IO: LockFileCreateRead

Un processo è in attesa di lettura durante la creazione del file da bloccare della directory dati.

IO: LockFileCreateSync

Un processo è in attesa che i dati raggiungano un'archiviazione durevole durante la creazione del file da bloccare della directory dati.

IO: LockFileCreateWrite

Un processo è in attesa di scrittura durante la creazione del file da bloccare della directory dati.

IO: LockFileReCheckDataDirRead

Un processo è in attesa di una lettura durante il ricontrollo del file da bloccare della directory dati.

IO: LogicalRewriteCheckpointSync

Un processo è in attesa che le mappature di riscrittura logica raggiungano un'archiviazione durevole durante un checkpoint.

IO: LogicalRewriteMappingSync

Un processo è in attesa che i dati di mappatura raggiungano un'archiviazione durevole durante una riscrittura logica.

IO: LogicalRewriteMappingWrite

Un processo è in attesa di una scrittura dei dati di mappatura durante una riscrittura logica.

IO: LogicalRewriteSync

Un processo è in attesa che le mappature di riscrittura logica raggiungano un'archiviazione durevole.

IO: LogicalRewriteTruncate

Un processo è in attesa del troncamento dei dati di mappatura durante una riscrittura logica.

IO: LogicalRewriteWrite

Un processo è in attesa di una scrittura di mappature di riscrittura logica.

IO: RelationMapRead

Un processo è in attesa di una lettura del file di mappa della relazione.

IO: RelationMapSync

Un processo è in attesa che il file di mappa della relazione raggiunga un'archiviazione durevole.

IO: RelationMapWrite

Un processo è in attesa di una scrittura sul file di mappatura della relazione.

IO: ReorderBufferRead

Un processo è in attesa di una lettura durante la gestione del buffer da riordinare.

IO: ReorderBufferWrite

Un processo è in attesa di una scrittura durante la gestione del buffer da riordinare.

IO: ReorderLogicalMappingRead

Un processo è in attesa di una lettura di una mappatura logica durante la gestione del buffer da riordinare.

IO: ReplicationSlotRead

Un processo è in attesa di una lettura da un file di controllo dello slot di replica.

IO: ReplicationSlotRestoreSync

Un processo è in attesa che un file di controllo dello slot di replica raggiunga un'archiviazione durevole durante il ripristino in memoria.

IO: ReplicationSlotSync

Un processo è in attesa che un file di controllo dello slot di replica raggiunga un'archiviazione durevole.

IO: ReplicationSlotWrite

Un processo è in attesa di una scrittura su un file di controllo dello slot di replica.

IO: SLRUFlush sincronizzazione

Un processo è in attesa che i dati semplici utilizzati meno di recente (SLRU) raggiungano un'archiviazione durevole durante un checkpoint o uno spegnimento del database.

IO: SLRURead

Un processo è in attesa di una lettura di una pagina semplice utilizzata meno di recente (SLRU).

IO: SLRUSync

Un processo è in attesa che i dati semplici utilizzati meno di recente (SLRU) raggiungano un'archiviazione durevole dopo la scrittura di una pagina.

IO: SLRUWrite

Un processo è in attesa della scrittura di una pagina semplice utilizzata meno di recente (SLRU).

IO: SnapbuildRead

Un processo è in attesa di una lettura di uno snapshot del catalogo storico serializzato.

IO: SnapbuildSync

Un processo è in attesa che uno snapshot del catalogo storico serializzato raggiunga un'archiviazione durevole.

IO: SnapbuildWrite

Un processo è in attesa della scrittura di uno snapshot del catalogo storico serializzato.

IO: TimelineHistoryFileSync

Un processo è in attesa che un file della cronologia della timeline ricevuto tramite la replica in streaming raggiunga un'archiviazione durevole.

IO: TimelineHistoryFileWrite

Un processo è in attesa di una scrittura di un file di cronologia della timeline ricevuto tramite la replica in streaming.

IO: TimelineHistoryRead

Un processo è in attesa di una lettura di un file cronologico della timeline.

IO: TimelineHistorySync

Un processo è in attesa che un file cronologico della timeline appena creato raggiunga un'archiviazione durevole.

IO: TimelineHistoryWrite

Un processo è in attesa di una scrittura di un file di cronologia della timeline appena creato.

IO: TwophaseFileRead

Un processo è in attesa di una lettura di un file di stato in due fasi.

IO: TwophaseFileSync

Un processo è in attesa che un file di stato in due fasi raggiunga un'archiviazione durevole.

IO: TwophaseFileWrite

Un processo è in attesa di scrittura di un file di stato in due fasi.

IO: WALBootstrap sincronizzazione

Un processo è in attesa che il WAL (write-ahead log) raggiunga un'archiviazione durevole durante il processo di bootstrap.

IO: WALBootstrap Scrivi

Un processo è in attesa di scrittura di una pagina WAL (write-ahead log) durante il processo di bootstrap.

IO: WALCopy Leggi

Un processo è in attesa di una lettura durante la creazione di un nuovo segmento WAL (write-ahead log) tramite copia di uno esistente.

IO: WALCopy sincronizzazione

Un processo è in attesa di un nuovo segmento WAL (write-ahead log) creato tramite copia di uno esistente per raggiungere un'archiviazione durevole.

IO: WALCopy Scrivi

Un processo è in attesa di una scrittura quando si crea un nuovo segmento WAL (write-ahead log) tramite copia di uno esistente.

IO: WALInit sincronizzazione

Un processo è in attesa che un file WAL (write-ahead log) di recente inizializzazione raggiunga l'archiviazione durevole.

IO: WALInit Scrivi

Un processo è in attesa di una scrittura durante l'inizializzazione di un nuovo file WAL (write-ahead log).

IO: WALRead

Un processo è in attesa di una lettura da un file WAL (write-ahead log).

IO: WALSender TimelineHistoryRead

Un processo è in attesa di una lettura da un file cronologico della timeline durante un comando della timeline del mittente WAL (write-ahead log).

IO: WALSync

Un processo è in attesa che un file WAL (write-ahead log) raggiunga un'archiviazione durevole.

IO: WALSync MethodAssign

Un processo è in attesa che i dati raggiungano un'archiviazione durevole durante l'assegnazione di un nuovo metodo di sincronizzazione WAL (write-ahead log).

IO: WALWrite

Un processo è in attesa di una scrittura su un file WAL (write-ahead log).

IO: XactSync

Un processo di backend è in attesa che il sottosistema archiviazione Aurora riconosca il commit di una transazione regolare o il commit o il rollback di una transazione preparata. Per ulteriori informazioni, consulta IO: XactSync.

IPC: AuroraLogicalSchemaUpdate

Due processi di backend stanno tentando di inserire la stessa voce nella cache dello schema. Un processo continuerà mentre l'altro attende il completamento.

IPC: AuroraOptimizedReadsCacheWriteStop

Un processo è in attesa che lo scrittore in background interrompa la scrittura nella cache a più livelli di Optimized Reads.

IPC: BackupWaitWalArchive

Un processo è in attesa dei file WAL (write-ahead log) necessari per l'archiviazione corretta di un backup.

IPC: BgWorkerShutdown

Un processo è in attesa dell'arresto del lavoro in background.

IPC: BgWorkerStartup

Un processo è in attesa dell'avvio di un lavoro in background.

IPC: BtreePage

Un processo è in attesa che il numero di pagina necessario per continuare la scansione parallela del B-tree diventi disponibile.

IPC: CheckpointDone

Un processo è in attesa del completamento di un checkpoint.

IPC: CheckpointStart

Un processo è in attesa dell'avvio di un checkpoint.

IPC: ClogGroupUpdate

Un processo è in attesa che il leader del gruppo aggiorni lo stato della transazione al termine di una transazione.

IPC: DamRecordTxAck

Un processo di backend ha generato un evento di flussi di attività del database ed è in attesa che l'evento diventi durevole. Per ulteriori informazioni, consulta IPC:DamRecordTxAck.

IPC: ExecuteGather

Un processo è in attesa di attività da un processo figlio durante l'esecuzione di un nodo del piano Gather.

IPC: Hash/Batch/Allocating

Un processo è in attesa dell’allocazione di una tabella hash da parte di un partecipante all’hash parallelo eletto.

IPC: Hash/Batch/Electing

Un processo sta eleggendo un partecipante all’hash parallelo per allocare una tabella hash.

IPC: Hash/Batch/Loading

Un processo è in attesa che altri partecipanti all’hash parallelo finiscano il caricamento di una tabella hash.

IPC: Hash/Build/Allocating

Un processo è in attesa dell’allocazione della tabella hash iniziale da parte di un partecipante all’hash parallelo eletto.

IPC: Hash/Build/Electing

Un processo sta eleggendo un partecipante all’hash parallelo per allocare la tabella hash iniziale.

IPC: Hash/Build/HashingInner

Un processo è in attesa che altri partecipanti all’hash parallelo finiscano di eseguire l'hashing della relazione interna.

IPC: Hash/Build/HashingOuter

Un processo è in attesa che altri partecipanti all’hash parallelo finiscano il partizionamento della relazione esterna.

IPC: Hash/GrowBatches/Allocating

Un processo è in attesa dell’allocazione di più batch da parte di un partecipante all’hash parallelo eletto.

IPC: Hash/GrowBatches/Deciding

Un processo sta eleggendo un partecipante all'hash parallelo per decidere sull’aumento futuro dei batch.

IPC: Hash/GrowBatches/Electing

Un processo sta eleggendo un partecipante all’hash parallelo per l’allocazione di più batch.

IPC: Hash/GrowBatches/Finishing

Un processo è in attesa che un partecipante all'hash parallelo eletto decida sulla crescita futura dei batch.

IPC: Hash/GrowBatches/Repartitioning

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il ripartizionamento.

IPC: Hash/GrowBuckets/Allocating

Un processo è in attesa che un partecipante all'hash parallelo eletto finisca l'allocazione di più bucket.

IPC: Hash/GrowBuckets/Electing

Un processo sta eleggendo un partecipante all'hash parallelo per l'allocazione di più bucket.

IPC: Hash/GrowBuckets/Reinserting

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano di inserire tuple in nuovi bucket.

IPC: HashBatchAllocate

Un processo è in attesa dell'allocazione di una tabella hash da parte di un partecipante all'hash parallelo eletto.

IPC: HashBatchElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare una tabella hash.

IPC: HashBatchLoad

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il caricamento di una tabella hash.

IPC: HashBuildAllocate

Un processo è in attesa dell'allocazione della tabella hash iniziale da parte di un partecipante all'hash parallelo eletto.

IPC: HashBuildElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare la tabella hash iniziale.

IPC: HashBuildHashInner

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano di eseguire l'hashing della relazione interna.

IPC:» HashBuildHashOuter

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il partizionamento della relazione esterna.

IPC: HashGrowBatchesAllocate

Un processo è in attesa dell'allocazione di più batch da parte di un partecipante all'hash parallelo eletto.

IPC:» HashGrowBatchesDecide

Un processo è in attesa di eleggere un partecipante all'hash parallelo per decidere sulla crescita futura dei batch.

IPC: HashGrowBatchesElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare più batch.

IPC: HashGrowBatchesFinish

Un processo è in attesa che un partecipante all'hash parallelo eletto decida sulla crescita futura dei batch.

IPC: HashGrowBatchesRepartition

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano il ripartizionamento.

IPC: HashGrowBucketsAllocate

Un processo è in attesa che un partecipante all'hash parallelo eletto finisca l'allocazione di più bucket.

IPC: HashGrowBucketsElect

Un processo è in attesa di eleggere un partecipante all'hash parallelo per allocare più bucket.

IPC: HashGrowBucketsReinsert

Un processo è in attesa che altri partecipanti all'hash parallelo finiscano di inserire tuple in nuovi bucket.

IPC: LogicalSyncData

Un processo è in attesa che un server remoto di replica logica invii dati per la sincronizzazione iniziale della tabella.

IPC: LogicalSyncStateChange

Un processo è in attesa che un server remoto di replica logica modifichi lo stato.

IPC: MessageQueueInternal

Un processo è in attesa che un altro processo sia allegato a una coda di messaggi condivisa.

IPC: MessageQueuePutMessage

Un processo è in attesa di scrivere un messaggio di protocollo in una coda di messaggi condivisa.

IPC: MessageQueueReceive

Un processo è in attesa di ricevere byte da una coda di messaggi condivisa.

IPC: MessageQueueSend

Un processo è in attesa di inviare byte a una coda di messaggi condivisa.

IPC: ParallelBitmapScan

Un processo è in attesa di inizializzazione da parte di una scansione bitmap parallela.

IPC: ParallelCreateIndexScan

Un processo è in attesa che i lavori CREATE INDEX (CREA INDICE) paralleli finiscano una scansione heap.

IPC: ParallelFinish

Un processo è in attesa che i lavori paralleli finiscano l'elaborazione.

IPC: ProcArrayGroupUpdate

Un processo è in attesa che il capogruppo cancelli l'ID della transazione al termine della transazione.

IPC: ProcSignalBarrier

Un processo è in attesa che un evento barriera venga elaborato da tutti i backend.

IPC:Promote

Un processo è in attesa di una promozione in standby.

IPC: RecoveryConflictSnapshot

Un processo è in attesa della risoluzione dei conflitti di ripristino per una pulizia vacuum.

IPC: RecoveryConflictTablespace

Un processo è in attesa della risoluzione dei conflitti di ripristino per l'eliminazione di uno spazio di tabella.

IPC: RecoveryPause

Un processo è in attesa di riavvio del ripristino.

IPC: ReplicationOriginDrop

Un processo è in attesa che un'origine di replica diventi inattiva in modo che possa essere eliminata.

IPC: ReplicationSlotDrop

Un processo è in attesa che uno slot di replica diventi inattivo in modo che possa essere eliminato.

IPC: SafeSnapshot

Un processo è in attesa di ottenere uno snapshot valido per una transazione READ ONLY DEFERRABLE (SOLA LETTURA DIFFERIBILE).

IPC: SyncRep

Un processo è in attesa di conferma da un server remoto durante la replica sincrona.

IPC: XactGroupUpdate

Un processo è in attesa che il capogruppo aggiorni lo stato della transazione al termine della transazione.

Lock:advisory

Un processo di back-end è in attesa del blocco di consulenza che ha richiesto. Per ulteriori informazioni, consulta Lock:advisory.

Lock:extend

Un processo di backend è in attesa del rilascio di un blocco in modo che possa estendere una relazione. Questo blocco è necessario perché solo un processo di back-end può estendere una relazione alla volta. Per ulteriori informazioni, consulta Lock:extend.

Lock:frozenid

Un processo è in attesa di aggiornare `pg_database.datfrozenxid` e `pg_database.datminmxid`.

Lock:object

Un processo è in attesa di ottenere un blocco su un oggetto di database non relazionale.

Lock:page

Un processo è in attesa di ottenere un blocco su una pagina di una relazione.

Lock:Relation

Un processo di backend è in attesa di acquisire un blocco su una relazione bloccata da un'altra transazione. Per ulteriori informazioni, consulta Lock:Relation.

Lock:spectoken

Un processo è in attesa di ottenere un blocco di inserimento speculativo.

Token Lock:speculative

Un processo è in attesa di acquisire un blocco di inserimento speculativo.

Lock:transactionid

Una transazione è in attesa di un blocco a livello di riga. Per ulteriori informazioni, consulta Lock:transactionid.

Lock:tuple

Un processo di backend è in attesa di acquisire un blocco su una tupla mentre un altro processo di backend tiene un blocco in conflitto sulla stessa tupla. Per ulteriori informazioni, consulta Lock:tuple.

Lock:userlock

Un processo è in attesa di ottenere un blocco utente.

Lock:virtualxid

Un processo è in attesa di ottenere un blocco ID di transazione virtuale.

LWLock:AddinShmemInit

Un processo è in attesa di gestire l'allocazione dello spazio di un'estensione nella memoria condivisa.

LWLock:AddinShmemInitLock

Un processo è in attesa di gestire l'allocazione dello spazio nella memoria condivisa.

LWLockUn processo è in attesa che il capogruppo aggiorni lo stato della transazione al termine della transazione. ----sep----:async

Un processo è in attesa di I/O su un buffer asincrono (notifica).

LWLock:AsyncCtlLock

Un processo è in attesa di leggere o aggiornare uno stato di notifica condiviso.

LWLock:AsyncQueueLock

Un processo è in attesa di leggere o aggiornare i messaggi di notifica.

LWLock:AuroraOptimizedReadsCacheMapping

Un processo è in attesa di associare un blocco di dati a una pagina nella cache a più livelli di Letture ottimizzate.

LWLock:AutoFile

Un processo è in attesa di aggiornare il file `postgresql.auto.conf`.

LWLock:AutoFileLock

Un processo è in attesa di aggiornare il file `postgresql.auto.conf`.

LWLock:async ----sep----:Autovacuum

Un processo è in attesa di leggere o aggiornare lo stato attuale dei lavori autovacuum.

LWLock:AutovacuumLock

Un lavoro o un programma di avvio autovacuum è in attesa di aggiornare o leggere lo stato attuale dei lavori autovacuum.

LWLock:AutovacuumSchedule

Un processo è in attesa di garantire che una tabella selezionata per l'autovacuum necessita ancora di vacuum.

LWLock:AutovacuumScheduleLock

Un processo è in attesa di garantire che la tabella selezionata per il vacuum necessiti ancora di vacuum.

LWLock:BackendRandomLock

Un processo è in attesa di generare un numero casuale.

LWLock:BackgroundWorker

Un processo è in attesa di leggere o aggiornare lo stato del lavoro in background.

LWLock:BackgroundWorkerLock

Un processo è in attesa di leggere o aggiornare lo stato del lavoro in background.

LWLock:BtreeVacuum

Un processo è in attesa di leggere o aggiornare le informazioni relative al vacuum per un indice B-tree.

LWLock:BtreeVacuumLock

Un processo è in attesa di leggere o aggiornare le informazioni relative al vacuum per un indice B-tree.

LWLock:Autovacuum ----sep----:buffer_content

Un processo di backend è in attesa di acquisire un blocco leggero sui contenuti di un buffer di memoria condiviso. Per ulteriori informazioni, consulta LWLock:buffer_content (BufferContent).

LWLock:buffer_content ----sep----:buffer_mapping

Un processo di backend è in attesa di associare un blocco di dati a un buffer nel pool di buffer condiviso. Per ulteriori informazioni, consulta LWLock:buffer_mapping.

LWLock:buffer_mapping ----Sep----:Bufferio

Un processo di back-end vuole leggere una pagina nella memoria condivisa. Il processo è in attesa che altri processi finiscano l'I/O per la pagina. Per ulteriori informazioni, consulta LWLock:bufferIO (IPC: bufferio).

LWLock:bufferIO ----Sep----:Punto di controllo

Un processo è in attesa di iniziare un checkpoint.

LWLock:CheckpointLock

Un processo è in attesa di eseguire il checkpoint.

LWLock:CheckpointerComm

Un processo è in attesa di gestire le richieste `fsync`.

LWLock:CheckpointerCommLock

Un processo è in attesa di gestire le richieste `fsync`.

LWLock:Punto di controllo ----sep----:zoccolo

Un processo è in attesa di I/O su un buffer di clog (stato transazione).

LWLock:CLogControlLock

Un processo è in attesa di leggere o aggiornare lo stato della transazione.

LWLock:CLogTruncationLock

Un processo è in attesa di eseguire `txid_status` o di aggiornare l'ID di transazione meno recente disponibile.

LWLock:clog ----sep----:commit_timestamp

Un processo è in attesa di I/O su un buffer timestamp commit.

LWLock:CommitTs

Un processo è in attesa di leggere o aggiornare l'ultimo valore impostato per un timestamp commit della transazione.

LWLock:CommitTsBuffer

Un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un timestamp commit.

LWLock:CommitTsControlLock

Un processo è in attesa di leggere o aggiornare i timestamp commit della transazione.

LWLock:CommitTsLock

Un processo è in attesa di leggere o aggiornare l'ultimo valore impostato per il timestamp della transazione.

LWLock: SLRU CommitTs

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un timestamp commit.

LWLock:ControlFile

Un processo è in attesa di leggere o aggiornare il file `pg_control` o di creare un nuovo file WAL (write-ahead log).

LWLock:ControlFileLock

Un processo è in attesa di leggere o aggiornare il control file o la creazione di un nuovo file WAL (write-ahead log).

LWLock:DynamicSharedMemoryControl

Un processo è in attesa di leggere o aggiornare le informazioni dinamiche di allocazione della memoria condivisa.

LWLock:DynamicSharedMemoryControlLock

Un processo è in attesa di leggere o aggiornare lo stato dinamico della memoria condivisa.

LWLock: SLRU ----sep----:lock_manager

Un processo di backend è in attesa di aggiungere o esaminare i blocchi per i processi di backend. Oppure è in attesa di unirsi o uscire da un gruppo di blocco utilizzato dalla query parallela. Per ulteriori informazioni, consulta LWLock:lock_manager.

LWLock:LockFastPath

Un processo è in attesa di leggere o aggiornare le informazioni di blocco del percorso rapido di un processo.

LWLock:LogicalRepWorker

Un processo è in attesa di leggere o aggiornare lo stato dei lavori di replica logica.

LWLock:LogicalRepWorkerLock

Un processo è in attesa del termine di un'azione su un lavoro di replica logica.

LWLock:LogicalSchemaCache

Un processo ha modificato la cache dello schema.

LWLockUn processo ha modificato la cache dello schema. ----sep----:multixact_member

Un processo è in attesa di I/O su un buffer multixact_member.

LWLock:multixact_member ----sep----:multixact_offset

Un processo è in attesa di I/O su un buffer offset multixact.

LWLock:MultiXactGen

Un processo è in attesa di leggere o aggiornare lo stato multixact condiviso.

LWLock:MultiXactGenLock

Un processo è in attesa di leggere o aggiornare uno stato multixact condiviso.

LWLock:MultiXactMemberBuffer

Un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un membro multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactMemberControlLock

Un processo è in attesa di leggere o aggiornare le mappature dei membri multixact.

LWLock: SLRU MultiXactMember

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un membro multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactOffsetBuffer

Un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un offset multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactOffsetControlLock

Un processo è in attesa di leggere o aggiornare le mappature offset multixact.

LWLock: MultiXactOffset SLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un offset multixact. Per ulteriori informazioni, consulta LWLock:MultiXact.

LWLock:MultiXactTruncation

Un processo è in attesa di leggere o troncare informazioni multixact.

LWLock:MultiXactTruncationLock

Un processo è in attesa di leggere o troncare informazioni multixact.

LWLock:NotifyBuffer

Un processo è in attesa di I/O sul buffer semplice utilizzato meno di recente (SLRU) per un messaggio NOTIFY (NOTIFICA).

LWLock:NotifyQueue

Un processo è in attesa di leggere o aggiornare i messaggi NOTIFY (NOTIFICA).

LWLock:NotifyQueueTail

Un processo è in attesa di aggiornare un limite per l'archiviazione dei messaggi NOTIFY (NOTIFICA).

LWLock:NotifyQueueTailLock

Un processo è in attesa di aggiornare il limite sull'archiviazione dei messaggi di notifica.

LWLock: SLRU ----SEP----:notifySLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un messaggio NOTIFY (NOTIFICA).

LWLock:OidGen

Un processo è in attesa di allocare un nuovo ID oggetto (OID).

LWLock:OidGenLock

Un processo è in attesa di allocare o assegnare un ID oggetto (OID).

LWLock:notifySLRU ----sep----:oldserxid

Un processo è in attesa di I/O su un buffer oldserxid.

LWLock:OldSerXidLock

Un processo è in attesa di leggere o registrare transazioni serializzabili in conflitto.

LWLock:OldSnapshotTimeMap

Un processo è in attesa di leggere o aggiornare le informazioni meno recenti sul controllo degli snapshot.

LWLock:OldSnapshotTimeMapLock

Un processo è in attesa di leggere o aggiornare le informazioni meno recenti sul controllo degli snapshot.

LWLock:oldserxid ----sep----:parallel_append

Un processo è in attesa di scegliere il subplan successivo durante l'esecuzione parallela del piano di aggiunta.

LWLock:parallel_append ----sep----:parallel_hash_join

Un processo è in attesa di allocare o scambiare un blocco di memoria o di aggiornare i contatori durante l'esecuzione di un piano hash parallelo.

LWLock:parallel_hash_join ----sep----:parallel_query_dsa

Un processo è in attesa di un blocco sull'allocazione dinamica della memoria condivisa per una query parallela.

LWLock:ParallelAppend

Un processo è in attesa di scegliere il subplan successivo durante l'esecuzione parallela del piano di aggiunta.

LWLock:ParallelHashJoin

Un processo è in attesa di sincronizzare i lavori durante l'esecuzione del piano per un hash join parallelo.

Lucchetto: DSA ParallelQuery

Un processo è in attesa di un'allocazione dinamica della memoria condivisa per una query parallela.

Lwlock: DSA PerSession

Un processo è in attesa di un'allocazione dinamica della memoria condivisa per una query parallela.

Serratura: PerSessionRecordType

Un processo è in attesa di accedere alle informazioni di una query parallela sui tipi compositi.

Chiusura: PerSessionRecordTypmod

Un processo è in attesa di accedere alle informazioni di una query parallela sui modificatori di tipo che identificano i tipi di record anonimi.

Chiusura: PerXactPredicateList

Un processo è in attesa di accedere all'elenco dei blocchi del predicato tenuti dalla transazione serializzabile corrente durante una query parallela.

Lwlock:predicate_lock_manager

Un processo è in attesa di aggiungere o esaminare le informazioni sul blocco dei predicati.

Chiusura: PredicateLockManager

Un processo è in attesa di accedere alle informazioni sul blocco predicato utilizzate dalle transazioni serializzabili.

Lwlock:proc

Un processo è in attesa di leggere o aggiornare le informazioni sul blocco del percorso rapido.

LWLock:ProcArray

Un processo è in attesa di accedere alle strutture dati condivise per processo (in genere, per ottenere uno snapshot o segnalare l'ID di transazione di una sessione).

LWLock:ProcArrayLock

Un processo è in attesa di ottenere uno snapshot o di cancellare un'ID di transazione alla fine di una transazione.

LWLock:RelationMapping

Un processo è in attesa di leggere o aggiornare un file `pg_filenode.map` (utilizzato monitorare le assegnazioni file-nodo di alcuni cataloghi di sistema).

LWLock:RelationMappingLock

Un processo è in attesa di aggiornare il file della mappa delle relazioni utilizzato per catalog-to-file-node memorizzare la mappatura.

LWLock:RelCacheInit

Un processo è in attesa di leggere o aggiornare un file `pg_internal.init` (un file di inizializzazione della cache delle relazioni).

LWLock:RelCacheInitLock

Un processo è in attesa di leggere o scrivere un file di inizializzazione della cache delle relazioni.

LWLockUn processo è in attesa di aggiornare il file della mappa delle relazioni utilizzato per memorizzare la mappatura. ----sep----:replication_origin

Un processo è in attesa di leggere o aggiornare l'avanzamento della replica.

LWLock:replication_origin ----sep----:replication_slot_io

Un processo è in attesa di I/O su uno slot di replica.

LWLock:ReplicationOrigin

Un processo è in attesa di creare, eliminare o utilizzare un'origine di replica.

LWLock:ReplicationOriginLock

Un processo è in attesa di configurare, eliminare o utilizzare un'origine di replica.

LWLock:ReplicationOriginState

Un processo è in attesa di leggere o aggiornare l’avanzamento di un'origine di replica.

LWLock:ReplicationSlotAllocation

Un processo è in attesa di allocare o liberare uno slot di replica.

LWLock:ReplicationSlotAllocationLock

Un processo è in attesa di allocare o liberare uno slot di replica.

LWLock:ReplicationSlotControl

Un processo è in attesa di leggere o aggiornare lo stato di uno slot di replica.

LWLock:ReplicationSlotControlLock

Un processo è in attesa di leggere o aggiornare lo stato dello slot di replica.

LWLock: IO ReplicationSlot

Un processo è in attesa di I/O su uno slot di replica.

LWLock:SerialBuffer

Un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per un conflitto di transazioni serializzabile.

LWLock:SerializableFinishedList

Un processo è in attesa di accedere all'elenco delle transazioni serializzabili finite.

LWLock:SerializableFinishedListLock

Un processo è in attesa di accedere all'elenco delle transazioni serializzabili finite.

LWLock:SerializablePredicateList

Un processo è in attesa di accedere all'elenco dei blocchi del predicato tenuti dalle transazioni serializzabili.

LWLock:SerializablePredicateLockListLock

Un processo è in attesa di eseguire un'operazione su un elenco di blocchi tenuti dalle transazioni serializzabili.

LWLock:SerializableXactHash

Un processo è in attesa di leggere o aggiornare le informazioni sulle transazioni serializzabili.

LWLock:SerializableXactHashLock

Un processo è in attesa di recuperare o archiviare informazioni sulle transazioni serializzabili.

LWLock: IO ----SEP----:SerialSLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per un conflitto di transazioni serializzabile.

LWLock:SharedTidBitmap

Un processo è in attesa di accedere a una bitmap TID (shared tuple identifier) durante una scansione parallela dell'indice bitmap.

LWLock:SharedTupleStore

Un processo è in attesa di accedere a un archivio tupla condiviso durante una query parallela.

LWLock:ShmemIndex

Un processo è in attesa di trovare o allocare spazio nella memoria condivisa.

LWLock:ShmemIndexLock

Un processo è in attesa di trovare o allocare spazio nella memoria condivisa.

LWLock: Leggi SInval

Un processo è in attesa di recuperare i messaggi dalla coda di invalidamento del catalogo condiviso.

LWLock:SInvalReadLock

Un processo è in attesa di recuperare o rimuovere messaggi da una coda di invalidazione condivisa.

LWLock: SInval Scrivi

Un processo è in attesa di aggiungere un messaggio alla coda di invalidazione del catalogo condivisa.

LWLock:SInvalWriteLock

Un processo è in attesa di aggiungere un messaggio in una coda di invalidazione condivisa.

LWLock: Scrivi ----sep----:subtrans

Un processo è in attesa di I/O su un buffer delle transazioni secondarie.

LWLock:SubtransBuffer

Un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per una transazione secondaria.

LWLock:SubtransControlLock

Un processo è in attesa di leggere o aggiornare le informazioni sulla transazione secondaria.

LWLock:subtrans ----sep----:SubTransSLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per una transazione secondaria.

LWLock:SyncRep

Un processo è in attesa di leggere o aggiornare le informazioni sullo stato della replica sincrona.

LWLock:SyncRepLock

Un processo è in attesa di leggere o aggiornare le informazioni sulle repliche sincrone.

LWLock:SyncScan

Un processo è in attesa di selezionare la posizione iniziale di una scansione sincronizzata della tabella.

LWLock:SyncScanLock

Un processo è in attesa di ottenere la posizione iniziale di una scansione su una tabella per le scansioni sincronizzate.

LWLock:TablespaceCreate

Un processo è in attesa di creare o eliminare uno spazio tabella.

LWLock:TablespaceCreateLock

Un processo è in attesa di creare o eliminare uno spazio tabella.

LWLock:SubTransSLRU ----sep----:tbm

Un processo è in attesa di un blocco iteratore condiviso su una bitmap ad albero (TBM).

LWLock:TwoPhaseState

Un processo è in attesa di leggere o aggiornare lo stato delle transazioni preparate.

LWLock:TwoPhaseStateLock

Un processo è in attesa di leggere o aggiornare lo stato delle transazioni preparate.

LWLock:tbm ----sep----:wal_insert

Un processo è in attesa di inserire il WAL (write-ahead log) in un buffer di memoria.

LWLock: Mappatura WALBuf

Un processo è in attesa di sostituire una pagina nei buffer WAL (write-ahead log).

LWLock:WALBufMappingLock

Un processo è in attesa di sostituire una pagina nei buffer WAL (write-ahead log).

LWLock:WALInsert

Un processo è in attesa di inserire i dati WAL (write-ahead log) in un buffer di memoria.

LWLock:WALWrite

Un processo è in attesa della scrittura dei buffer WAL (write-ahead log) su disco.

LWLock: Serratura WALWrite

Un processo è in attesa della scrittura dei buffer WAL (write-ahead log) su disco.

LWLock:WrapLimitsVacuum

Un processo è in attesa di aggiornare i limiti relativi all'ID di transazione e al consumo multixact.

LWLock:WrapLimitsVacuumLock

Un processo è in attesa di aggiornare i limiti relativi all'ID di transazione e al consumo multixact.

LWLock:XactBuffer

Un processo è in attesa di I/O su un buffer semplice utilizzato meno di recente (SLRU) per lo stato di una transazione.

LWLock: Blocca ----sep----:xACTSLRU

Un processo è in attesa di accedere alla cache semplice utilizzata meno di recente (SLRU) per lo stato di una transazione.

LWLock:XactTruncation

Un processo è in attesa di eseguire pg_xact_status o di aggiornare l'ID di transazione meno recente disponibile.

LWLock:XidGen

Un processo è in attesa di allocare un nuovo ID di transazione.

LWLock:XidGenLock

Un processo è in attesa di allocare o assegnare un ID di transazione.

Timeout: BaseBackupThrottle

Un processo è in attesa durante il backup di base quando si limita l'attività.

Tempo scaduto: PgSleep

Un processo di backend ha chiamato la funzione pg_sleep ed è in attesa della scadenza del timeout di riposo. Per ulteriori informazioni, consulta Timeout: PG Sleep.

Tempo scaduto: RecoveryApplyDelay

Un processo è in attesa di applicare il WAL (write-ahead log) durante il ripristino a causa di un'impostazione di ritardo.

Tempo scaduto: RecoveryRetrieveRetryInterval

Un processo è in attesa durante il ripristino quando i dati WAL (write-ahead log) non sono disponibili da nessuna fonte (pg_wal, archivio o flusso).

Tempo scaduto: VacuumDelay

Un processo è in attesa in un punto di ritardo del vacuum basato sui costi.

Per un elenco completo degli eventi di attesa PostgreSQL, consulta la pagina relativa al processo di raccolta delle statistiche e alle tabelle degli eventi di attesa nella documentazione di PostgreSQL.

Aggiornamenti del motore di database per Amazon Aurora Postgre SQL

Di seguito, puoi trovare informazioni sulle versioni e gli aggiornamenti del SQL motore Amazon Aurora Postgre. Puoi anche trovare informazioni su come aggiornare il tuo motore Aurora SQL Postgre. Per ulteriori informazioni generali sulle versioni di Aurora, consulta Versioni di Amazon Aurora.

> **Tip**
>
> È possibile ridurre al minimo i tempi di inattività necessari per l'aggiornamento di un cluster di database utilizzando un'implementazione blu/verde. Per ulteriori informazioni, consulta Utilizzo delle implementazioni blu/verde per gli aggiornamenti del database.

Argomenti

- Identificazione delle versioni di Amazon Aurora Postgre SQL
- Versioni e versioni del motore di Amazon Aurora Postgre SQL
- Versioni di estensione per Amazon Aurora Postgre SQL
- Aggiornamento dei cluster database Amazon Aurora PostgreSQL
- Utilizzo di una versione di Aurora PostgreSQL con supporto a lungo termine (LTS)

Identificazione delle versioni di Amazon Aurora Postgre SQL

Amazon Aurora include alcune funzionalità generiche per Aurora e disponibili per tutti i cluster database Aurora. Aurora include altre funzionalità specifiche per un particolare motore del database supportato da Aurora. Queste funzionalità sono disponibili solo per i cluster Aurora DB che utilizzano quel motore di database, come Aurora Postgre. SQL

Una versione del database Aurora in genere possiede due numeri di versione: il numero di versione del motore del motore del database e il numero di versione di Aurora. Se una versione di Aurora Postgre SQL ha un numero di versione di Aurora, viene incluso dopo il numero di versione del motore nell'elenco. Versioni e versioni del motore di Amazon Aurora Postgre SQL

Argomenti

- Numero versione Aurora

- Numeri di versione del motore Postgre SQL

Numero versione Aurora

I numeri di versione di Aurora utilizzano. *major minor*. *patch*schema di denominazione. Una versione della patch di Aurora include importanti correzioni di bug aggiunte a una versione secondaria dopo il rilascio. Per ulteriori informazioni sulle versioni principale, secondaria e patch di Amazon Aurora, consulta Versioni principali di Amazon Aurora, Versioni secondarie di Amazon Aurora e Versioni delle patch di Amazon Aurora.

È possibile scoprire il numero di versione di Aurora della propria istanza Aurora Postgre SQL DB con la seguente query: SQL

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

A partire dal rilascio delle SQL versioni 13.3, 12.8, 11.13, 10.18 e per tutte le altre versioni successive, i numeri di versione di Aurora si allineano maggiormente alla versione del motore Postgre. SQL Ad esempio, l'interrogazione di un cluster Aurora SQL Postgre 13.3 DB restituisce quanto segue:

```
aurora_version
----------------
 13.3.1
(1 row)
```

Le versioni precedenti, come il cluster Aurora Postgre SQL 10.14 DB, restituiscono numeri di versione simili ai seguenti:

```
aurora_version
----------------
 2.7.3
(1 row)
```

Numeri di versione del motore Postgre SQL

A partire da Postgre SQL 10, le versioni del motore di database Postgre utilizzano un. SQL *major minor*schema di numerazione per tutte le versioni. Alcuni esempi includono Postgre SQL 10.18, Postgre 12.7 e Postgre 13.3. SQL SQL

Le versioni precedenti SQL a *major* Postgre 10 utilizzavano un. *major*. *minor*schema di numerazione in cui le prime due cifre costituiscono il numero della versione principale e una terza cifra indica una versione secondaria. Ad esempio, Postgre SQL 9.6 era una versione principale, con le versioni secondarie 9.6.21 o 9.6.22 indicate dalla terza cifra.

> **Note**
>
> La versione 9.6 del motore SQL Postgre non è più supportata. Per eseguire l'aggiornamento, consulta Aggiornamento dei cluster database Amazon Aurora PostgreSQL. Per le policy delle versioni e le tempistiche delle versioni, consulta Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili.

Puoi scoprire il numero di versione del motore di SQL database Postgre con la seguente query: SQL

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

Per un cluster Aurora Postgre SQL 13.3 DB, i risultati sono i seguenti:

```
version
-----------------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
(1 row)
```

Versioni e versioni del motore di Amazon Aurora Postgre SQL

Le versioni di Amazon Aurora Postgre SQL -Compatible Edition vengono aggiornate regolarmente. Gli aggiornamenti vengono applicati ai cluster Aurora Postgre SQL DB durante le finestre di manutenzione del sistema. Il momento in cui vengono applicati gli aggiornamenti dipende dal tipo di aggiornamento, dall'impostazione della finestra di manutenzione e dall'impostazione della Regione AWS finestra di manutenzione per il cluster DB. Molte delle versioni elencate includono sia un numero di versione di Postgre che un numero di SQL versione di Amazon Aurora. Tuttavia, a partire dal rilascio delle SQL versioni 13.3, 12.8, 11.13, 10.18 di Postgre e per tutte le altre versioni successive, i numeri di versione di Aurora non vengono utilizzati. Per identificare i numeri di versione del database Aurora PostgreSQL, consulta. Identificazione delle versioni di Amazon Aurora Postgre SQL

Per informazioni su estensioni e moduli, consulta Versioni di estensione per Amazon Aurora Postgre SQL.

> **Note**
>
> Per ulteriori informazioni sulle policy delle versioni Amazon Aurora e sulle tempistiche delle versioni, consulta Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili.

Per informazioni sul supporto per Amazon Aurora, consulta Amazon. RDS FAQs

Per determinare quali versioni del motore Aurora Postgre SQL DB sono disponibili in un Regione AWS, utilizzare il comando. describe-db-engine-versions AWS CLI Per esempio:

```
aws rds describe-db-engine-versions --engine aurora-postgresql --query '*[].
[EngineVersion]' --output text --region aws-region
```

Per un elenco di Regioni AWS, vedi. Disponibilità nelle regioni Aurora PostgreSQL

Per informazioni dettagliate sulle SQL versioni di Postgre disponibili su Aurora PostgreSQL, consulta le Note di rilascio per Aurora Postgre. SQL

Versioni di estensione per Amazon Aurora Postgre SQL

È possibile installare e configurare varie SQL estensioni Postgre da utilizzare con i cluster Aurora Postgre DB. SQL Ad esempio, è possibile utilizzare l'SQL`pg_partman`estensione Postgre per automatizzare la creazione e la manutenzione delle partizioni delle tabelle. Per ulteriori informazioni su questa e altre estensioni disponibili per Aurora SQL Postgre, consulta. Utilizzo di estensioni e wrapper di dati esterni

Per informazioni dettagliate sulle SQL estensioni Postgre supportate su Aurora PostgreSQL, consulta Versioni di estensione per Amazon Aurora Postgre nelle Note di rilascio per Aurora SQL Postgre. SQL

Aggiornamento dei cluster database Amazon Aurora PostgreSQL

Amazon Aurora rende disponibili nuove versioni del motore di database PostgreSQL in Regioni AWS solo dopo test approfonditi. Puoi aggiornare i cluster database Aurora PostgreSQL alla nuova versione quando è disponibile nella tua regione.

A seconda della versione di Aurora PostgreSQL attualmente in esecuzione sul cluster database, un aggiornamento alla nuova versione è un aggiornamento secondario o principale. Ad esempio, l'aggiornamento di un cluster database Aurora PostgreSQL 11.15 ad Aurora PostgreSQL 13.6, è un aggiornamento della versione principale. L'aggiornamento di un cluster database Aurora PostgreSQL 13.3 ad Aurora PostgreSQL 13.7, è un aggiornamento della versione secondaria. Negli argomenti seguenti sono disponibili informazioni su come eseguire entrambi i tipi di aggiornamenti.

Indice

- Panoramica dei processi di aggiornamento di Aurora PostgreSQL
- Ottenere un elenco di versioni disponibili nel Regione AWS
- Esecuzione di un aggiornamento della versione principale
  - Test di un aggiornamento del cluster database di produzione a una nuova versione principale
  - Consigli per la fase successiva all'aggiornamento
  - Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale
    - Principali aggiornamenti per database globali
- Esecuzione di un aggiornamento di versione minore
  - Prima di eseguire un aggiornamento di una versione secondaria
  - Come eseguire aggiornamenti della versione secondaria e applicare patch
  - Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività
  - Limitazioni dell'applicazione di patch senza tempi di inattività
  - Aggiornamento del motore Aurora PostgreSQL a una nuova versione secondaria
- Aggiornamento estensioni PostgreSQL
- Tecnica alternativa di aggiornamento blu/verde

Panoramica dei processi di aggiornamento di Aurora PostgreSQL

Le differenze tra aggiornamenti della versione principale e secondaria sono le seguenti:

Aggiornamenti della versione secondaria e patch

> Aggiornamenti della versione secondaria e patch includono solo le modifiche compatibili con le versioni precedenti delle applicazioni esistenti. Gli aggiornamenti della versione secondaria e patch diventano disponibili solo dopo i test Aurora PostgreSQ e l'approvazione.

Aurora può applicare automaticamente aggiornamenti di versione minori. Quando si crea un nuovo cluster Aurora PostgreSQL DB, l'opzione Abilita aggiornamento della versione secondaria è abilitata per impostazione predefinita. A meno che non si disattivi manualmente questa opzione, Aurora applica periodicamente aggiornamenti automatici delle versioni secondarie durante la finestra di manutenzione programmata. Per ulteriori informazioni sull'opzione Aggiornamento automatico delle versioni minori (AmVU) e su come modificare il cluster database Aurora per utilizzarla, consulta Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora.

Se l'aggiornamento automatico della versione secondaria non è abilitato per il cluster Aurora PostgreSQL DB, Aurora PostgreSQL non viene aggiornato automaticamente a una nuova versione secondaria. Invece, quando una nuova versione secondaria viene rilasciata nella Regione AWS e il cluster database Aurora PostgreSQL sta eseguendo una versione secondaria precedente, Aurora richiede di eseguire l'aggiornamento. A questo scopo, viene aggiunto un suggerimento alle attività di manutenzione per il cluster.

Le patch non sono considerate un aggiornamento e non vengono applicate automaticamente. Aurora PostgreSQL richiede di applicare eventuali patch aggiungendo una raccomandazione alle attività di manutenzione per il cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Come eseguire aggiornamenti della versione secondaria e applicare patch.

> **Note**
>
> Anche le patch che risolvono la sicurezza o altri problemi critici vengono aggiunte come attività di manutenzione. Tuttavia, queste patch sono obbligatorie. Assicurati di applicare le patch di sicurezza al cluster database Aurora PostgreSQL quando diventano disponibili nelle attività di manutenzione in sospeso.

Il processo di aggiornamento implica la possibilità di brevi interruzioni poiché ogni istanza nel cluster viene aggiornata alla nuova versione. Tuttavia, dopo Aurora PostgreSQL versioni 14.3.3, 13.7.3, 12.11.3, 11.16.3, 10.21.3 e altre versioni successive di queste versioni secondarie e delle versioni principali più recenti, il processo di aggiornamento utilizza la funzionalità di applicazione di patch senza tempi di inattività (ZDP). Questa funzionalità riduce al minimo le interruzioni e nella maggior parte dei casi le elimina completamente. Per ulteriori informazioni, consulta Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività. Per ulteriori informazioni sulle funzionalità e le limitazioni supportate di ZDP, vedere. Limitazioni dell'applicazione di patch senza tempi di inattività

Aggiornamenti di una versione principale

A differenza degli aggiornamenti della versione secondaria e delle patch, Aurora PostgreSQL non dispone di un'opzione di aggiornamento automatico della versione principale. Nuove versioni PostgreSQL principali potrebbero contenere modifiche al database non compatibili con le versioni precedenti delle applicazioni esistenti. La nuova funzionalità può causare l'interruzione del corretto funzionamento delle applicazioni esistenti.

Per evitare problemi, è fortemente consigliato seguire il processo descritto in Test di un aggiornamento del cluster database di produzione a una nuova versione principale prima di aggiornare le istanze database nei cluster database Aurora PostgreSQL. Assicurati innanzitutto che le applicazioni possano essere eseguite sulla nuova versione seguendo tale procedura. Quindi, puoi aggiornare manualmente il cluster database Aurora PostgreSQL alla nuova versione.

Il processo di aggiornamento prevede la possibilità di una breve interruzione quando tutte le istanze del cluster vengono aggiornate alla nuova versione. Anche il processo di pianificazione preliminare richiede tempo. Si consiglia di eseguire sempre attività di aggiornamento durante la finestra di manutenzione del cluster o quando le operazioni sono minime. Per ulteriori informazioni, consulta Esecuzione di un aggiornamento della versione principale.

> **Note**
>
> Gli aggiornamenti della versione secondaria e quelli della versione principale potrebbero entrambi comportare brevi interruzioni. Per questo motivo, è fortemente consigliato eseguire o pianificare gli aggiornamenti durante la finestra di manutenzione o durante altri periodi di scarso utilizzo.

I cluster di database Aurora PostgreSQL richiedono occasionalmente gli aggiornamenti del sistema operativo. Questi aggiornamenti a volte includono una nuova versione della libreria glibc. Durante gli aggiornamenti, ti consigliamo di seguire le linee guida descritte in .

Ottenere un elenco di versioni disponibili nel Regione AWS

È possibile ottenere un elenco di tutte le versioni del motore disponibili come obiettivi di aggiornamento per il cluster Aurora PostgreSQL DB eseguendo una query utilizzando il comando, come segue. Regione AWS describe-db-engine-versions AWS CLI

Per, o: Linux macOS Unix

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Per Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Ad esempio, per identificare gli obiettivi di aggiornamento validi per un cluster DB Aurora PostgreSQL versione 12.10, esegui il comando seguente: AWS CLI

Per, o: Linux macOS Unix

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version 12.10 \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Per Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version 12.10 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Nella tabella seguente, puoi trovare gli obiettivi di aggiornamento delle versioni principali e secondarie per diverse versioni di Aurora PostgreSQL DB. Per mantenere la compatibilità, non tutte le versioni sono offerte come obiettivi di aggiornamento. Aurora PostgreSQL introduce nuove funzionalità e correzioni di bug con ogni rilascio trimestrale di versioni secondarie. Per informazioni sulle versioni secondarie di Aurora PostgreSQL, consulta le Note di rilascio per Aurora PostgreSQL.

Versione di origine corrente	Destinazioni di aggiornamento
17,4	17,4
16,8	17,4
16,6	17,4 16,8
16,4	17,4 16,8, 16,6
16,3	17,4 16,8, 16,6, 16,4
16,2	17,4 16,8 16,6, 16,4, 16,3
16,1	17,4 16,8, 16,6, 16,4,

Versione di origine corrente	Destinazioni di aggiornamento
	16,3, 16,2
15,12	17,4 16,8
15,10	17,4 16,8, 16,6 15,12
15,8	17,4 16,8, 16,6, 16,4 15,12, 15,10
15,7	17,4 16,8, 16,6, 16,4, 16,3 15,12, 15,10, 15,8

Versione di origine corrente	Destinazioni di aggiornamento
15,6	17,4 16,8, 16,6, 16,4, 16,3, 16,2 15,12, 15,10, 15,8, 15,7
15,5	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6

Versione di origine corrente	Destoni di aggiento
15,4	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5

Versione di origine corrente	Destinazioni di aggiornamento
15,3	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4

Versione di origine corrente	Destinazioni di aggiornamento
15,2	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3
14,17	17,4 16,8 15,12

Versione di origine corrente	Destinazioni di aggiornamento
14,15	17,4 16,8, 16,6 15,12, 15,10 14,17
14,13	17,4 16,8, 16,6, 16,4 15,12, 15,10, 15,8 14,17, 14,15

Versione di origine corrente	Destinazioni di aggiornamento
14,12	17,4 16,8 , 16,6, 16,4, 16,3 15,12, 15,10, 15,8 , 15,7 14,17, 14,15, 14,13

Versione di origine corrente	Destinazioni di aggiornamento
14,11	17,4 16,8, 16,6, 16,4, 16,3, 16,2 15,12, 15,10, 15,8, 15,6 14,17, 14,15, 14,13, 14,12

Versione di origine corrente	Destinazioni di aggiornamento
14,10	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5 14,17, 14,15, 14,13, 14,12, 14,11

Versione di origine corrente	Destinazioni di aggiornamento
14,9	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4 14,17, 14,15, 14,13, 14,12, 14,11, 14,10

Versione di origine corrente	Destinazioni di aggiornamento
14,8	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12 , 14,11, 14,10, 14,9

Versione di origine corrente	Destinazioni di aggiornamento
14,7	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8

Versione di origine corrente	Destinazioni di aggiornamento
14,6	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7

Versione di origine corrente	Destinazioni di aggiornamento
14,5	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8,

Versione di origine corrente	Destoni di aggiento
	14,7, 14,6

Versione di origine corrente	Destinazioni di aggiornamento
14,4	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7, 14,6

Versione di origine corrente	Destinazioni di aggiornamento
	, 14,5

Versione di origine corrente	Destinazioni di aggiornamento
14,3	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7,

Versione di origine corrente	Destinazioni di aggiornamento
	14,6, 14,5, 14,4
13,20	17,4 16,8 15,12 14,17
13,18	17,4 16,8, 16,6 15,12, 15,10 14,17, 14,15 13,20

Versione di origine corrente	Destinazioni di aggiornamento
13,16	17,4 16,8, 16,6, 16,4 15,12, 15,10, 15,8 14,17, 14,15, 14,13 13,20, 13,18

Versione di origine corrente	Destinazioni di aggiornamento
13,15	17,4 16,8, 16,6, 16,4, 16,3 15,12, 15,10, 15,8, 15,7 14,17, 14,15, 14,13, 14,12 13,20, 13,18, 13,16

Versione di origine corrente	Destinazioni di aggiornamento
13,14	17,4 16,8 , 16,6, 16,4, 16,3, 16,2 15,12, 15,10, 15,8 , 15,7, 15,6 14,17, 14,15 , 14,13, 14,11 13,20, 13,18, 13,16 , 13,15

Versione di origine corrente	Destinazioni di aggiornamento
13,13	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5 14,17, 14,15, 14,13, 14,12, 14,11, 14,10 13,20, 13,18, 13,16, 13,15

Versione di origine corrente	Destinazioni di aggiornamento
	13,14

Versione di origine corrente	Destinazioni di aggiornamento
13,12	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6 , 15,5, 15,4 14,17, 14,15, 14,13, 14,12, 14,11, 14,10 , 14,9 13,20, 13,18 , 13,16,

Versione di origine corrente	Destinazioni di aggiornamento
	13,15, 13,14, 13,13

Versione di origine corrente	Destinazioni di aggiornamento
13,11	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6 , 15,5, 15,4, 15,3 14,17, 14,15, 14,13, 14,12, 14,11, 14,10 , 14,9, 14,8 13,20, 13,18

Versione di origine corrente	Destinazioni di aggiornamento
	13.16, 13.15, 13.14, 13.13, 13.12

Versione di origine corrente	Destinazioni di aggiornamento
13,10	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7

Versione di origine corrente	Destinazioni di aggiornamento
	13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11

Versione di origine corrente	Destinazioni di aggiornamento
13,9	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8,

Versione di origine corrente	Destinazioni di aggiornamento
	14,7, 14,6 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10

Versione di origine corrente	Destinazioni di aggiornamento
13,8	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7, 14,6

Versione di origine corrente	Destinazioni di aggiornamento
	14,5 13,20, 13,18, 13,16, 13,15, 13,14, 13,13 13,12, 13,11, 13,10, 13,9

Versione di origine corrente	Destinazioni di aggiornamento
13,7	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7,

Versione di origine corrente	Destinazioni di aggiornamento
	14,6, 14,5, 14,4, 14,3 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10, 13,9, 13,8
12,22	17,4 16,8, 16,6 15,12, 15,10 14,17, 14,15 13,20, 13,18

Versione di origine corrente	Destinazioni di aggiornamento
12,20	17,4 16,8, 16,6, 16,4 15,12, 15,10, 15,8 14,15, 14,13 13,20, 13,18, 13,16 12,22

Versione di origine corrente	Destinazioni di aggiornamento
12,19	17,4 16,8 , 16,6, 16,4, 16,3 15,12, 15,10, 15,8 , 15,7 14,17, 14,15, 14,13 , 14,12 13,20, 13,18, 13,16 , 13,15 12,22, 12,20

Versione di origine corrente	Destinazioni di aggiornamento
12,18	17,4 16,8 , 16,6, 16,4, 16,3, 16,2 15,12, 15,10, 15,8 , 15,7, 15,6 14,17, 14,15, 14,13 , 14,12, 14,11 13,20, 13,18, 13,16 , 13,15, 13,14

Versione di origine corrente	Destinazioni di aggiornamento
	12,22, 12,20, 12,19

Versione di origine corrente	Destinazioni di aggiornamento
12,17	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8 , 15,7, 15,6, 15,5 14,17, 14,15, 14,13 , 14,12, 14,11, 14,10 13,20, 13,18, 13,16 , 13,15,

Versione di origine corrente	Destinazioni di aggiornamento
	13,14, 13,13 12,22, 12,20, 12,19, 12,18

Versione di origine corrente	Destinazioni di aggiornamento
12,16	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6 , 15,5, 15,4 14,17, 14,15, 14,13, 14,12, 14,11, 14,10 , 14,9 13,20, 13,18, 13,16, 13,15,

Versione di origine corrente	Destinazioni di aggiornamento
	13,14, 13,13, 13,12 12,22, 12,20, 12,19, 12,18, 12,17

Versione di origine corrente	Destinazioni di aggiornamento
12,15	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6 , 15,5, 15,4, 15,3 14,17, 14,15, 14,13, 14,12, 14,11, 14,10 , 14,9, 14,8 13,20, 13,18,

Versione di origine corrente	Destinazioni di aggiornamento
	13,16, 13,15, 13,14, 13,13, 13,12, 13,11 12,22, 12,20, 12,19, 12,18, 12,17, 12,16

Versione di origine corrente	Destinazioni di aggiornamento
12,14	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9,

Versione di origine corrente	Destinazioni di aggiornamento
	14,8, 14,7 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10 12,22, 12,20, 12,19, 12,18, 12,17, 12,16, 12,15

Versione di origine corrente	Destinazioni di aggiornamento
12,13	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8 ,

Versione di origine corrente	Destinazioni di aggiornamento
	14,7, 14,6 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10, 13,9 12,22, 12,20, 12,19, 12,18, 12,17, 12,16, 12,15, 12,14

Versione di origine corrente	Destinazioni di aggiornamento
12,12	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7, 14,6

Versione di origine corrente	Destinazioni di aggiornamento
	14,5 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10 13,9, 13,8 12,22, 12,20, 12,19, 12,18, 12,17, 12,16, 12,15, 12,14 12,13

Versione di origine corrente	Destinazioni di aggiornamento
12,11	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17 , 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7,

Versione di origine corrente	Destoni di aggiento
	14,5, 14,4, 14,3 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10, 13,9, 13,8, 13,7 12,22, 12,20, 12,19, 12,18, 12,17, 12,16, 12,15, 12,14, 12,13, 12,12

Versione di origine corrente	Destinazioni di aggiornamento
12,9	17,4 16,8, 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8, 14,7

Versione di origine corrente	Destinazioni di aggiornamento
	13,20, 13,18, 13,16, 13,14, 13,13, 13,12, 13,11, 13,10, 13,9, 13,8, 13,7 12,22, 12,20, 12,19, 12,18, 12,17, 12,16, 12,15, 12,14, 12,13, 12,12, 12,11

Versione di origine corrente	Destinazioni di aggiornamento
11,21	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12, 15,10, 15,8, 15,7, 15,6 , 15,5, 15,4 14,17, 14,15, 14,13, 14,12, 14,11, 14,10 , 14,9 13,20, 13,18, 13,16, 13,15,

Versione di origine corrente	Destinazioni di aggiornamento
	13,14, 13,13, 13,12 12,22, 12,20, 12,19, 12,18, 12,17, 12,16

Versione di origine corrente	Destoni di aggiento
11.9	17,4 16,8 , 16,6, 16,4, 16,3, 16,2, 16,1 15,12 , 15,10, 15,8, 15,7, 15,6, 15,5, 15,4, 15,3, 15,2 14,17, 14,15, 14,13, 14,12, 14,11, 14,10, 14,9, 14,8 ,

Versione di origine corrente	Destinazioni di aggiornamento
	14,7, 14,6 13,20, 13,18, 13,16, 13,15, 13,14, 13,13, 13,12, 13,11, 13,10, 13,9 12,22, 12,20, 12,19, 12,18, 12,17, 12,16, 12,15, 12,14, 12,13, 12,12, 12,11, 12,09 11,21

Per qualsiasi versione che stai considerando, controlla sempre la disponibilità della classe di istanza database del cluster. Ad esempio, `db.r4` non è supportato per Aurora PostgreSQL 13. Se il tuo cluster database Aurora PostgreSQL utilizza attualmente una classe di istanza db.r4, devi passare a db.r5 prima di provare a eseguire l'aggiornamento. Per ulteriori informazioni sulle classi di istanza database, tra cui quali sono basate su Graviton2 e quali su Intel, consulta Classi di istanze DB Amazon Aurora.

Esecuzione di un aggiornamento della versione principale

Gli aggiornamenti a una versione principale potrebbero contenere modifiche al database non compatibili con le versioni precedenti del database. La nuova funzionalità in una nuova versione può causare l'interruzione del funzionamento corretto delle applicazioni esistenti. Per evitare problemi, Amazon Aurora non applica automaticamente aggiornamenti alla versione principale. Si consiglia, invece, di pianificare attentamente un aggiornamento alla versione principale seguendo questi passaggi:

1. Scegli la versione principale desiderata dall'elenco delle destinazioni disponibili tra quelle elencate per la versione nella tabella. È possibile ottenere un elenco preciso delle versioni disponibili nella Regione AWS versione corrente utilizzando il AWS CLI. Per informazioni dettagliate, consultare Ottenere un elenco di versioni disponibili nel Regione AWS.
2. Verifica che le applicazioni funzionino come previsto su un'implementazione di prova della nuova versione. Per informazioni sul processo completo, consulta Test di un aggiornamento del cluster database di produzione a una nuova versione principale.
3. Dopo aver verificato che le applicazioni funzionano come previsto nell'implementazione di prova, puoi aggiornare il cluster. Per informazioni dettagliate, consultare Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.

> **Note**
>
> È possibile eseguire un aggiornamento della versione principale da Babelfish per Aurora PostgreSQL dalle versioni di Aurora PostgreSQL 13 a partire dalla 13.6 alle versioni di Aurora PostgreSQL 14 a partire dalla 14.6. Babelfish per Aurora PostgreSQL 13.4 e 13.5 non supporta l'aggiornamento della versione principale.

È possibile ottenere un elenco delle versioni del motore disponibili come obiettivi di aggiornamento delle versioni principali per il cluster Aurora PostgreSQL DB interrogando l'utente utilizzando il comando, come segue. Regione AWS describe-db-engine-versions AWS CLI

In Linux, macOS, oppure Unix:

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}' \
  --output text
```

In Windows:

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}" ^
  --output text
```

In alcuni casi, la versione di destinazione dell'aggiornamento non è una destinazione per la versione corrente. In questi casi, utilizza le informazioni contenute in versions table per eseguire aggiornamenti della versione secondaria fino a quando la versione del cluster non dispone della destinazione scelta nella relativa riga di destinazioni.

Test di un aggiornamento del cluster database di produzione a una nuova versione principale

Ogni nuova versione principale include miglioramenti all'ottimizzatore di query progettati per migliorare le prestazioni. Tuttavia, il carico di lavoro può includere query che restituiscono prestazioni peggiori nella nuova versione. Ecco perché ti consigliamo di testare e rivedere le prestazioni prima di eseguire l'aggiornamento in produzione. È possibile gestire la stabilità del piano di query tra le varie versioni utilizzando l'estensione Query Plan Management (QPM), come descritto in Garantire la stabilità del piano dopo un aggiornamento di versione principale.

Prima di aggiornare i cluster database Aurora PostgreSQL di produzione a una nuova versione principale, è fortemente consigliato eseguire il test dell'aggiornamento per verificare che tutte le applicazioni funzionino correttamente:

1. Tieni a portata di mano un gruppo di parametri compatibile con la versione.

Se utilizzi un'istanza database personalizzata o un gruppo di parametri del cluster database, puoi scegliere tra due opzioni:

a. Puoi specificare l'istanza database predefinita, il gruppo di parametri del cluster di database o entrambi per la nuova versione del motore di database.

b. Oppure è possibile creare un gruppo di parametri personalizzato per la nuova versione del motore database.

Se associ una nuova istanza database o gruppo di parametri del cluster di database come una parte della richiesta di aggiornamento, devi riavviare il database al termine dell'aggiornamento per applicare i parametri. Se, per applicare le modifiche del gruppo di parametri, un'istanza deve essere riavviata, lo stato del gruppo di parametri è `pending-reboot`. È possibile visualizzare lo stato del gruppo di parametri dell'istanza nella console oppure utilizzando un comando CLI come describe-db-instances o describe-db-clusters.

2. Verifica la presenza di database non validi ed elimina quelli esistenti.

La `datconnlimit` colonna del `pg_database` catalogo include il valore `-2` per contrassegnare come non validi tutti i database che sono stati interrotti durante un'operazione. `DROP DATABASE` Utilizzate la seguente query per verificare se esistono database non validi.

```
SELECT
    datname
FROM
    pg_database
WHERE
    datconnlimit = - 2;
```

Se la query restituisce nomi di database, questi database non sono validi. Utilizzate l'`DROP DATABASE invalid_db_name`istruzione per eliminare i database non validi. È possibile utilizzare la seguente istruzione dinamica per eliminare tutti i database non validi.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

3. Controllare l'utilizzo non supportato:

- Eseguire il commit o il rollback di tutte le transazioni preparate aperte prima di provare a eseguire un aggiornamento. È possibile utilizzare la seguente query per verificare che sull'istanza non siano presenti transazioni preparate aperte.

```
SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
```

- Rimuovere tutti gli utilizzi dei tipi di dati reg* prima di tentare un aggiornamento. Ad eccezione di `regtype` e `regclass`, non è possibile aggiornare i tipi di dati reg*. L'utilità pg_upgrade (utilizzata da Amazon Aurora per eseguire l'aggiornamento) non può preservare questo tipo di dati. Per ulteriori informazioni su questa utilità, consulta pg_upgrade nella documentazione di PostgreSQL.

  Per verificare che non siano presenti utilizzi di tipi di dati reg* non supportati, utilizzare la query seguente per ogni database.

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
 pg_catalog.pg_attribute a
  WHERE c.oid = a.attrelid
      AND NOT a.attisdropped
      AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                         'pg_catalog.regprocedure'::pg_catalog.regtype,
                         'pg_catalog.regoper'::pg_catalog.regtype,
                         'pg_catalog.regoperator'::pg_catalog.regtype,
                         'pg_catalog.regconfig'::pg_catalog.regtype,
                         'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

- Se stai eseguendo l'aggiornamento di un cluster di database Aurora PostgreSQL versione 10.18 o successive e l'estensione `pgRouting` è installata, rimuovila prima di eseguire l'aggiornamento alla versione 12.4 o successive.

  Se stai eseguendo l'aggiornamento di Aurora PostgreSQL 10.x che ha l'estensione `pg_repack` versione 1.4.3 installata, rimuovi l'estensione prima di eseguire l'aggiornamento a una versione successiva.

4. Controllare i database template1 e template0.

   Per un aggiornamento corretto, i database template1 e template0 devono esistere e devono essere elencati come modello. Per eseguire questa verifica, utilizzare il seguente comando:

```
SELECT datname, datistemplate FROM pg_database;

datname    | datistemplate
-----------+---------------
template0  | t
rdsadmin   | f
template1  | t
postgres   | f
```

Nell'output del comando, il valore `datistemplate` per i database template1 e template0 deve essere `t`.

5. Rimuovi slot di replica logica.

   Il processo di aggiornamento non può continuare se il cluster database Aurora PostgreSQL utilizza slot di replica logici. Gli slot di replica logica vengono in genere utilizzati per attività di migrazione dei dati a breve termine, come la migrazione dei dati utilizzando AWS DMS o per replicare tabelle dal database ai data lake, agli strumenti di BI o ad altre destinazioni. Prima di eseguire l'aggiornamento, assicurati di conoscere lo scopo degli eventuali slot di replica logica esistenti e verifica che sia corretto eliminarli. Puoi verificare gli slot di replica logica utilizzando la seguente query:

   ```
   SELECT * FROM pg_replication_slots;
   ```

   Se gli slot di replica logica sono ancora in uso, non devono essere eliminati e non è possibile procedere con l'aggiornamento. Tuttavia, se gli slot di replica logica non sono necessari, è possibile eliminarli utilizzando il seguente SQL:

   ```
   SELECT pg_drop_replication_slot(slot_name);
   ```

   Negli scenari di replica logica che utilizzano l'estensione `pglogical` devono inoltre essere eliminati gli slot dal nodo publisher per un corretto aggiornamento della versione principale su quel nodo. Tuttavia, è possibile riavviare il processo di replica dal nodo sottoscrittore dopo l'aggiornamento. Per ulteriori informazioni, consulta Riconnessione della replica logica dopo un aggiornamento principale.

6. Eseguire un backup.

Il processo di aggiornamento crea uno snapshot del cluster di database durante l'aggiornamento. Se desideri eseguire anche un backup manuale prima del processo di aggiornamento, consulta Creazione di uno snapshot del cluster database per ulteriori informazioni.

7. Aggiornare alcune estensioni alla versione più recente disponibile prima di eseguire l'aggiornamento della versione principale. Le estensioni da aggiornare includono le seguenti:
   - `pgRouting`
   - `postgis_raster`
   - `postgis_tiger_geocoder`
   - `postgis_topology`
   - `address_standardizer`
   - `address_standardizer_data_us`

   Esegui il comando seguente per ogni estensione attualmente installata.

   ```
   ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
   ```

   Per ulteriori informazioni, consulta Aggiornamento estensioni PostgreSQL. Per ulteriori informazioni sull'aggiornamento di PostGIS, consulta Passaggio 6: Aggiorna l'GISestensione Post.
8. Se si esegue l'aggiornamento alla versione 11.x, eliminare le estensioni che non supporta prima di eseguire l'aggiornamento della versione principale. Le estensioni da eliminare includono:
   - `chkpass`
   - `tsearch2`
9. Eliminare i tipi di dati `unknown`, a seconda della versione di destinazione.

   La versione 10 di PostgreSQL non supporta il tipo di dati `unknown`. Se un database versione 9.6 utilizza il tipo di dati `unknown`, un aggiornamento alla versione 10 mostra un messaggio di errore del tipo seguente:

   ```
   Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
   The instance could not be upgraded because the 'unknown' data type is used in user
    tables.
   Please remove all usages of the 'unknown' data type and try again."
   ```

   Per trovare il tipo di dati `unknown` nel database in modo da poter rimuovere tali colonne o modificarle in un tipo di dati supportato, utilizza il seguente codice SQL per ciascun database.

```
SELECT n.nspname, c.relname, a.attname
    FROM pg_catalog.pg_class c,
    pg_catalog.pg_namespace n,
    pg_catalog.pg_attribute a
    WHERE c.oid = a.attrelid AND NOT a.attisdropped AND
    a.atttypid = 'pg_catalog.unknown'::pg_catalog.regtype AND
    c.relkind IN ('r','m','c') AND
    c.relnamespace = n.oid AND
    n.nspname !~ '^pg_temp_' AND
    n.nspname !~ '^pg_toast_temp_' AND n.nspname NOT IN ('pg_catalog',
 'information_schema');
```

10 Esegui un aggiornamento di prova.

Si consiglia di testare un aggiornamento della versione principale su un duplicato del database di produzione prima di eseguire l'aggiornamento sul database di produzione. È possibile monitorare i piani di esecuzione sull'istanza di test duplicata per eventuali regressioni del piano di esecuzione e valutarne le prestazioni. Per creare un'istanza di test duplicata, è possibile ripristinare il proprio database da uno snapshot recente o clonare il database. Per ulteriori informazioni, consulta Ripristino da uno snapshot o Clonazione di un volume per un cluster di database Amazon Aurora.

Per ulteriori informazioni, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.

11 Aggiornare la propria istanza di produzione.

Dopo aver testato correttamente l'aggiornamento a una versione principale, dovrebbe essere possibile aggiornare il database di produzione senza problemi. Per ulteriori informazioni, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale.

> **Note**
>
> Durante il processo di aggiornamento, Aurora PostgreSQL acquisisce uno snapshot cluster DB se il periodo di conservazione del backup del cluster è maggiore di 0. Non è possibile eseguire un point-in-time ripristino del cluster durante questo processo. Successivamente, puoi eseguire un point-in-time ripristino ai tempi precedenti all'inizio dell'aggiornamento e dopo il completamento dello snapshot automatico dell'istanza.

Tuttavia, non è possibile eseguire il point-in-time ripristino di una versione secondaria precedente.

Per informazioni su un aggiornamento in corso, è possibile utilizzare Amazon RDS per visualizzare due log prodotti dall'utilità. Questi sono `pg_upgrade_internal.log` e `pg_upgrade_server.log`. Amazon Aurora RDS accoda un timestamp al nome file per questi log. Puoi visualizzare questi log come qualsiasi altro log. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Aurora.

12 Aggiornare le estensioni PostgreSQL. Un processo di aggiornamento PostgreSQL non aggiorna alcuna estensione PostgreSQL. Per ulteriori informazioni, consulta Aggiornamento estensioni PostgreSQL.

Consigli per la fase successiva all'aggiornamento

Dopo aver completato un aggiornamento della versione principale, è consigliabile quanto segue:

- Eseguire l'operazione `ANALYZE` per aggiornare la tabella `pg_statistic`. È necessario farlo per ogni database su tutte le istanze database di PostgreSQL. Le statistiche di ottimizzazione non vengono trasferite durante un aggiornamento della versione principale, quindi è necessario rigenerare tutte le statistiche per evitare problemi di prestazioni. Esegui il comando senza parametri per generare statistiche per tutte le tabelle regolari del database corrente, come segue:

```
ANALYZE VERBOSE;
```

Il flag `VERBOSE` è facoltativo, ma usandolo viene mostrato lo stato di avanzamento. Per ulteriori informazioni, consulta ANALYZE nella documentazione di PostgreSQL.

> **Note**
>
> Esegui ANALYZE sul tuo sistema dopo l'aggiornamento per evitare problemi di prestazioni.

- Se hai eseguito l'aggiornamento a PostgreSQL versione 10, esegui `REINDEX` su qualsiasi indice hash che hai. Gli indici hash sono stati modificati nella versione 10 e devono essere ricostruiti. Per individuare indici hash non validi, eseguire il seguente SQL per ogni database che contiene indici hash.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

- Ti consigliamo di testare l'applicazione sul database aggiornato con un carico di lavoro analogo per verificare che tutto funzioni come previsto. Dopo la verifica dell'aggiornamento è possibile eliminare l'istanza di test.

Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale

Quando avvii il processo di aggiornamento a una nuova versione principale, Aurora PostgreSQL acquisisce uno snapshot del cluster database Aurora prima di apportare eventuali modifiche al cluster. Questo snapshot viene creato solo per gli aggiornamenti della versione principale, non per gli aggiornamenti della versione secondaria. Al termine del processo di aggiornamento, questo snapshot è disponibile tra gli snapshot manuali elencati in Snapshots nella console RDS. Il nome dello snapshot include `preupgrade` come prefisso, il nome del cluster database Aurora PostgreSQL, la versione di origine, la versione di destinazione e la data e il timestamp, come illustrato nell'esempio seguente.

```
preupgrade-docs-lab-apg-global-db-12-8-to-13-6-2022-05-19-00-19
```

Al termine dell'aggiornamento, puoi utilizzare lo snapshot creato e archiviato da Aurora nell'elenco di snapshot manuali per ripristinare il cluster database alla versione precedente, se necessario.

> **Tip**
>
> In generale, gli snapshot forniscono molti modi per ripristinare il cluster database Aurora in momenti diversi. Per ulteriori informazioni, consultare Ripristino da uno snapshot cluster database e Ripristino di un cluster di database a un determinato momento. Tuttavia, Aurora PostgreSQL non supporta l'utilizzo di uno snapshot per il ripristino a una versione secondaria precedente.

Durante il processo di aggiornamento della versione principale, Aurora alloca un volume e clona il cluster database Aurora PostgreSQL di origine. Se l'aggiornamento non va a buon fine per qualsiasi motivo, Aurora PostgreSQL utilizza il clone per eseguire il rollback dell'aggiornamento. Quando vengono allocati più di 15 cloni di un volume di origine, i cloni successivi diventano copie complete e richiedono più tempo. Di conseguenza anche il processo di aggiornamento richiede più tempo. Se Aurora PostgreSQL esegue il rollback dell'aggiornamento, tenere presente quanto segue:

- È possibile che vengano visualizzate voci di fatturazione e parametri per il volume originale e per il volume clonato allocati durante l'aggiornamento. Aurora PostgreSQL elimina i dati sul volume aggiuntivo quando la finestra di conservazione del backup del cluster supera il termine dell'aggiornamento.
- La successiva copia snapshot tra regioni da questo cluster sarà una copia completa anziché una copia incrementale.

Per aggiornare in modo sicuro le istanze database che compongono il cluster, Aurora PostgreSQL utilizza l'utilità pg_upgrade. Al termine dell'aggiornamento dell'istanza di scrittura, ogni istanza di lettura riscontra una breve interruzione mentre viene aggiornato alla nuova versione principale. Per ulteriori informazioni su questa utilità, consulta pg_upgrade nella documentazione di PostgreSQL.

È possibile aggiornare il cluster Aurora PostgreSQL DB a una nuova versione utilizzando l'API, the AWS Management Console o RDS. AWS CLI

Console

Modificare la versione del motore di un cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) quindi selezionare il cluster di database da aggiornare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. In Engine version (Versione motore) scegliere la nuova versione.
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente). In alcuni casi, la chiusura di questa opzione può causare un'interruzione. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify Cluster (Modifica cluster) per salvare le modifiche.

   Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per aggiornare la versione del motore di un cluster DB, usa il modify-db-cluster AWS CLI comando. Specifica i seguenti parametri:

- `--db-cluster-identifier`: il nome del cluster di database.
- `--engine-version` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni valide del motore, utilizzate il AWS CLI describe-db-engine-versionscomando.
- `--allow-major-version-upgrade` – Un flag obbligatorio quando il parametro `--engine-version` è una versione principale diversa rispetto alla versione principale corrente del cluster database.
- `--no-apply-immediately` – Applica le modifiche durante la finestra di manutenzione successiva. Per applicare immediatamente le modifiche utilizzare `--apply-immediately`.

Example

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API RDS

Per aggiornare la versione del motore di un cluster DB, utilizzare l'DBClusteroperazione Modifica. Specifica i seguenti parametri:

- `DBClusterIdentifier` – Nome del cluster database, ad esempio *`mydbcluster`*.
- `EngineVersion` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni valide del motore, utilizzate l'operazione Descrivi DBEngine le versioni.
- `AllowMajorVersionUpgrade` – Un flag obbligatorio quando il parametro `EngineVersion` è una versione principale diversa rispetto alla versione principale corrente del cluster database.
- `ApplyImmediately` – Indica se applicare le modifiche immediatamente o durante la finestra di manutenzione successiva. Per applicare le modifiche immediatamente, imposta il valore su `true`. Per applicare le modifiche durante la finestra di manutenzione successiva imposta il valore su `false`.

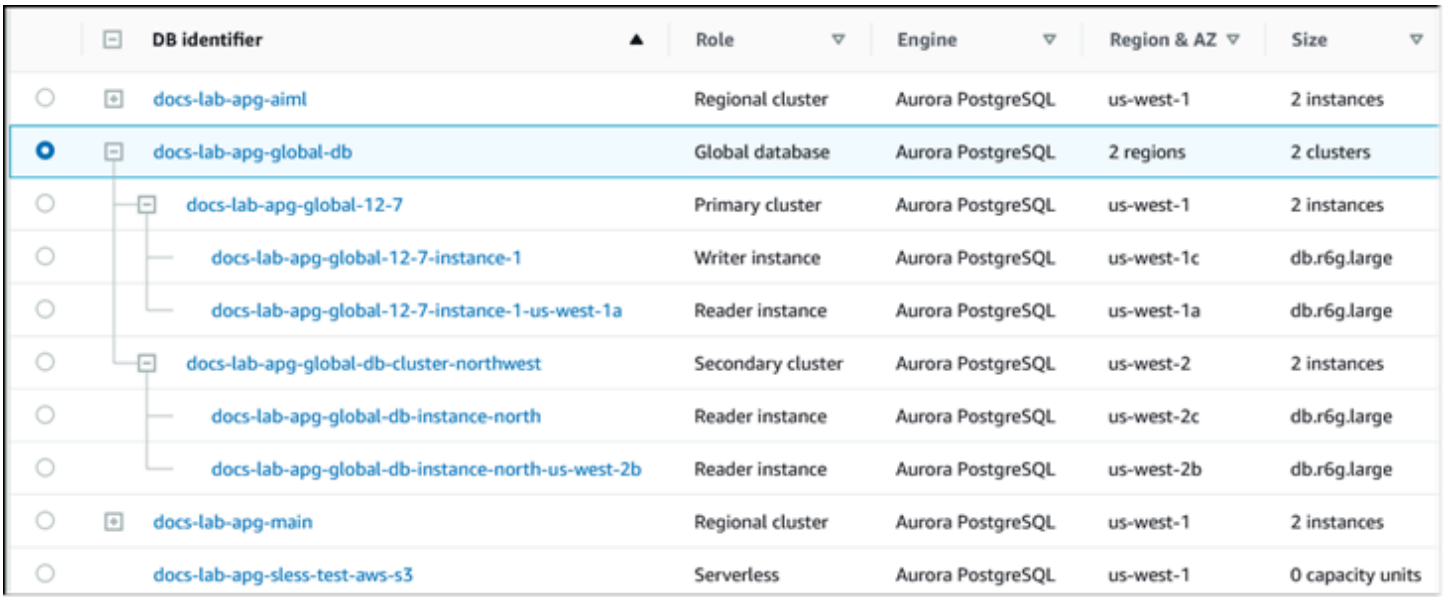
Principali aggiornamenti per database globali

Per un cluster database globale Aurora, il processo di aggiornamento aggiorna tutti i cluster database che compongono contemporaneamente il database globale Aurora. Esegue questa operazione per garantire che ognuno esegua la stessa versione di Aurora PostgreSQL. Inoltre, assicura che le eventuali modifiche apportate alle tabelle di sistema, ai formati di file di dati e così via vengono replicate automaticamente in tutti i cluster secondari.

Per aggiornare un cluster database globale a una nuova versione principale di Aurora PostgreSQL, si consiglia di eseguire il test delle applicazioni nella versione aggiornata, come descritto in Test di un aggiornamento del cluster database di produzione a una nuova versione principale. Assicurati di preparare le impostazioni del gruppo di parametri del cluster DB e del gruppo di parametri DB per ciascuno Regione AWS nel database globale Aurora prima dell'aggiornamento, come descritto instep 1. . Test di un aggiornamento del cluster database di produzione a una nuova versione principale

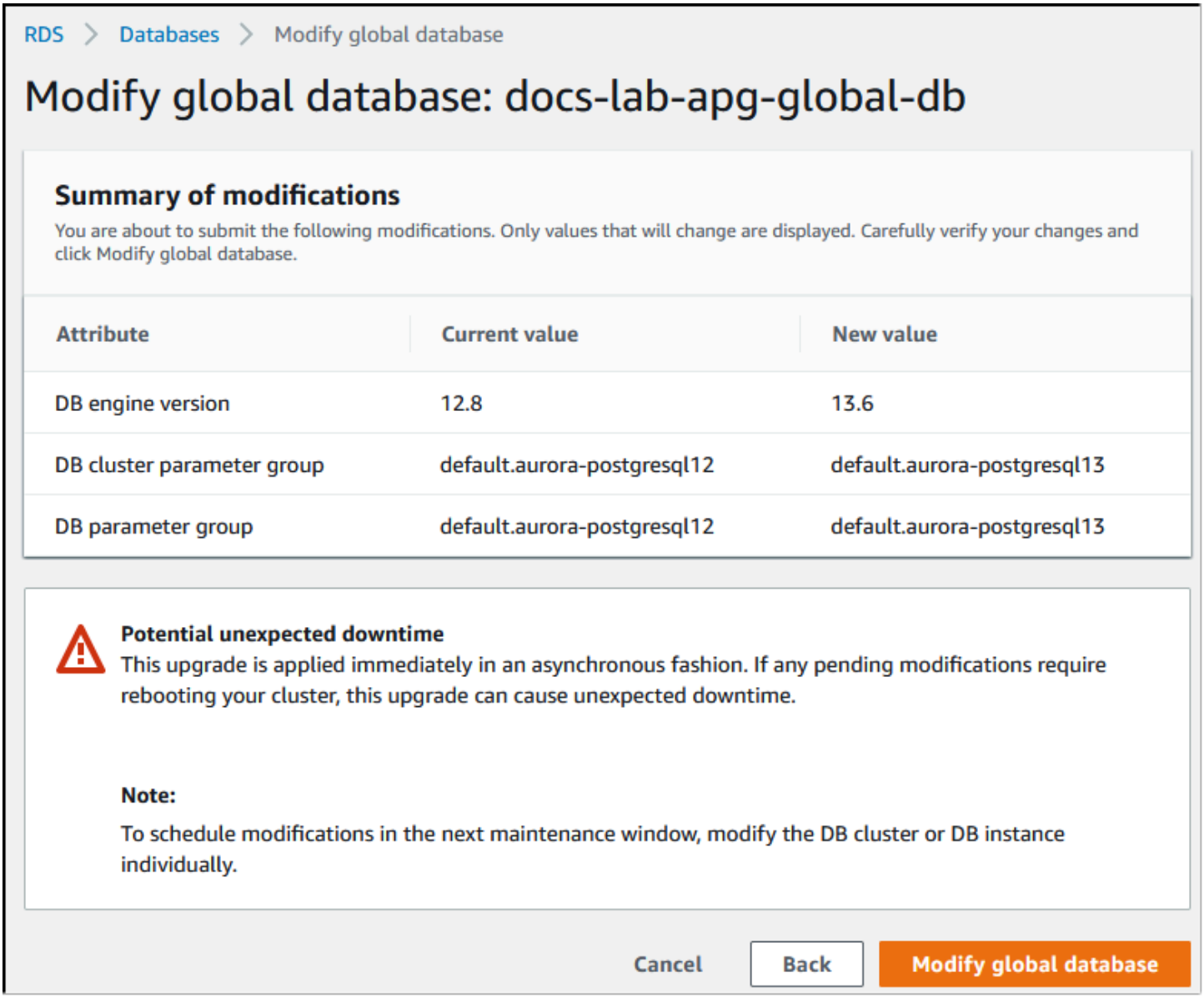
Se il cluster database globale Aurora PostgreSQL dispone di un Obiettivo del punto di ripristino (RPO) impostato per il parametro `rds.global_db_rpo`, assicurati di ripristinare il parametro prima dell'aggiornamento. Il processo di aggiornamento della versione principale non funziona se l'RPO è attivato. Per impostazione predefinita, questo parametro è disattivato. Per ulteriori informazioni su database globali Aurora PostgreSQL e RPO, consulta Gestione RPOs di database globali basati su Aurora PostgreSQL.

Se verifichi che le applicazioni possano essere eseguite come previsto durante l'implementazione di prova della nuova versione, puoi avviare il processo di aggiornamento. A questo proposito, consulta Aggiornamento del motore Aurora PostgreSQL a una nuova versione principale. Assicurati di scegliere la voce di primo livello dall'elenco Databases (Database) nella console RDS, Database globale, come mostrato nell'immagine seguente.



DB identifier	Role	Engine	Region & AZ	Size
docs-lab-apg-aiml	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-db	Global database	Aurora PostgreSQL	2 regions	2 clusters
docs-lab-apg-global-12-7	Primary cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-12-7-instance-1	Writer instance	Aurora PostgreSQL	us-west-1c	db.r6g.large
docs-lab-apg-global-12-7-instance-1-us-west-1a	Reader instance	Aurora PostgreSQL	us-west-1a	db.r6g.large
docs-lab-apg-global-db-cluster-northwest	Secondary cluster	Aurora PostgreSQL	us-west-2	2 instances
docs-lab-apg-global-db-instance-north	Reader instance	Aurora PostgreSQL	us-west-2c	db.r6g.large
docs-lab-apg-global-db-instance-north-us-west-2b	Reader instance	Aurora PostgreSQL	us-west-2b	db.r6g.large
docs-lab-apg-main	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-sless-test-aws-s3	Serverless	Aurora PostgreSQL	us-west-1	0 capacity units

Come per qualsiasi modifica, puoi confermare che desideri che il processo proceda quando richiesto.



Invece di utilizzare la console, puoi avviare il processo di aggiornamento utilizzando la AWS CLI o l'API RDS. Come per la console, si opera sul cluster database globale Aurora anziché su uno qualsiasi dei suoi componenti, come segue:

- Usa il modify-global-cluster AWS CLI comando per avviare l'aggiornamento del tuo database globale Aurora utilizzando. AWS CLI
- Utilizza l'ModifyGlobalClusterAPI per avviare l'aggiornamento.

Esecuzione di un aggiornamento di versione minore

Puoi utilizzare i seguenti metodi per aggiornare la versione secondaria di un cluster di database o per applicare le patch a un cluster di database:

Argomenti

- Prima di eseguire un aggiornamento di una versione secondaria
- Come eseguire aggiornamenti della versione secondaria e applicare patch
- Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività
- Limitazioni dell'applicazione di patch senza tempi di inattività
- Aggiornamento del motore Aurora PostgreSQL a una nuova versione secondaria

Prima di eseguire un aggiornamento di una versione secondaria

Si consiglia di eseguire le seguenti azioni per ridurre i tempi di inattività durante l'aggiornamento di una versione secondaria:

- La manutenzione del cluster Aurora DB deve essere eseguita durante un periodo di traffico ridotto. Usa Performance Insights per identificare questi periodi di tempo al fine di configurare correttamente le finestre di manutenzione. Per ulteriori informazioni su Performance Insights, consulta Monitoraggio del carico del DB con Performance Insights su Amazon RDS. Per ulteriori informazioni sulla finestra di manutenzione del cluster DB,Impostazione della finestra di manutenzione preferita del cluster database.
- Utilizzate AWS SDKs questo supporto per il backoff e il jitter esponenziali come best practice. Per ulteriori informazioni, consulta Exponential Backoff And Jitter.

Come eseguire aggiornamenti della versione secondaria e applicare patch

Gli aggiornamenti e le patch delle versioni minori sono disponibili solo dopo test rigorosi. Regioni AWS Prima di rilasciare aggiornamenti e patch, Aurora PostgreSQL esegue dei test per garantire che i problemi di sicurezza noti, i bug e altri problemi che emergono dopo il rilascio della versione comunitaria minore non compromettano la stabilità della flotta di Aurora PostgreSQL.

Se attivi Abilita l'aggiornamento automatico della versione secondaria, Aurora PostgreSQL aggiorna periodicamente il cluster DB durante la finestra di manutenzione specificata. Assicurati che l'opzione Abilita l'aggiornamento automatico della versione secondaria sia attivata per tutte le istanze del

cluster Aurora PostgreSQL DB. Per informazioni su come impostare l'aggiornamento automatico delle versioni secondarie e su come funziona l'impostazione quando viene applicata a livello di cluster e istanza, consulta Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora.

Controlla il valore dell'opzione Abilita l'aggiornamento automatico della versione secondaria per tutti i cluster Aurora PostgreSQL DB utilizzando il comando seguente. describe-db-instances AWS CLI

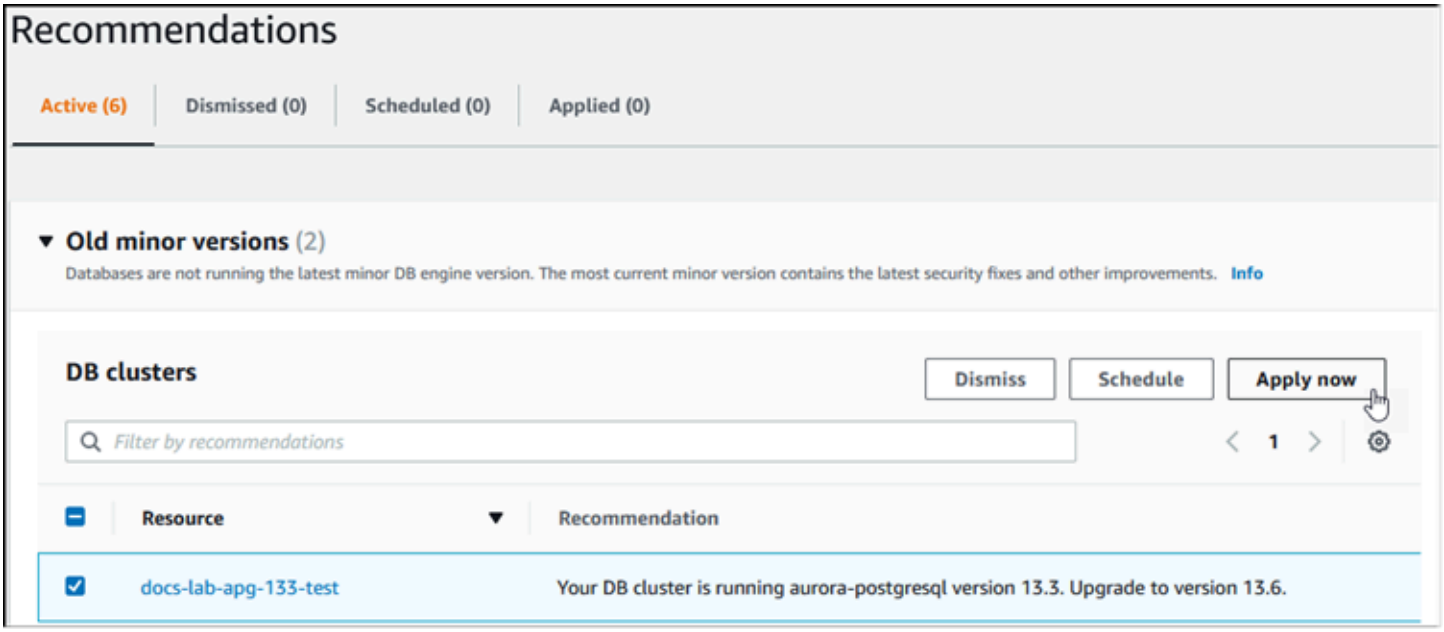
```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Questa query restituisce un elenco di tutti i cluster database Aurora e delle relative istanze con un valore `true` o `false` per lo stato dell'impostazione `AutoMinorVersionUpgrade`. Il comando presuppone che tu abbia configurato come target AWS CLI quello predefinito. Regione AWS

Per ulteriori informazioni sull'opzione Aggiornamento automatico delle versioni minori (AmVU) e su come modificare il cluster database Aurora per utilizzarla, consulta Aggiornamenti automatici delle versioni secondarie per cluster DB Aurora.

Puoi aggiornare i cluster database Aurora PostgreSQL alle nuove versioni secondarie rispondendo alle attività di manutenzione o modificando il cluster per utilizzare la nuova versione.

Puoi identificare eventuali aggiornamenti o patch disponibili per i cluster database Aurora PostgreSQL utilizzando la console RDS e aprendo il menu Recommendations (Consigli). Qui è disponibile un elenco dei diversi problemi di manutenzione come Old minor versions (Versioni secondarie precedenti). A seconda dell'ambiente di produzione, puoi scegliere di pianificare l'aggiornamento o eseguire un'azione immediata, scegliendo Apply now (Applica ora) come mostrato di seguito.



Per ulteriori informazioni su come gestire un cluster database Aurora, incluso come applicare manualmente le patch e gli aggiornamenti della versione secondaria, consulta Manutenzione di un cluster database Amazon Aurora.

Aggiornamenti della versione secondaria e applicazione di patch senza tempi di inattività

L'aggiornamento di un cluster database Aurora PostgreSQL comporta la possibilità di un'interruzione. Durante il processo di aggiornamento, il database viene arrestato. Se si avvia l'aggiornamento mentre il database è occupato, si perdono tutte le connessioni e le transazioni elaborate dal cluster database. Se si aspetta che il database sia inattivo per eseguire l'aggiornamento, potrebbe essere necessario attendere a lungo.

La funzione ZDP (applicazione delle patch senza tempi di inattività) migliora il processo di aggiornamento. Grazie a ZDP, è possibile applicare aggiornamenti della versione secondaria e patch con un impatto minimo sul cluster database Aurora PostgreSQL. ZDP viene utilizzato quando si applicano patch o aggiornamenti di versioni secondarie più recenti a versioni di Aurora PostgreSQL e altre versioni successive di queste versioni secondarie e le versioni principali più recenti. Ovvero, l'aggiornamento a nuove versioni secondarie da una qualsiasi di queste versioni in poi utilizza ZDP.

La tabella seguente mostra le versioni di Aurora PostgreSQL e le classi di istanze database in cui è disponibile ZDP:

Versione	Classi di istanza db.r*	Classi di istanza db.t*	Classi di istanza db.x*	Classe di istanza db.serverless
10.21 e versioni successive	Sì	Sì	Sì	N/D
11.16 e versioni successive	Sì	Sì	Sì	N/D
11.17 e versioni successive	Sì	Sì	Sì	N/D
12.11 e versioni successive	Sì	Sì	Sì	N/D
12.12 e versioni successive	Sì	Sì	Sì	N/D
13.7 e versioni successive	Sì	Sì	Sì	N/D
13.8 e versioni successive	Sì	Sì	Sì	Sì
14.3 e versioni successive	Sì	Sì	Sì	N/D
14.4 e versioni successive	Sì	Sì	Sì	N/D
14.5 e versioni successive	Sì	Sì	Sì	Sì
15.3 e versioni successive	Sì	Sì	Sì	Sì
16.1 e versioni successive	Sì	Sì	Sì	Sì

Durante il processo di aggiornamento tramite l'applicazione di patch senza tempi di inattività, il motore di database cerca un punto idoneo per sospendere tutte le nuove transazioni. Questa azione protegge il database durante le patch e gli aggiornamenti. Per assicurarsi che le applicazioni funzionino senza problemi con transazioni sospese, è consigliabile integrare la logica dell'esecuzione di nuovi tentativi nel codice. Questo approccio garantisce che il sistema sia in grado di gestire qualsiasi breve periodo di inattività senza errori e di riprovare le nuove transazioni dopo l'aggiornamento.

Quando l'applicazione di patch senza tempi di inattività è completata, le sessioni dell'applicazione vengono conservate, ad eccezione di quelle con connessioni eliminate; il motore di database viene riavviato mentre l'aggiornamento è ancora in corso. Il riavvio del motore di database può causare una riduzione temporanea della velocità di trasmissione effettiva, che dura in genere alcuni secondi o al massimo circa un minuto.

In alcuni casi, l'applicazione di patch senza tempi di inattività (ZDP) potrebbe non avere esito positivo. Ad esempio, le modifiche ai parametri che si trovano nello stato `pending` sul cluster di database Aurora PostgreSQL o le sue istanze possono interferire con l'applicazione di patch senza tempi di inattività.

Puoi trovare parametri ed eventi per le operazioni ZDP nella pagina Events (Eventi) nella console. Gli eventi includono l'avvio dell'aggiornamento ZDP e il completamento dell'aggiornamento. In questo caso puoi individuare il tempo richiesto dal processo e il numero di connessioni conservate e interrotte che si sono verificate durante il riavvio. Puoi individuare i dettagli nel registro degli errori del database.

Limitazioni dell'applicazione di patch senza tempi di inattività

Le seguenti limitazioni si applicano alle patch senza tempi di inattività:

- ZDP cerca di preservare le connessioni client correnti all'istanza di Aurora PostgreSQL writer durante tutto il processo di aggiornamento di Aurora PostgreSQL. Tuttavia, nei seguenti casi, le connessioni verranno interrotte per l'applicazione di patch senza tempi di inattività:
  - Sono in esecuzione query o transazioni di lunga durata.
  - Sono in esecuzione istruzioni DDL (Data Definition Language).
  - Si utilizzando blocchi di tabelle o tabelle temporanee.
  - Vengono ascoltate tutte le sessioni sui canali di notifica.
  - È in uso un cursore nello stato "WITH HOLD".

  - TLSv1.1 le connessioni sono in uso. A partire dalle versioni di Aurora PostgreSQL successive alla 16.1, 15.3, 14.8, 13.11, 12.15 e 11.20, ZDP è supportato con connessioni 1.3. TLSv1
- ZDP non è supportato nei seguenti casi:
  - Quando i cluster Aurora PostgreSQL DB sono configurati come Aurora Serverless v1.
  - Durante l'aggiornamento di qualsiasi istanza del lettore Aurora.
  - Durante l'aggiornamento di qualsiasi istanza di lettore Aurora che fa parte di un cluster Aurora Global Database in una regione secondaria.
  - Durante le patch e gli aggiornamenti del sistema operativo.

Aggiornamento del motore Aurora PostgreSQL a una nuova versione secondaria

Puoi aggiornare il cluster Aurora PostgreSQL DB a una nuova versione secondaria utilizzando la console, l'o l'API RDS. AWS CLI Prima di eseguire l'aggiornamento, si consiglia di seguire le stesse best practice consigliate per gli aggiornamenti della versione principale. Come per le nuove versioni principali, anche le nuove versioni secondarie possono contenere miglioramenti all'ottimizzatore, ad esempio correzioni, che possono causare regressioni del piano di query. Per garantire la stabilità del piano, si consiglia di utilizzare l'estensione Query Plan Management (QPM) come descritto in Garantire la stabilità del piano dopo un aggiornamento di versione principale.

Console

Aggiornare la versione del motore del cluster database Aurora PostgreSQL

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) quindi selezionare il cluster di database da aggiornare.
3. Scegliere Modify (Modifica). Viene visualizzata la pagina Modify DB cluster (Modifica cluster di database).
4. In Engine version (Versione motore) scegliere la nuova versione.
5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente). In alcuni casi, la chiusura di questa opzione può causare un'interruzione. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora.

7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, selezionare Modify Cluster (Modifica cluster) per salvare le modifiche.

   Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per aggiornare la versione del motore di un cluster DB, usa il modify-db-cluster AWS CLI comando con i seguenti parametri:

- `--db-cluster-identifier` – Nome del cluster database Aurora PostgreSQL.
- `--engine-version` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni valide del motore, utilizzate il AWS CLI describe-db-engine-versionscomando.
- `--no-apply-immediately` – Applica le modifiche durante la finestra di manutenzione successiva. Per applicare immediatamente le modifiche, utilizza invece `--apply-immediately`.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --no-apply-immediately
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --no-apply-immediately
```

API RDS

Per aggiornare la versione del motore di un cluster DB, utilizzare l'DBClusteroperazione Modifica. Specifica i seguenti parametri:

- `DBClusterIdentifier` – Nome del cluster database, ad esempio *`mydbcluster`*.

- `EngineVersion` – Numero di versione del motore di database a cui effettuare l'aggiornamento. Per informazioni sulle versioni valide del motore, utilizzate l'operazione Descrivi DBEngine le versioni.
- `ApplyImmediately` – Indica se applicare le modifiche immediatamente o durante la finestra di manutenzione successiva. Per applicare le modifiche immediatamente, imposta il valore su `true`. Per applicare le modifiche durante la finestra di manutenzione successiva imposta il valore su `false`.

Aggiornamento estensioni PostgreSQL

L'aggiornamento del cluster di database Aurora PostgreSQL a una nuova versione principale o secondaria non aggiorna contemporaneamente le estensioni PostgreSQL. Per la maggior parte delle estensioni, l'estensione viene aggiornata dopo il completamento dell'aggiornamento della versione principale o secondaria. Tuttavia, in alcuni casi, l'estensione viene aggiornata prima di aggiornare il motore di database Aurora PostgreSQL. Per ulteriori informazioni, consulta list of extensions to update in Test di un aggiornamento del cluster database di produzione a una nuova versione principale.

L'installazione delle estensioni PostgreSQL richiede privilegi `rds_superuser`. In genere, un `rds_superuser` delega le autorizzazioni su estensioni specifiche agli utenti (ruoli) pertinenti, per facilitare la gestione di una determinata estensione. Ciò significa che il compito di aggiornare tutte le estensioni nel cluster database Aurora PostgreSQL potrebbe coinvolgere molti utenti (ruoli) diversi. Tenerlo presente se si desidera automatizzare il processo di aggiornamento utilizzando gli script. Per ulteriori informazioni sui privilegi e sui ruoli PostgreSQL, consulta Sicurezza con Amazon Aurora PostgreSQL.

> **Note**
>
> Per informazioni sull'aggiornamento dell'estensione PostGIS, consulta Gestione dei dati spaziali con l'estensione Post GIS (Passaggio 6: Aggiorna l'GISestensione Post).
> Per aggiornare l'estensione `pg_repack`, rimuovi l'estensione e quindi crea la nuova versione nell'istanza database aggiornata. Per ulteriori informazioni, consulta pg_repack installation nella documentazione `pg_repack`.

Per aggiornare un'estensione dopo un aggiornamento del motore, utilizza il comando `ALTER EXTENSION UPDATE`.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

Per elencare le estensioni attualmente installate, usa il catalogo PostgreSQL pg_extension nel seguente comando.

```
SELECT * FROM pg_extension;
```

Per visualizzare l'elenco delle versioni delle estensioni specifiche disponibili per l'installazione, utilizza la vista PostgreSQL pg_available_extension_versions nel seguente comando.

```
SELECT * FROM pg_available_extension_versions;
```

Tecnica alternativa di aggiornamento blu/verde

In alcune situazioni, la priorità principale è eseguire il passaggio immediato dal cluster precedente a quello aggiornato. In tali situazioni, è possibile utilizzare un processo in più fasi che esegue i cluster vecchi e nuovi. side-by-side Qui, i dati vengono replicati dal cluster precedente a quello nuovo fino a quando il nuovo cluster non prende il controllo. Per informazioni dettagliate, consulta Utilizzo delle implementazioni blu/verde per gli aggiornamenti del database.

Utilizzo di una versione di Aurora PostgreSQL con supporto a lungo termine (LTS)

Ogni nuova versione di Aurora PostgreSQL resta disponibile per l'utilizzo per un certo periodo di tempo quando crei o aggiorni un cluster di database. Trascorso questo periodo, devi aggiornare i cluster che utilizzano questa versione. Puoi aggiornare manualmente il cluster prima della scadenza del periodo di supporto oppure lasciare che Aurora lo faccia al posto tuo quando la versione di Aurora PostgreSQL non è più supportata.

Aurora indica alcune versioni di Aurora PostgreSQL come versioni con supporto a lungo termine (LTS). I cluster di database che utilizzano le versioni LTS possono restare più a lungo con la stessa versione ed essere sottoposti a un numero minore di cicli di aggiornamento rispetto ai cluster che utilizzano versioni non LTS. Le versioni secondarie LTS includono solo correzioni di bug (attraverso versioni patch); una versione LTS non include nuove funzionalità rilasciate dopo la sua introduzione.

Una volta all'anno, ai cluster di database in esecuzione su una versione secondaria LTS viene assegnata una patch con versione più recente della versione LTS. Questa applicazione della patch

viene eseguita per garantire che si tragga vantaggio dalle correzioni cumulative per la sicurezza e la stabilità. Potremmo applicare patch a una versione secondaria di LTS più frequentemente nel caso in cui fossero presenti correzioni critiche, ad esempio per la sicurezza, che devono essere applicate.

> **Note**
>
> Se desideri rimanere su una versione secondaria LTS per tutta la durata del suo ciclo di vita, assicurati di disabilitare l'aggiornamento automatico della versione secondaria per le tue istanze DB. Per evitare l'aggiornamento automatico del cluster DB dalla versione secondaria LTS, deseleziona la casella di controllo Abilita aggiornamento automatico della versione secondaria su qualsiasi istanza DB del cluster Aurora.

Per la maggior parte dei cluster Aurora PostgreSQL, consigliamo di eseguire l'aggiornamento all'ultima versione invece di utilizzare la versione LTS. In questo modo, puoi sfruttare Aurora anche come servizio gestito e puoi accedere alle ultime funzionalità e correzioni di bug. Le versioni LTS sono destinate a cluster con le seguenti caratteristiche:

- L'applicazione Aurora PostgreSQL non può avere tempi di inattività dovuti agli aggiornamenti, ad eccezione di rari casi per la distribuzione di patch critiche.
- Il ciclo di test per il cluster e le applicazioni associate richiede molto tempo per ogni aggiornamento al motore del database di Aurora PostgreSQL.
- La versione del database per il cluster Aurora PostgreSQL ha tutte le funzionalità del motore del database e le correzioni di bug necessarie per l'applicazione.

Le versioni LTS correnti per Aurora PostgreSQL sono le seguenti:

- PostgreSQL 15.10. È stato rilasciato il 27 dicembre 2024. Per ulteriori informazioni, vedere PostgreSQL 15.10 nelle Note di rilascio per Aurora PostgreSQL.
- PostgreSQL 14.6. È stato rilasciato il 20 gennaio 2023. Per ulteriori informazioni, consulta PostgreSQL 14.6 nelle Note di rilascio di Aurora PostgreSQL.
- PostgreSQL 13.9. È stato rilasciato il 20 gennaio 2023. Per ulteriori informazioni, consulta PostgreSQL 13.9 nelle Note di rilascio di Aurora PostgreSQL.
- PostgreSQL 12.9. È stato rilasciato il 25 febbraio 2022. Per ulteriori informazioni, consultare PostgreSQL 12.9 nelle Notedi rilascio di Aurora PostgreSQL.

- PostgreSQL 11.9 (Aurora PostgreSQL versione 3.4). È stato rilasciato l'11 dicembre 2020. Per ulteriori informazioni su questa versione, consultare PostgreSQL 11.9, Aurora PostgreSQL versione 3.4 nelle Note di rilascio di Aurora PostgreSQL.

Per dettagli sulle tempistiche di supporto e sui cicli di rilascio per le versioni LTS, consulta Calendari di rilascio per Aurora PostgreSQL.

Per informazioni su come identificare le versioni di Aurora e del motore del database, consulta Identificazione delle versioni di Amazon Aurora Postgre SQL.

Utilizzo del database Amazon Aurora SQL Postgre Limitless

Amazon Aurora Postgre SQL Limitless Database offre una scalabilità orizzontale automatizzata per elaborare milioni di transazioni di scrittura al secondo e gestisce petabyte di dati mantenendo la semplicità di funzionamento all'interno di un singolo database. Con Aurora Postgre SQL Limitless Database, puoi concentrarti sulla creazione di applicazioni su larga scala senza dover creare e mantenere soluzioni complesse per scalare i tuoi dati su più istanze DB per supportare i tuoi carichi di lavoro.

Argomenti

- Architettura di database Aurora PostgreSQL Limitless
- Guida introduttiva al database Aurora SQL Postgre Limitless
- Requisiti e considerazioni del database Aurora PostgreSQL Limitless
- Prerequisiti per l'utilizzo del database SQL Aurora Postgre Limitless
- Creazione di un cluster DB che utilizza Aurora SQL Postgre Limitless Database
- Lavorare con gruppi di shard DB
- Creazione di tabelle del database Aurora SQL Postgre Limitless
- Caricamento dei dati nel database Aurora SQL Postgre Limitless
- Interrogazione del database Aurora SQL Postgre Limitless
- Gestione del database Aurora SQL Postgre Limitless
- Monitoraggio del database Aurora PostgreSQL Limitless
- Backup e ripristino del database Aurora Postgre Limitless SQL
- Aggiornamento del database Amazon Aurora PostgreSQL Limitless
- Riferimento al database Aurora SQL Postgre Limitless

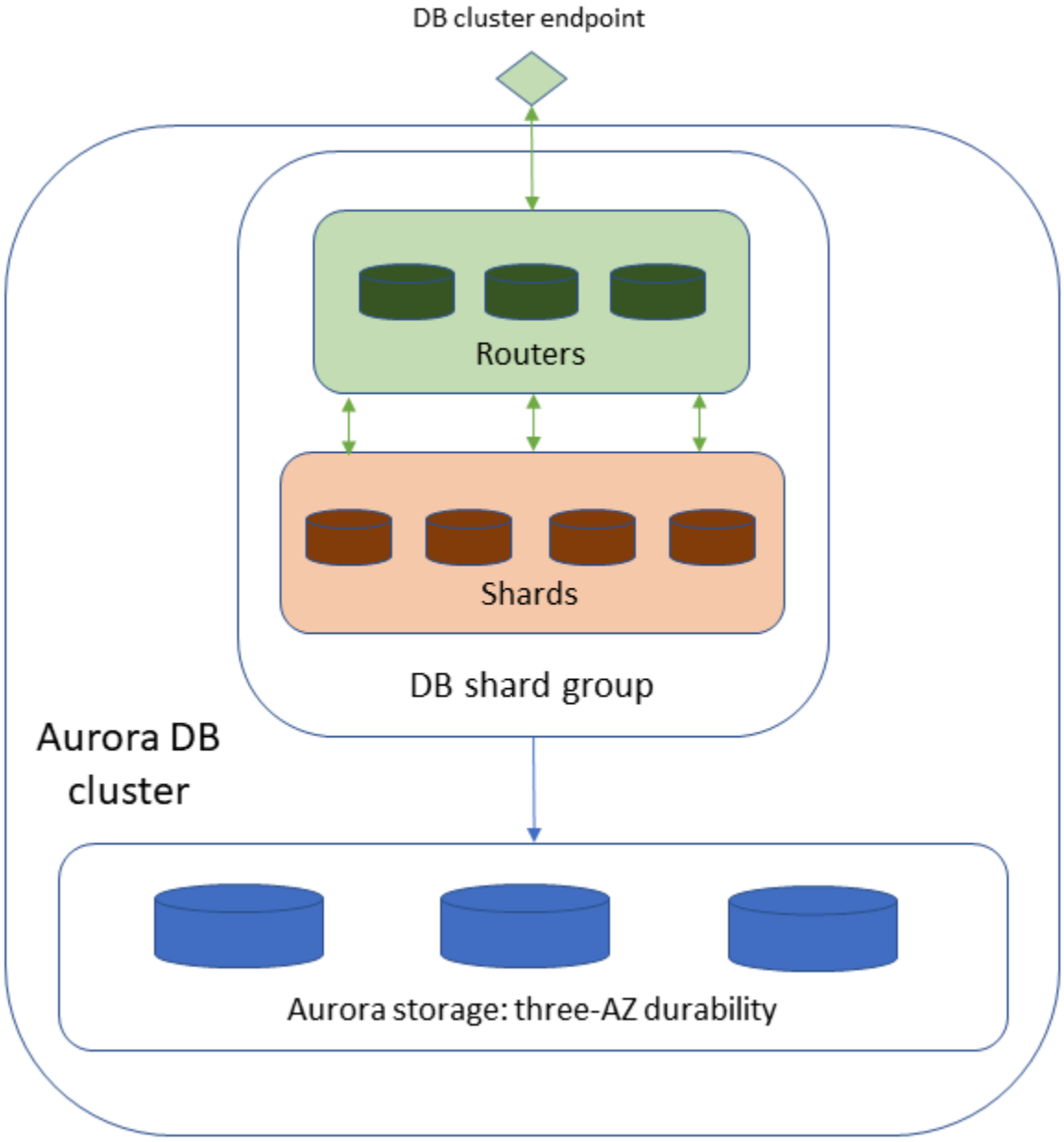
Architettura di database Aurora PostgreSQL Limitless

Limitless Database raggiunge la scalabilità con un'architettura a due livelli composta da più nodi di database. I nodi sono router o shard.

- Gli shard sono istanze DB di Aurora PostgreSQL, ognuna delle quali archivia un sottoinsieme di dati per il database, consentendo l'elaborazione simultanea per ottenere un throughput di scrittura più elevato.
- I router gestiscono la natura distribuita del database e presentano un'unica immagine del database ai client del database. I router conservano i metadati relativi alla posizione in cui sono archiviati i dati, analizzano i comandi SQL in entrata e li inviano agli shard. Quindi aggregano i dati dagli shard per restituire un singolo risultato al cliente e gestiscono le transazioni distribuite per mantenere la coerenza nell'intero database distribuito.

Aurora PostgreSQL Limitless Database si differenzia Cluster di database Aurora dallo standard perché dispone di un gruppo di shard DB invece di un'istanza DB writer e di istanze DB reader. Tutti i nodi che compongono l'architettura Limitless Database sono contenuti nel gruppo di shard DB. I singoli shard e router del gruppo di shard DB non sono visibili nel tuo. Account AWS Si utilizza l'endpoint del cluster DB per accedere a Limitless Database.

La figura seguente mostra l'architettura di alto livello di Aurora PostgreSQL Limitless Database.



Per ulteriori informazioni sull'architettura di Aurora PostgreSQL Limitless Database e su come utilizzarlo, consulta Raggiungere la scalabilità con Amazon Aurora PostgreSQL Limitless Database sul canale Eventi su. AWS YouTube

Per ulteriori informazioni sull'architettura di un cluster Aurora DB standard, vedere. Cluster database Amazon Aurora

Termini chiave per Aurora PostgreSQL Limitless Database

Gruppo di condivisioni DB

Un contenitore per nodi Limitless Database (shard e router).

Router

Un nodo che accetta connessioni SQL dai client, invia comandi SQL agli shard, mantiene la coerenza a livello di sistema e restituisce i risultati ai client.

Shard

Un nodo che memorizza un sottoinsieme di tabelle suddivise, copie complete di tabelle di riferimento e tabelle standard. Accetta richieste dai router, ma non può essere collegato direttamente dai client.

Tavolo condiviso

Una tabella con i dati partizionati su frammenti.

Chiave condivisa

Una colonna o un insieme di colonne in una tabella condivisa utilizzata per determinare il partizionamento tra gli shard.

Tabelle collocate

Due tabelle condivise che condividono la stessa chiave shard e sono dichiarate esplicitamente come collocate. Tutti i dati per lo stesso valore della chiave shard vengono inviati allo stesso shard.

Tabella di riferimento

Una tabella con i dati copiati per intero su ogni frammento.

Tabella standard

Il tipo di tabella predefinito in Limitless Database. È possibile convertire tabelle standard in tabelle frammentate e di riferimento.

Tutte le tabelle standard sono memorizzate sullo stesso shard selezionato dal sistema, in modo da consentire l'esecuzione di join tra tabelle standard all'interno di un singolo shard. Tuttavia, le tabelle standard sono limitate dalla capacità massima dello shard (128 TiB). Questo shard memorizza anche i dati da tabelle frammentate e di riferimento, quindi il limite effettivo per le tabelle standard è inferiore a 128 TiB.

Tipi di tabelle per Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database supporta tre tipi di tabelle: frammentate, di riferimento e standard.

Le tabelle frammentate hanno i dati distribuiti su tutti gli shard del gruppo di shard DB. Limitless Database esegue questa operazione automaticamente utilizzando una chiave shard, che è una colonna o un set di colonne specificato durante il partizionamento della tabella. Tutti i dati con lo stesso valore per la chiave shard vengono inviati allo stesso shard. Lo sharding è basato su hash, non su intervalli o elenchi.

I seguenti sono buoni casi d'uso per le tabelle ripartite:

- L'applicazione funziona con un sottoinsieme distinto di dati.
- La tabella è molto grande.
- La tabella potenzialmente cresce più velocemente rispetto alle altre tabelle.

Le tabelle suddivise possono essere collocate, il che significa che condividono la stessa chiave shard e che tutti i dati di entrambe le tabelle con lo stesso valore di shard key vengono inviati allo stesso shard. Se si collocano le tabelle e le si unisce utilizzando la chiave shard, l'unione può essere eseguita su un singolo shard perché in quello shard sono presenti tutti i dati necessari.

Le tabelle di riferimento hanno una copia completa di tutti i relativi dati su ogni shard del gruppo di shard DB. Le tabelle di riferimento sono comunemente utilizzate per tabelle più piccole con un volume di scrittura inferiore, ma che devono comunque essere unite frequentemente e non si prestano allo sharding. Esempi di tabelle di riferimento includono tabelle di date e tabelle di dati geografici come stato, città e codice postale.

Le tabelle standard sono il tipo di tabella predefinito in Aurora PostgreSQL Limitless Database. Non sono tabelle distribuite. Aurora PostgreSQL Limitless Database supporta i join tra tabelle standard e tabelle standard, frammentate e di riferimento.

Fatturazione per Aurora PostgreSQL Limitless Database

Per informazioni su come ti viene addebitato il database Aurora PostgreSQL Limitless, consulta. DB per Aurora

Per informazioni sui prezzi di Aurora, consulta la pagina dei prezzi di Aurora.

Guida introduttiva al database Aurora SQL Postgre Limitless

Esegui le seguenti azioni per iniziare a usare Aurora SQL Postgre Limitless Database:

1. Crea un cluster Aurora Postgree SQL DB e un gruppo di shard DB per Limitless Database. Per ulteriori informazioni, consulta Creazione di un cluster DB che utilizza Aurora SQL Postgre Limitless Database.
2. Crea tabelle frammentate e di riferimento nel gruppo di shard DB. Per ulteriori informazioni, consulta Creazione di tabelle del database Aurora SQL Postgre Limitless.
3. Modifica la capacità del tuo gruppo di shard DB, dividi gli shard e aggiungi router. Per ulteriori informazioni, consulta Lavorare con gruppi di shard DB.
4. Carica i dati nel gruppo di shard DB. Per ulteriori informazioni, consulta Caricamento dei dati nel database Aurora SQL Postgre Limitless.
5. Esegui query e altre SQL istruzioni sul gruppo di shard DB. Per ulteriori informazioni, consulta Interrogazione del database Aurora SQL Postgre Limitless.
6. Monitora le prestazioni di Limitless Database. Per ulteriori informazioni, consulta Monitoraggio del database Aurora PostgreSQL Limitless.

Requisiti e considerazioni del database Aurora PostgreSQL Limitless

Aurora PostgreSQL Limitless Database ha i seguenti requisiti e considerazioni.

Argomenti

- Requisiti per Aurora PostgreSQL Limitless Database
- Considerazioni per Aurora PostgreSQL Limitless Database
- Funzionalità non supportate in Aurora PostgreSQL Limitless Database

Requisiti per Aurora PostgreSQL Limitless Database

Assicurati di rispettare questi requisiti per Aurora PostgreSQL Limitless Database.

- Sono disponibili i seguenti: Regioni AWS
  - Asia Pacifico (Hong Kong)
  - Asia Pacifico (Singapore)
  - Asia Pacifico (Sydney)
  - Asia Pacifico (Tokyo)
  - Europa (Francoforte)
  - Europa (Irlanda)
  - Europa (Stoccolma)
  - Stati Uniti orientali (Virginia settentrionale)

    > **Note**
    >
    > Se create il vostro cluster Limitless Database DB in questo ambiente Regione AWS, non includete `us-east-1e` la zona di disponibilità (AZ) nel sottogruppo di database. A causa delle limitazioni delle risorse, Aurora Serverless v2 e quindi Limitless Database, non è supportato nella AZ. `us-east-1e`

  - Stati Uniti orientali (Ohio)
  - US West (Oregon)

Le AWS CLI procedure descritte in questa guida presuppongono che si stia utilizzando una delle procedure disponibili. Regioni AWS Per informazioni su come impostare l'impostazione predefinita Regione AWS per la AWS CLI, vedere Come impostare le variabili di ambiente nella Guida per l'AWS Command Line Interface utente della versione 2.

- Aurora PostgreSQL Limitless Database supporta solo la configurazione di storage del cluster DB. Aurora I/O-Optimized Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.
- Aurora PostgreSQL Limitless Database utilizza versioni speciali del motore Aurora PostgreSQL DB per Aurora PostgreSQL Limitless Database:
  - `16.4-limitless`
  - `16.6-limitless`
- Il cluster DB non può avere istanze DB di tipo writer o reader.
- È necessario utilizzare Enhanced Monitoring and Performance Insights. Il tempo di conservazione di Performance Insights deve essere di almeno 1 mese (31 giorni).
- È necessario esportare il log PostgreSQL in Amazon Logs. CloudWatch

> **Note**
>
> Alcune funzionalità richieste, come Enhanced Monitoring, Performance Insights e CloudWatch Logs, sono soggette a costi aggiuntivi. Per informazioni sui prezzi di Aurora, consulta la pagina dei prezzi di Aurora.

Considerazioni per Aurora PostgreSQL Limitless Database

Le seguenti considerazioni si applicano ai gruppi di shard DB in Aurora PostgreSQL Limitless Database:

- È possibile avere un solo gruppo di shard DB per cluster DB.
- È possibile avere fino a cinque gruppi di shard DB per. Regione AWS

  Pertanto, puoi avere fino a cinque cluster Aurora PostgreSQL Limitless Database DB per ogni cluster. Regione AWS Per ulteriori informazioni, consulta Quote in Amazon Aurora.
- È possibile impostare la capacità massima di un gruppo di shard DB su 16-6144. ACUs Per limiti di capacità superiori a 6144, contattare. ACUs AWS

Il numero iniziale di router e shard è determinato dalla capacità massima impostata quando si crea un gruppo di shard DB. Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati.

- Il numero di router e shard non cambia quando si modifica la capacità massima di un gruppo di shard DB.
- Assicuratevi che la sottorete DB in cui create il gruppo di shard DB disponga di un numero sufficiente di indirizzi IP liberi per la connessione con il gruppo di shard DB. È necessario un indirizzo IP per ogni router e fino a tre indirizzi IP per ogni shard del gruppo di shard DB.

  Per ulteriori informazioni sul numero di router creati quando si crea un gruppo di shard DB, vedere. Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati
- Se rendi il tuo gruppo di shard DB accessibile al pubblico, assicurati di configurare un gateway Internet nel tuo VPC.
- Utilizzate le funzioni SQL per dividere gli shard e aggiungere router.
- L'unione degli shard non è supportata.
- Non è possibile eliminare singoli shard e router.
- Non è possibile modificare (eseguire `UPDATE` operazioni sulle) chiavi degli shard in alcun modo, inclusa la modifica dei loro valori nelle righe della tabella.

  Per modificare una chiave shard, eliminala e poi ricreala.
- Sono supportati i livelli di isolamento Repeable Read, Read Committed e Read Uncommit. Non è possibile impostare il livello di isolamento su serializzabile.
- Alcuni comandi SQL non sono supportati. Per ulteriori informazioni, consulta Riferimento al database Aurora SQL Postgre Limitless.
- Non tutte le estensioni PostgreSQL sono supportate. Per ulteriori informazioni, consulta Estensioni.
- Quando si crea un gruppo di shard o quando si aggiungono nuovi nodi di un gruppo di shard (shard o router), tali nodi vengono creati in una delle zone di disponibilità (AZs) disponibili per il cluster DB. Non puoi scegliere una AZ specifica per i singoli nodi.
- Se utilizzi una ridondanza di calcolo di 2 (due standby di calcolo per il gruppo di shard DB), assicurati che il tuo gruppo di sottoreti DB ne abbia almeno tre. AZs
- Aurora PostgreSQL Limitless Database supporta fino a 54 caratteri per i nomi di tabella condivisi.

Le seguenti considerazioni si applicano al cluster Aurora PostgreSQL Limitless Database DB:

- Si consiglia di utilizzare policy AWS gestite per limitare le autorizzazioni per il database e le applicazioni a quelle di cui i clienti hanno bisogno per i loro casi d'uso. Per ulteriori informazioni, consulta Best practice per le policy.
- Quando crei il cluster Aurora PostgreSQL Limitless Database DB, imposti solo i parametri di scalabilità per il gruppo di shard DB.
- Non è possibile arrestare o avviare un cluster DB che contiene un gruppo di shard DB.
- Se devi eliminare il tuo cluster DB, devi prima eliminare il gruppo di shard DB.
- Aurora PostgreSQL Limitless Database non può essere una fonte di replica.

Funzionalità non supportate in Aurora PostgreSQL Limitless Database

Le seguenti funzionalità di Aurora PostgreSQL non sono supportate in Aurora PostgreSQL Limitless Database:

- Autenticazione Active Directory (Kerberos)
- Amazon DevOps Guru
- Amazon ElastiCache
- Implementazioni blu/verde di Amazon RDS
- Server proxy per Amazon RDS
- Aurora Auto Scaling (aggiunta automatica di istanze di lettura al cluster DB)
- Database globale Aurora
- Aurora machine learning
- Raccomandazioni Aurora
- Aurora Serverless v1
- Integrazioni Aurora zero-ETL
- AWS Backup
- AWS Lambda integrazione
- AWS Secrets Manager
- Babelfish per Aurora PostgreSQL
- Clonazione di cluster DB
- Endpoint personalizzati
- Flussi di attività di database
- Repliche di lettura
- API di dati RDS

Prerequisiti per l'utilizzo del database SQL Aurora Postgre Limitless

Per utilizzare Aurora Postgre SQL Limitless Database, è necessario innanzitutto eseguire le seguenti attività.

Argomenti

- Abilitazione delle operazioni dei gruppi condivisi DB

Abilitazione delle operazioni dei gruppi condivisi DB

Prima di poter creare un gruppo di shard DB, è necessario abilitare le operazioni del gruppo di shard DB.

- Aggiungi la seguente sezione alla IAM politica del IAM ruolo dell'utente che accede al database Aurora SQL Postgre Limitless:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDBShardGroup",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBShardGroup",
                "rds:DescribeDBShardGroups",
                "rds:DeleteDBShardGroup",
                "rds:ModifyDBShardGroup",
                "rds:RebootDBShardGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:*:shard-group:*",
                "arn:aws:rds:*:*:cluster:*"
            ]
        }
    ]
}
```

Creazione di un cluster DB che utilizza Aurora SQL Postgre Limitless Database

È possibile creare un nuovo cluster Aurora DB utilizzando la versione Limitless Database di Aurora Postgre e aggiungere un gruppo di SQL shard DB al cluster. Quando si aggiunge un gruppo di shard DB, si specifica la capacità di calcolo massima per l'intero gruppo di shard DB (somma delle capacità per tutti i router e gli shard) nelle unità di capacità Aurora (). ACUs Ciascuno ACU è una combinazione di circa 2 gibibyte (GiB) di memoria, corrispondente e rete. CPU La scalabilità aumenta o diminuisce la capacità del gruppo di shard DB, a seconda del carico di lavoro dell'applicazione, in modo analogo a quanto segue Aurora Serverless v2funziona.

Argomenti

- Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati
- Creazione del cluster DB

Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati

Il numero iniziale di router e shard in un gruppo di shard DB è determinato dalla capacità massima impostata al momento della creazione del gruppo di shard DB. Maggiore è la capacità massima, maggiore è il numero di router e shard creati nel gruppo di shard DB.

Ogni nodo (shard o router) ha il proprio valore di capacità attuale, anch'esso misurato in. ACUs

- Limitless Database ridimensiona un nodo fino a una capacità superiore quando la sua capacità attuale è troppo bassa per gestire il carico. Tuttavia, i nodi non si espandono quando la capacità totale è al massimo.
- Limitless Database riduce il nodo a una capacità inferiore quando la sua capacità attuale è superiore a quella necessaria. Tuttavia, i nodi non si ridimensionano quando la capacità totale è al minimo.

La tabella seguente mostra la correlazione tra la capacità massima del gruppo di shard DB nelle unità di capacità Aurora (ACUs) e il numero di nodi (router e shard) creati.

> **Note**
>
> Questi valori sono soggetti a modifiche.
> Se si imposta la ridondanza di calcolo su un valore diverso da zero, il numero totale di shard verrà raddoppiato o triplicato. Ciò comporterà costi aggiuntivi.
> I nodi in standby di elaborazione vengono ridimensionati verso l'alto e verso il basso fino alla stessa capacità del writer. Non si imposta l'intervallo di capacità separatamente per gli standby.

Nodi totali	Router	Partizioni	Capacità minima predefinita () ACUs	Intervallo di capacità massima (ACUs)
4	2	2	16	16—400
5	2	3	20	401—500
6	2	4	24	501—600
7	3	4	28	601—700
8	3	5	32	701—800
9	3	6	36	801—900
10	4	6	40	901—1000
11	4	7	44	1001—1100
12	4	8	48	1101—1200
13	5	8	52	1201—1300
14	5	9	56	1301—1400
15	5	10	60	1401—1500
16	6	10	64	1501—1600

Nodi totali	Router	Partizioni	Capacità minima predefinita () ACUs	Intervallo di capacità massima (ACUs)
17	6	11	68	1601—1700
18	6	12	72	1701—1800
19	7	12	76	1801—1900
20	7	13	80	1901—2000
21	7	14	84	2001—2100
22	8	14	88	2101—2020
23	8	15	92	2201—2300
24	8	16	96	2301—6144

La configurazione dinamica basata sulla capacità massima per il gruppo di shard DB è disponibile solo durante la creazione. Il numero di router e shard rimane lo stesso quando viene modificata la capacità massima. Per ulteriori informazioni, consulta Modifica della capacità di un gruppo di shard DB.

È possibile utilizzare SQL i comandi per aggiungere shard e router a un gruppo di shard DB. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Dividere uno shard in un gruppo di shard DB
- Aggiungere un router a un gruppo di shard DB

> **Note**
>
> Non è possibile eliminare shard o router.

Creazione del cluster DB

Utilizzare le seguenti procedure per creare un cluster Aurora PostgreSQL DB che utilizza Aurora PostgreSQL Limitless Database.

È possibile utilizzare AWS Management Console o il AWS CLI per creare il cluster DB che utilizza Aurora PostgreSQL Limitless Database. Si creano il cluster DB primario e il gruppo di shard DB.

Console

Quando si utilizza il AWS Management Console per creare il cluster DB primario, anche il gruppo di shard DB viene creato con la stessa procedura.

Per creare il cluster DB utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Crea database.

   Viene visualizzata la pagina Crea database.
3. Per Tipo di motore scegli Aurora (compatibile con PostgreSQL).
4. Per Versione, scegli una delle seguenti opzioni:

   - Aurora PostgreSQL con database Limitless (compatibile con PostgreSQL 16.4)
   - Aurora PostgreSQL con database Limitless (compatibile con PostgreSQL 16.6)
5. Per Aurora PostgreSQL Limitless Database:

a. Immettere un identificatore di gruppo di shard DB.

> **Important**
>
> Dopo aver creato il gruppo di shard DB, non è possibile modificare l'identificatore del cluster DB o l'identificatore del gruppo di shard DB.

b. Per l'intervallo di capacità del gruppo di shard DB:

   i. Immettere la capacità minima (ACUs). Utilizzate un valore di almeno 16 ACUs.

      Per un ambiente di sviluppo, il valore predefinito è 16 ACUs. Per un ambiente di produzione, il valore predefinito è 24 ACUs.

   ii. Immettere la capacità massima (ACUs). Utilizzate un valore di almeno 16 ACUs o al massimo 6144 ACUs.

      Per un ambiente di sviluppo, il valore predefinito è 64 ACUs. Per un ambiente di produzione, il valore predefinito è 384 ACUs.

   Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati.

c. Per la distribuzione di gruppi di shard DB, scegli se creare standby per il gruppo di shard DB:

- Nessuna ridondanza di elaborazione: crea un gruppo di shard DB senza standby per ogni shard. Si tratta del valore di default.
- Ridondanza di calcolo con una singola destinazione di failover: crea un gruppo di shard DB con uno standby di elaborazione in una zona di disponibilità (AZ) diversa.
- Ridondanza di calcolo con due destinazioni di failover: crea un gruppo di shard DB con due standby di elaborazione in due diversi. AZs

> **Note**
>
> Se si imposta la ridondanza di elaborazione su un valore diverso da zero, il numero totale di shard di istanze DB verrà raddoppiato o triplicato. Queste istanze DB aggiuntive sono istanze di calcolo in standby, scalate verso l'alto e verso il basso fino alla stessa capacità dell'istanza writer. L'intervallo di capacità non viene impostato separatamente per gli standby. Pertanto, l'utilizzo e la fattura dell'ACU raddoppiano e triplicano di conseguenza. Per conoscere l'esatto utilizzo dell'ACU derivante dalla ridondanza di calcolo, consulta la metrica in. `DBShardGroupComputeRedundancyCapacity` DBShardMetriche di gruppo

d. Scegli se rendere il gruppo di shard DB accessibile al pubblico.

> **Note**
>
> Non è possibile modificare questa impostazione dopo aver creato il gruppo di shard DB.

6. Per la connettività:

a. (Facoltativo) Seleziona Connetti a una risorsa di EC2 calcolo, quindi scegli un' EC2 istanza esistente o creane una nuova.

> **Note**
>
> Se ti connetti a un' EC2 istanza, non puoi rendere il gruppo di shard DB accessibile pubblicamente.

b. Per Tipo di rete, scegli una delle due modalità IPv4o Dual-stack.

c. Scegli il cloud privato virtuale (VPC) e il gruppo di sottoreti DB o utilizza le impostazioni predefinite.

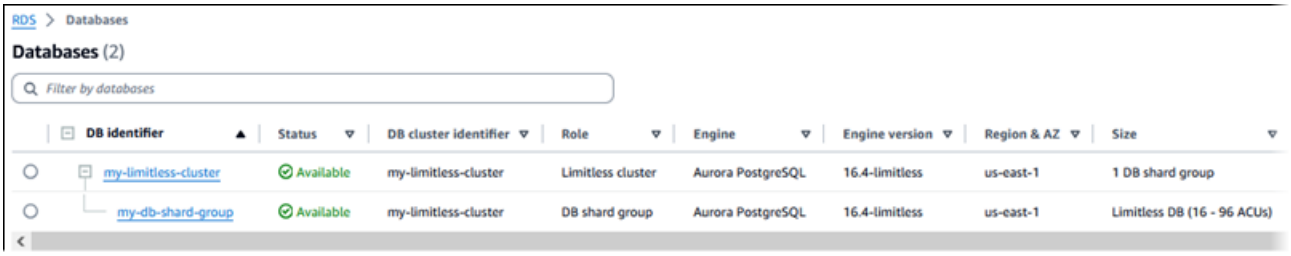
> **Note**
>
> Se create il cluster Limitless Database DB nella regione Stati Uniti orientali (Virginia settentrionale), non includete `us-east-1e` la zona di disponibilità (AZ) nel gruppo di sottoreti DB. A causa delle limitazioni delle risorse, Aurora Serverless v2 e quindi Limitless Database, non è supportato nella AZ. `us-east-1e`

d. Scegli il gruppo di sicurezza VPC (firewall) o utilizza l'impostazione predefinita.

7. Per l'autenticazione del database, scegli Autenticazione tramite password o Autenticazione del database tramite password e IAM.
8. Per il monitoraggio, assicurati che le caselle di controllo Attiva Performance Insights e Abilita monitoraggio avanzato siano selezionate.

   Per Performance Insights, scegli un periodo di conservazione di almeno 1 mese.
9. Espandi l'ultima configurazione aggiuntiva nella pagina.
10. Per le esportazioni di log, assicurati che la casella di controllo del registro PostgreSQL sia selezionata.
11. Specificate altre impostazioni, se necessario. Per ulteriori informazioni, consulta Impostazioni per cluster di database Aurora.
12. Scegliere Crea database.

Dopo aver creato il cluster DB primario e il gruppo di shard DB, vengono visualizzati nella pagina Database.


RDS > Databases
Databases (2)
Filter by databases
DB identifier
Status
DB cluster identifier
Role
Engine
Engine version
Region & AZ
Size
my-limitless-cluster
Available
my-limitless-cluster
Limitless cluster
Aurora PostgreSQL
16.4-limitless
us-east-1
1 DB shard group
my-db-shard-group
Available
my-limitless-cluster
DB shard group
Aurora PostgreSQL
16.4-limitless
us-east-1
Limitless DB (16 - 96 ACUs)

CLI

Quando si utilizza AWS CLI per creare un cluster DB che utilizza Aurora PostgreSQL Limitless Database, si eseguono le seguenti attività:

1. Creare il cluster DB primario.
2. Crea il gruppo di shard DB.

Creare il cluster DB primario

Per creare il cluster DB sono necessari i seguenti parametri:

- `--db-cluster-identifier`: nome del cluster di database.
- `--engine`— Il cluster DB deve utilizzare il motore `aurora-postgresql` DB.
- `--engine-version`— Il cluster DB deve utilizzare una delle versioni del motore DB:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`— Il cluster DB deve utilizzare la configurazione di archiviazione del cluster `aurora-iopt1` DB.
- `--cluster-scalability-type`— specifica la modalità di scalabilità del cluster Aurora DB. Se impostato su`limitless`, il cluster funziona come un database Aurora PostgreSQL Limitless. Se impostato su `standard` (impostazione predefinita), il cluster utilizza la normale creazione di istanze DB.

  > **Note**
  >
  > Non è possibile modificare questa impostazione dopo aver creato il cluster DB.

- `--master-username`— Il nome dell'utente principale per il cluster DB.
- `--master-user-password`— La password per l'utente principale.
- `--enable-performance-insights`— È necessario abilitare Performance Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione di Performance Insights deve essere di almeno 31 giorni.
- `--monitoring-interval`— L'intervallo, in secondi, tra i punti in cui vengono raccolte le metriche di Enhanced Monitoring per il cluster DB. Questo valore non può essere. `0`

- `--monitoring-role-arn`— Amazon Resource Name (ARN) per il ruolo IAM che consente a RDS di inviare metriche di Enhanced Monitoring ad Amazon Logs. CloudWatch
- `--enable-cloudwatch-logs-exports`— È necessario esportare i log in Logs. `postgresql` CloudWatch

I parametri seguenti sono facoltativi:

- `--db-subnet-group-name`— Il gruppo di sottoreti DB da associare al cluster DB. Ciò determina anche il VPC associato al cluster DB.

  > **Note**
  >
  > Se create il vostro cluster Limitless Database DB nella regione Stati Uniti orientali (Virginia settentrionale), non includete `us-east-1e` la zona di disponibilità (AZ) nel sottogruppo di database. A causa delle limitazioni delle risorse, Aurora Serverless v2 e quindi Limitless Database, non è supportato nella AZ. `us-east-1e`

- `--vpc-security-group-ids`— Un elenco di gruppi di sicurezza VPC da associare al cluster DB.
- `--performance-insights-kms-key-id`— L' AWS KMS key identificatore per la crittografia dei dati di Performance Insights. Se non specifichi una chiave KMS, Account AWS viene utilizzata la chiave predefinita per te.
- `--region`— Il Regione AWS luogo in cui si crea il cluster DB. Deve essere uno che supporti Aurora PostgreSQL Limitless Database.

Per utilizzare il VPC e il gruppo di sicurezza VPC predefiniti, ometti le opzioni and. `--db-subnet-group-name` `--vpc-security-group-ids`

Per creare il cluster DB primario

- ```
  aws rds create-db-cluster \
      --db-cluster-identifier my-limitless-cluster \
      --engine aurora-postgresql \
      --engine-version 16.6-limitless \
      --storage-type aurora-iopt1 \
      --cluster-scalability-type limitless \
      --master-username myuser \
  ```
  ```

```
    --master-user-password mypassword \
    --db-subnet-group-name mysubnetgroup \
    --vpc-security-group-ids sg-c7e5b0d2 \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
    --enable-cloudwatch-logs-exports postgresql
```

Per ulteriori informazioni, consulta create-db-cluster.

Crea il gruppo di shard DB

Successivamente crei il gruppo di shard DB nel cluster DB che hai appena creato. I parametri seguenti sono obbligatori:

- `--db-shard-group-identifier`— Il nome del gruppo di shard DB.

  L'identificatore del gruppo di shard DB presenta i seguenti vincoli:

  - Deve essere unico nel luogo in Regione AWS cui viene Account AWS creato.
  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  -

    > **Important**
    >
    > Dopo aver creato il gruppo di shard DB, non è possibile modificare l'identificatore del cluster DB o l'identificatore del gruppo di shard DB.

- `--db-cluster-identifier`— Il nome del cluster DB in cui state creando il gruppo di shard DB.
- `--max-acu`— La capacità massima del gruppo di shard DB. Deve essere compresa tra 16 ACUs e 6144. Per limiti di capacità superiori a ACUs 6144, contattare. AWS

  Il numero iniziale di router e shard è determinato dalla capacità massima impostata al momento della creazione del gruppo di shard DB. Maggiore è la capacità massima, maggiore è il numero di router e shard creati nel gruppo di shard DB. Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati.

I parametri seguenti sono facoltativi:

- `--compute-redundancy`— Se creare standby per il gruppo di shard DB. Questo parametro può avere i seguenti valori:
  - `0`— Crea un gruppo di shard DB senza standby per ogni shard. Si tratta del valore di default.
  - `1`— Crea un gruppo di shard DB con uno standby di calcolo in una zona di disponibilità (AZ) diversa.
  - `2`— Crea un gruppo di shard DB con due standby di calcolo in due diversi. AZs

> **Note**
>
> Se si imposta la ridondanza di calcolo su un valore diverso da zero, il numero totale di shard verrà raddoppiato o triplicato. Ciò comporterà costi aggiuntivi.
> I nodi in standby di elaborazione vengono ridimensionati verso l'alto e verso il basso fino alla stessa capacità del writer. Non si imposta l'intervallo di capacità separatamente per gli standby.

- `--min-acu`— La capacità minima del gruppo di shard DB. Deve essere almeno 16 ACUs, che è il valore predefinito.
- `--publicly-accessible|--no-publicly-accessible`— Se assegnare indirizzi IP accessibili pubblicamente al gruppo di shard DB. L'accesso al gruppo di shard DB è controllato dai gruppi di sicurezza utilizzati dal cluster.

  Il valore predefinito è `--no-publicly-accessible`.

> **Note**
>
> Non è possibile modificare questa impostazione dopo aver creato il gruppo di shard DB.

Per creare il gruppo di shard DB

- ```
  aws rds create-db-shard-group \
      --db-shard-group-identifier my-db-shard-group \
      --db-cluster-identifier my-limitless-cluster \
      --max-acu 1000
  ```
  ```

Lavorare con gruppi di shard DB

È possibile eseguire le seguenti attività per aggiungere e gestire un gruppo di shard DB per Aurora PostgreSQL Limitless Database.

Argomenti

- Connessione al cluster Aurora PostgreSQL Limitless Database DB
- Individuazione del numero di router e shard in un gruppo di shard DB
- Descrizione dei gruppi di shard DB
- Riavvio di un gruppo di shard DB
- Modifica della capacità di un gruppo di shard DB
- Dividere uno shard in un gruppo di shard DB
- Aggiungere un router a un gruppo di shard DB
- Eliminazione di un gruppo di shard DB
- Aggiungere un gruppo di shard DB a un cluster Aurora SQL Postgre Limitless Database DB esistente

Connessione al cluster Aurora PostgreSQL Limitless Database DB

Per lavorare con Aurora PostgreSQL Limitless Database, ci si connette all'endpoint cluster writer o reader. Puoi usare `psql` o qualsiasi altra utilità di connessione che funzioni con PostgreSQL:

```
$ psql -h DB_cluster_endpoint -p port_number -U database_username -d postgres_limitless
```

L'esempio seguente utilizza l'endpoint per il cluster DB in cui è stato creato. CLI

```
$ psql -h my-limitless-cluster.cluster-ckifpdyyyxxx.us-east-1.rds.amazonaws.com -p 5432
 -U postgres -d postgres_limitless
```

> **Note**
>
> Il database predefinito per il gruppo di shard DB in Aurora PostgreSQL Limitless Database è. `postgres_limitless`

Utilizzo del plug-in Limitless Connection

Quando si connettono a Aurora PostgreSQL Limitless Database, i client si connettono utilizzando l'endpoint del cluster e vengono instradati verso un router di transazione da Amazon Route 53. Tuttavia, Route 53 ha una capacità limitata di bilanciamento del carico e può consentire carichi di lavoro irregolari sui router di transazione. Il Limitless Connection Plugin per il driver AWS JDBC risolve questo problema eseguendo il bilanciamento del carico sul lato client con riconoscimento del carico. Per ulteriori informazioni sul driver JDBC, consulta.AWS Connessione ad Aurora PostgreSQL con il driver JDBC Amazon Web Services ()AWS

Individuazione del numero di router e shard in un gruppo di shard DB

È possibile utilizzare la seguente query per trovare il numero di router e shard:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Descrizione dei gruppi di shard DB

Usa il `describe-db-shard-groups` AWS CLI comando per descrivere i tuoi gruppi di shard DB. Il parametro seguente è facoltativo:

- `--db-shard-group-identifier`— Il nome di un gruppo di shard DB.

L'esempio seguente descrive uno specifico gruppo di shard DB.

```
aws rds describe-db-shard-groups --db-shard-group-identifier my-db-shard-group
```

L'output è simile al seguente esempio.

```
{
    "DBShardGroups": [
        {
            "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
            "DBShardGroupIdentifier": "my-shard-group",
            "DBClusterIdentifier": "my-limitless-cluster",
            "MaxACU": 1000.0,
            "ComputeRedundancy": 0,
            "Status": "available",
            "PubliclyAccessible": false,
            "Endpoint": "my-limitless-cluster.limitless-ccetp2abcdef.us-
east-1.rds.amazonaws.com"
        }
```

```
    ]
}
```

Riavvio di un gruppo di shard DB

A volte è necessario riavviare il gruppo di shard DB, ad esempio quando il `max_connections` parametro cambia a causa di una modifica della capacità massima.

È possibile utilizzare AWS Management Console o AWS CLI per modificare la capacità di un gruppo di shard DB.

Console

Attenersi alla seguente procedura:

Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

1. Passare alla pagina Databases (Database).
2. Seleziona il gruppo di shard DB che desideri riavviare.
3. In Actions (Operazioni), scegliere Reboot (Riavvia).
4. Scegli Conferma.

CLI

Per riavviare un gruppo di shard DB, utilizzate il `reboot-db-shard-group` AWS CLI comando con il seguente parametro:

- `--db-shard-group-identifier`— Il nome di un gruppo di shard DB.

L'esempio seguente riavvia un gruppo di shard DB.

```
aws rds reboot-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Modifica della capacità di un gruppo di shard DB

È possibile utilizzare AWS Management Console o AWS CLI per modificare la capacità di un gruppo di shard DB.

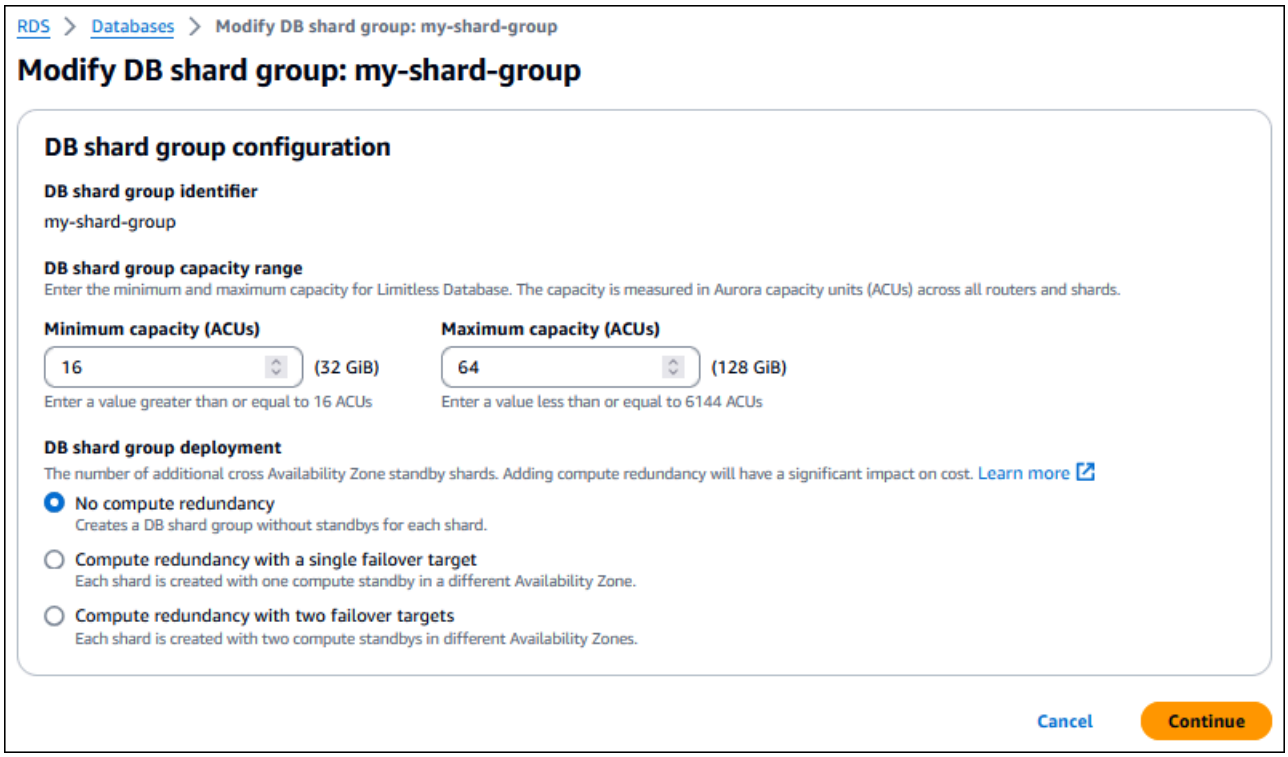
Console

Attenersi alla seguente procedura:

Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

1. Passare alla pagina Databases (Database).
2. Seleziona il gruppo di shard DB che desideri modificare.
3. Per Operazioni, scegli Modifica.

   Viene visualizzata la pagina Modifica gruppo di shard DB.



4. Ad esempio**100**, immettete un nuovo valore di capacità minima (ACUs).
5. Ad esempio, immettete un nuovo valore di capacità massima (ACUs)**1000**.
6. Scegli Continua.

   Viene visualizzata la pagina di conferma con un riepilogo delle modifiche.

7. Controlla le modifiche, quindi scegli Modifica gruppo di shard DB.

CLI

Usa il comando `modify-db-shard-group` AWS CLI con i parametri seguenti.

- `--db-shard-group-identifier`— Il nome del gruppo di shard DB.
- `--max-acu`— La nuova capacità massima del gruppo di shard DB. È possibile impostare la capacità massima del gruppo di shard DB su 16—6144. ACUs Per limiti di capacità superiori a 6144, contattare. ACUs AWS

  Il numero di router e shard non cambia.
- `--min-acu`— La nuova capacità minima del gruppo di shard DB. Deve essere almeno 16ACUs, che è il valore predefinito.

L'CLIesempio seguente modifica l'intervallo di capacità di un gruppo di shard DB portandolo a ACUs 100-1000.

```
aws rds modify-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --min-acu 100 \
    --max-acu 1000
```

Dividere uno shard in un gruppo di shard DB

È possibile dividere manualmente uno shard in un gruppo di shard DB in due shard più piccoli. Questa operazione è denominata divisione dello shard avviata dall'utente.

Aurora PostgreSQL Limitless Database può anche dividere gli shard quando hanno grandi quantità di dati o un utilizzo molto elevato. Questa operazione è denominata divisione dello shard avviata dal sistema.

Argomenti

- Prerequisiti
- Dividere uno shard
- Monitoraggio delle divisioni dei frammenti
- Finalizzazione delle suddivisioni degli shard
- Annullamento di uno shard split

Prerequisiti

Gli shard split avviati dall'utente hanno i seguenti prerequisiti:

- È necessario disporre di un gruppo di shard DB.
- Il gruppo di shard DB non può essere vuoto: deve contenere almeno una tabella condivisa.
- Un utente deve avere il privilegio. `rds_aurora_limitless_cluster_admin` `rds_superuser`Ha questo privilegio, quindi lo possiede anche l'utente principale. `rds_superuser`Possono concedere il privilegio ad altri utenti:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

- È necessario conoscere l'ID del sottocluster (nodo) dello shard che si desidera dividere. È possibile ottenere l'ID utilizzando la seguente query:

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
```

```
2               | router
3               | shard
4               | shard
5               | shard
6               | shard
```

Per abilitare le suddivisioni degli shard avviate dal sistema, impostate i seguenti parametri del cluster DB in un gruppo di parametri del cluster DB personalizzato associato al cluster DB:

Parametro	Valore
`rds_aurora.limitless_enable_auto_scale`	on
`rds_aurora.limitless_auto_scale_options`	O `split_shard add_router,split_shard`
`rds_aurora.limitless_finalize_split_shard_mode`	Questo parametro determina come vengono finalizzate le suddivisioni degli shard avviate dal sistema. Il valore può essere uno dei seguenti: • `user_initiated` — Siete voi a decidere quando finalizzare lo shard split. Si tratta del valore di default. • `immediate` — Le divisioni degli shard vengono finalizzate immediatamente. Per ulteriori informazioni, consulta Finalizzazione delle suddivisioni degli shard. **Note** Questo parametro si applica solo alle suddivisioni di shard avviate dal sistema.

Per ulteriori informazioni, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

Dividere uno shard

Per dividere uno shard in un gruppo di shard DB, usa la funzione. `rds_aurora.limitless_split_shard` Questa funzione avvia un processo shard-split che viene eseguito in modo asincrono.

```
SELECT rds_aurora.limitless_split_shard('subcluster_id');
```

Attendi la restituzione di un ID di lavoro dopo l'invio corretto del lavoro, ad esempio:

```
SELECT rds_aurora.limitless_split_shard('3');

    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Le operazioni simultanee di divisione degli shard non sono supportate. Esegui ogni operazione in sequenza e completa ogni operazione prima di iniziare un'altra operazione di aggiunta.

Monitoraggio delle divisioni dei frammenti

È possibile utilizzare l'ID del lavoro per tenere traccia di un lavoro suddiviso in frammenti. Per descrivere un particolare lavoro e ottenere maggiori dettagli su di esso, esegui la seguente query:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(job_id);
```

Per esempio:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(1691300000000);

    job_id     |    action    |      job_details      | status  |    submission_time
 |                 message
```

```
--------------+------------+----------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | SPLIT_SHARD | Split Shard 3 by User | SUCCESS | 2023-08-06 05:33:20+00
 | Scaling job succeeded.                +
               |            |                       |         |
 | New shard instance with ID 7 was created.
(1 row)
```

La query restituisce un errore quando si passa un lavoro inesistente come input.

```
SELECT * from rds_aurora.limitless_list_shard_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

È possibile tenere traccia dello stato di tutti i job suddivisi in shard-split utilizzando la stessa query senza un ID di lavoro, ad esempio:

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs();

    job_id     |   action    |  job_details          |   status    |    submission_time
     |                 message
--------------+------------+----------------------+-------------
+------------------------
+-------------------------------------------------------------
 1691200000000 | SPLIT_SHARD | Split Shard 3 by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | SPLIT_SHARD | Split Shard 4 by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded. +
               |            |                       |             |
     | New shard instance with ID 7 was created.
 1691400000000 | SPLIT_SHARD | Split Shard 5 by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add shard job 1691400000000.
               |            |                       |             |
     | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | SPLIT_SHARD | Split Shard 5 by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

Lo stato del lavoro può essere uno dei seguenti:

- `IN_PROGRESS`— Il lavoro frazionato è stato inviato ed è in corso. È possibile avere un solo lavoro in corso alla volta.

- `PENDING`— Il lavoro shard-split è in attesa che tu lo finalizzi. Per ulteriori informazioni, consulta Finalizzazione delle suddivisioni degli shard.
- `CANCELLATION_IN_PROGRESS`— Il lavoro shard-Split viene annullato dall'utente.
- `CANCELED`— Il lavoro shard-Split è stato annullato con successo dall'utente o dal sistema.
- `SUCCESS`— Il lavoro shard-Split è stato completato correttamente. Il `message` campo contiene l'ID dell'istanza del nuovo shard.
- `FAILED`— Il processo shard-Split non è riuscito. Il `message` campo contiene i dettagli dell'errore e tutte le azioni che possono essere intraprese in seguito al processo fallito.

Finalizzazione delle suddivisioni degli shard

La finalizzazione è l'ultima fase del processo di suddivisione degli shard-split. Ciò causa alcuni tempi di inattività. Se si avvia un lavoro suddiviso in frammenti, la finalizzazione avviene immediatamente dopo il completamento corretto del lavoro.

A volte il sistema divide gli shard in base al carico di lavoro, se avete abilitato le suddivisioni di shard avviate dal sistema utilizzando il parametro. `rds_aurora.limitless_enable_auto_scale`

In questo caso, puoi scegliere se la finalizzazione avvenga immediatamente o in un momento a tua scelta. Utilizzate il parametro del cluster `rds_aurora.limitless_finalize_split_shard_mode` DB per scegliere quando ciò avviene:

- Se si imposta il valore su`immediate`, ciò avviene immediatamente.
- Se si imposta il valore su`user_initiated`, è necessario finalizzare manualmente il lavoro shard-split.

  Ti viene inviato un evento RDS e lo stato del processo shard-Split è impostato su. `PENDING`

Quando è impostato su`user_initiated`, si utilizza la `rds_aurora.limitless_finalize_split_shard` funzione per finalizzare il lavoro shard-Split:

```
SELECT * FROM rds_aurora.limitless_finalize_split_shard(job_id);
```

> **Note**
>
> Questa funzione si applica solo alle suddivisioni di shard avviate dal sistema, non dall'utente.

Annullamento di uno shard split

È possibile annullare uno shard split avviato dall'utente o dal sistema che è o. `IN_PROGRESS` `PENDING` È necessario l'ID del lavoro per annullarlo.

```
SELECT * from rds_aurora.limitless_cancel_shard_scale_jobs(job_id);
```

Non viene restituito alcun output a meno che non si verifichi un errore. Puoi tenere traccia dell'annullamento utilizzando una query di tracciamento del lavoro.

Aggiungere un router a un gruppo di shard DB

È possibile aggiungere un router a un gruppo di shard DB.

Argomenti

- Prerequisiti
- Aggiungere un router
- Monitoraggio delle aggiunte al router
- Annullamento dell'aggiunta di un router

Prerequisiti

L'aggiunta di un router presenta i seguenti prerequisiti:

- È necessario disporre di un gruppo di shard DB.
- Un utente deve avere il `rds_aurora_limitless_cluster_admin` privilegio. `rds_superuser`Ha questo privilegio, quindi lo possiede anche l'utente principale. `rds_superuser`Possono concedere il privilegio ad altri utenti:

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

> **Note**
>
> Se modifichi il certificato CA predefinito dopo la creazione Account AWS del gruppo di shard DB, il nuovo router utilizzerà il nuovo certificato CA, che è diverso dal certificato CA del router esistente. A seconda del trust store, alcune connessioni potrebbero fallire.

- Per abilitare l'aggiunta di router avviata dal sistema, imposta i seguenti parametri del cluster DB in un gruppo di parametri del cluster DB personalizzato associato al cluster DB:

Parametro	Valore
`rds_aurora.limitless_enable_auto_scale`	on

Parametro	Valore
`rds_aurora.limitless_auto_scale_options`	O `add_router` `add_router,split_shard`

Per ulteriori informazioni, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

Aggiungere un router

Per aggiungere un router, usa la `rds_aurora.limitless_add_router` funzione. Questa funzione avvia un processo di aggiunta del router che viene eseguito in modo asincrono.

```
SELECT rds_aurora.limitless_add_router();
```

Attendi la restituzione di un ID di lavoro dopo l'invio corretto del lavoro, ad esempio:

```
    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Le operazioni di aggiunta simultanea del router non sono supportate. Esegui le operazioni in sequenza e completa ogni operazione prima di iniziare un'altra operazione di aggiunta.

Monitoraggio delle aggiunte al router

È possibile utilizzare l'ID del lavoro per tenere traccia di un processo di aggiunta al router. Per descrivere un particolare lavoro e ottenere maggiori dettagli al riguardo, esegui la seguente query:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(job_id);
```

Per esempio:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(1691300000000);

    job_id     |   action   |        job_details       | status  |    submission_time
   |                   message
---------------+------------+--------------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                  +
               |            |                          |         |
   | New router instance with ID 7 was created.
(1 row)
```

La query restituisce un errore quando si passa un lavoro inesistente come input.

```
SELECT * from rds_aurora.limitless_list_router_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

È possibile tenere traccia dello stato di tutti i processi di aggiunta del router utilizzando la stessa query senza un ID di lavoro, ad esempio:

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs();

    job_id     |   action   |        job_details       |   status    |
 submission_time     |                  message
---------------+------------+--------------------------+-------------
+------------------------+-------------------------------------------
 1691200000000 | ADD_ROUTER | Add 1 new Router by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                 +
               |            |                          |             |
       | New router instance with ID 7 was created.
 1691400000000 | ADD_ROUTER | Add 1 new Router by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add router job 1691400000000.
               |            |                          |             |
       | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | ADD_ROUTER | Add 1 new Router by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

Lo stato del lavoro può essere uno dei seguenti:

- `IN_PROGRESS`— Il processo di aggiunta del router è stato inviato ed è in corso. È possibile avere un solo lavoro in corso alla volta.
- `CANCELLATION_IN_PROGRESS`— Il processo di aggiunta del router viene annullato dall'utente.
- `CANCELED`— Il processo di aggiunta del router è stato annullato con successo dall'utente o dal sistema.
- `SUCCESS`— Il processo di aggiunta del router è stato completato correttamente. Il `message` campo contiene l'ID di istanza del nuovo router.
- `FAILED`— Il processo di aggiunta del router non è riuscito. Il `message` campo contiene i dettagli dell'errore e tutte le azioni che possono essere intraprese in seguito al processo fallito.

> **Note**
>
> Non c'è nessuno `PENDING` status perché non è necessario finalizzare le aggiunte al router. Non comportano tempi di inattività.

Annullamento dell'aggiunta di un router

Puoi annullare l'aggiunta di un router. `IN_PROGRESS` È necessario l'ID del lavoro per annullarlo.

```
SELECT * from rds_aurora.limitless_cancel_router_scale_jobs(job_id);
```

Non viene restituito alcun output a meno che non si verifichi un errore. Puoi tenere traccia dell'annullamento utilizzando una query di tracciamento del lavoro.

Eliminazione di un gruppo di shard DB

È possibile eliminare un gruppo di shard DB se necessario. L'eliminazione del gruppo di shard DB elimina i nodi di calcolo (shard e router), ma non lo storage.

Console

Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

1. Passare alla pagina Databases (Database).
2. Seleziona il gruppo di shard DB che desideri eliminare.
3. In Actions (Azioni), scegliere Delete (Elimina).
4. Entra **`delete me`** nella casella, quindi scegli Elimina.

Il gruppo di shard DB viene eliminato.

AWS CLI

Per eliminare il gruppo di shard DB, utilizzate il `delete-db-shard-group` AWS CLI comando con il seguente parametro:

- `--db-shard-group-identifier`— Il nome del gruppo di shard DB.

L'esempio seguente elimina un gruppo di shard DB nel cluster SQL DB Aurora Postgre creato in precedenza.

```
aws rds delete-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Aggiungere un gruppo di shard DB a un cluster Aurora SQL Postgre Limitless Database DB esistente

Puoi creare un gruppo di shard DB in un cluster DB esistente, ad esempio se stai ripristinando un cluster DB o hai eliminato il gruppo di shard DB.

Per ulteriori informazioni sui requisiti del cluster DB primario e del gruppo di shard DB, consulta. Requisiti e considerazioni del database Aurora PostgreSQL Limitless

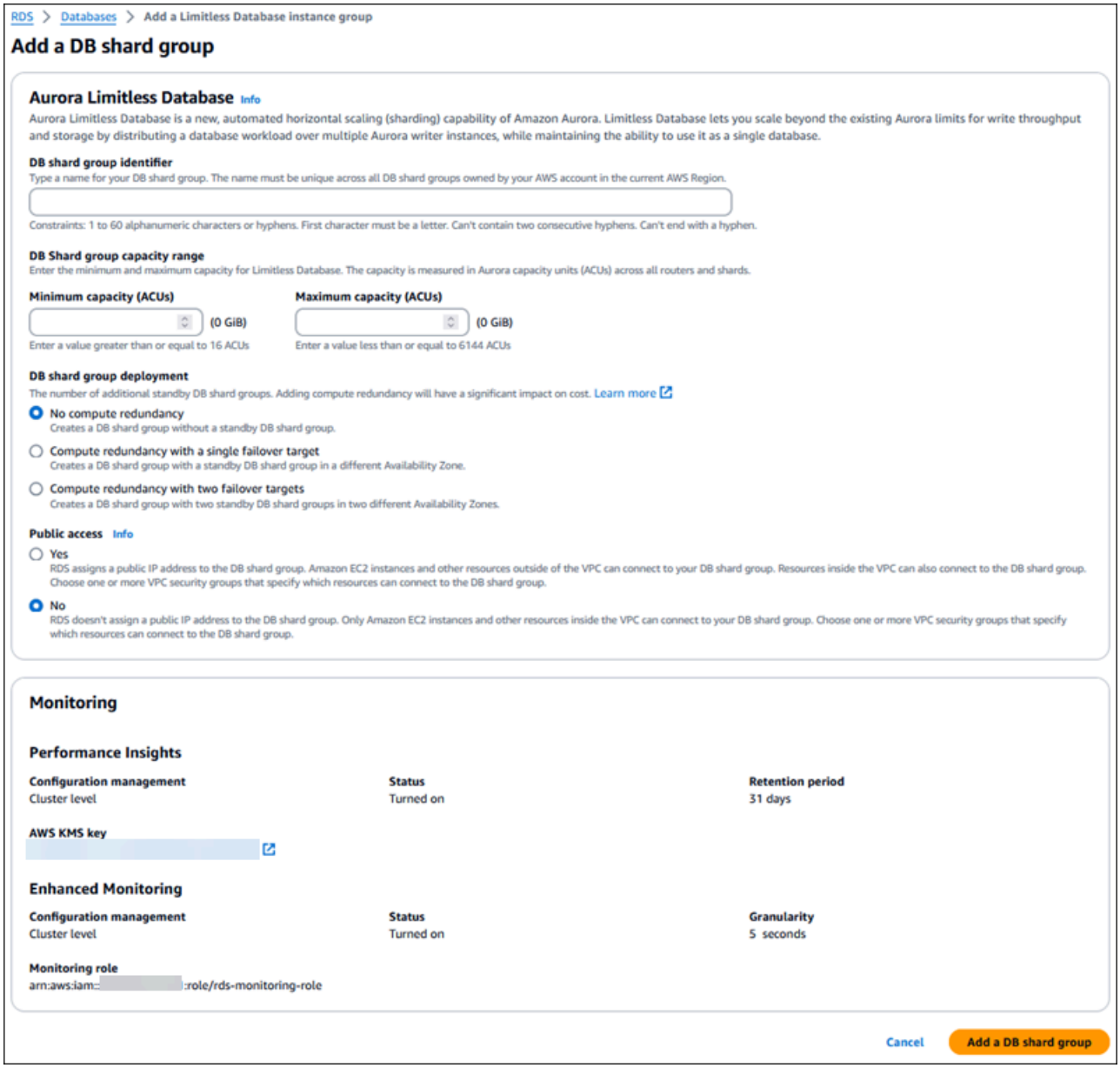
> **Note**
>
> È possibile avere un solo gruppo di shard DB per cluster.
> Il cluster Limitless Database DB deve trovarsi `available` nello stato in cui si trova prima di poter creare un gruppo di shard DB.

Console

È possibile utilizzare il AWS Management Console per aggiungere un gruppo di shard DB a un cluster DB esistente.

Per aggiungere un gruppo di shard DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Passare alla pagina Databases (Database).
3. Seleziona il cluster Limitless Database DB a cui desideri aggiungere un gruppo di shard DB.
4. Per Azioni, scegli Aggiungi un gruppo di shard DB.



5. Immettere un identificatore di gruppo di shard DB.

> **Important**
>
> Dopo aver creato il gruppo di shard DB, non è possibile modificare l'identificatore del cluster DB o l'identificatore del gruppo di shard DB.

6. Immettete la capacità minima (). ACUs Utilizzate un valore di almeno 16ACUs.
7. Immettere la capacità massima (ACUs). Utilizzate un valore compreso tra 16 ACUs e 6144.

Per ulteriori informazioni, consulta Correlazione della capacità massima del gruppo di shard DB con il numero di router e shard creati.

8. Per la distribuzione di gruppi di shard DB, scegli se creare standby per il gruppo di shard DB:
   - Nessuna ridondanza di elaborazione: crea un gruppo di shard DB senza standby per ogni shard. Si tratta del valore di default.
   - Ridondanza di calcolo con una singola destinazione di failover: crea un gruppo di shard DB con uno standby di elaborazione in una zona di disponibilità (AZ) diversa.
   - Ridondanza di calcolo con due destinazioni di failover: crea un gruppo di shard DB con due standby di elaborazione in due diversi. AZs
9. Scegli se rendere il gruppo di shard DB accessibile al pubblico.

   > **Note**
   >
   > Non è possibile modificare questa impostazione dopo aver creato il gruppo di shard DB.

10. Scegli Aggiungi un gruppo di shard DB.

AWS CLI

Usa il `create-db-shard-group` AWS CLI comando per creare un gruppo di shard DB.

I parametri seguenti sono obbligatori:

- `--db-cluster-identifier`— Il cluster DB a cui appartiene il gruppo di shard DB.
- `--db-shard-group-identifier`— Il nome del gruppo di shard DB.

  L'identificatore del gruppo di shard DB presenta i seguenti vincoli:

  - Deve essere unico nel luogo in Regione AWS cui viene Account AWS creato.
  - Deve contenere da 1 a 63 lettere, numeri o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.

  > **Important**
  >
  > Dopo aver creato il gruppo di shard DB, non è possibile modificare l'identificatore del cluster DB o l'identificatore del gruppo di shard DB.

- `--max-acu`— La capacità massima del gruppo di shard DB. Utilizzate un valore compreso tra 16 ACUs e 6144.

I parametri seguenti sono facoltativi:

- `--compute-redundancy`— Se creare standby per il gruppo di condivisione DB. Questo parametro può avere i seguenti valori:
  - `0`— Crea un gruppo di shard DB senza standby per ogni shard. Si tratta del valore di default.
  - `1`— Crea un gruppo di shard DB con uno standby di calcolo in una zona di disponibilità (AZ) diversa.
  - `2`— Crea un gruppo di shard DB con due standby di calcolo in due diversi. AZs

  > **Note**
  >
  > Se si imposta la ridondanza di calcolo su un valore diverso da zero, il numero totale di nodi verrà raddoppiato o triplicato. Ciò comporterà costi aggiuntivi.

- `--min-acu`— La capacità minima del gruppo di shard DB. Deve essere almeno 16ACUs, che è il valore predefinito.
- `--publicly-accessible|--no-publicly-accessible`— Se assegnare indirizzi IP accessibili pubblicamente al gruppo di shard DB. L'accesso al gruppo di shard DB è controllato dai gruppi di sicurezza utilizzati dal cluster.

  Il valore predefinito è `--no-publicly-accessible`.

  > **Note**
  >
  > Non è possibile modificare questa impostazione dopo aver creato il gruppo di shard DB.

L'esempio seguente crea un gruppo di shard DB in un cluster DB Aurora SQL Postgree.

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-db-cluster \
    --db-shard-group-identifier my-new-shard-group \
    --max-acu 1000
```

L'output è simile all'esempio seguente.

```
{
    "Status": "CREATING",
    "Endpoint": "my-db-cluster.limitless-ckifpdyyyxxx.us-east-1.rds.amazonaws.com",
    "PubliclyAccessible": false,
    "DBClusterIdentifier": "my-db-cluster",
    "MaxACU": 1000.0,
    "DBShardGroupIdentifier": "my-new-shard-group",
    "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
    "ComputeRedundancy": 0
}
```

Creazione di tabelle del database Aurora SQL Postgre Limitless

Esistono tre tipi di tabelle che contengono i dati in Aurora SQL Postgre Limitless Database:

- Standard: questo è il tipo di tabella predefinito in Aurora SQL Postgre Limitless Database. È possibile creare tabelle standard utilizzando il CREATETABLEcomando e su di esse eseguire operazioni Data Description Language (DDL) e Data Manipulation Language (DML).

  Le tabelle standard non sono tabelle distribuite. Sono archiviate su uno degli shard scelti internamente dal sistema.
- Sharded: queste tabelle sono distribuite su più frammenti. I dati vengono suddivisi tra i frammenti in base ai valori delle colonne designate nella tabella. Questo set di colonne è chiamato shard key.
- Riferimento: queste tabelle vengono replicate su tutti gli shard. Vengono utilizzate per dati di riferimento modificati raramente, come cataloghi di prodotti e codici postali.

  Le query di unione tra tabelle di riferimento e tabelle frammentate possono essere eseguite su shard, eliminando lo spostamento non necessario di dati tra shard e router.

Esistono due modi per creare tabelle illimitate:

- Creazione di tabelle illimitate utilizzando variabili— Utilizzate questo metodo quando desiderate creare nuove tabelle frammentate e di riferimento.
- Conversione di tabelle standard in tabelle illimitate— Utilizzate questo metodo per convertire tabelle standard esistenti in tabelle frammentate e di riferimento.

Forniamo anche schemi di esempio per Aurora SQL Postgre Limitless Database.

Creazione di tabelle illimitate utilizzando variabili

È possibile utilizzare le variabili per creare tabelle frammentate e di riferimento impostando la modalità di creazione delle tabelle. Le tabelle che creerai utilizzeranno quindi questa modalità finché non imposterai una modalità diversa.

Utilizzate le seguenti variabili per creare tabelle frammentate e di riferimento:

- `rds_aurora.limitless_create_table_mode`— Imposta questa variabile di sessione su `sharded` o `reference`. Il valore predefinito di questa variabile è `standard`.
- `rds_aurora.limitless_create_table_shard_key`— Imposta questa variabile di sessione su una matrice di nomi di colonna da utilizzare come chiavi di condivisione. Questa variabile viene ignorata quando `rds_aurora.limitless_create_table_mode` non lo è. `sharded`

  Formatta il valore come `untyped array literal` quando inserisci valori letterali in una colonna di matrice. Per maggiori informazioni, consulta Arrays nella documentazione di SQL Postgre.
- `rds_aurora.limitless_create_table_collocate_with`— Imposta questa variabile di sessione su un nome di tabella specifico per collocare le tabelle appena create con quella tabella.

  Se due o più tabelle vengono suddivise utilizzando la stessa chiave shard, è possibile allineare (collocare) esplicitamente tali tabelle. Quando vengono collocate due o più tabelle, le righe di tali tabelle con gli stessi valori della chiave dello shard vengono posizionate sullo stesso frammento. La collocazione aiuta a limitare alcune operazioni a un singolo shard, il che si traduce in prestazioni migliori.

> **Note**
>
> Tutte le chiavi primarie e univoche devono includere la chiave shard. Ciò significa che la chiave shard è un sottoinsieme della chiave primaria o unica.
> Le tabelle illimitate presentano alcune limitazioni. Per ulteriori informazioni, consulta Limitazioni DDL e altre informazioni per Aurora PostgreSQL Limitless Database.

Argomenti

- Esempi di utilizzo di variabili per creare tabelle illimitate
- Visualizzazioni della tabella del database Aurora Postgre Limitless SQL

Esempi di utilizzo di variabili per creare tabelle illimitate

Gli esempi seguenti mostrano come utilizzare queste variabili per creare tabelle frammentate e di riferimento.

Crea una tabella condivisa denominata`items`, con la chiave shard. `id`

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items(id int, val int, item text);
COMMIT;
```

Crea una tabella condivisa denominata`items`, con una chiave shard composta da e colonne. `item_id` `item_cat`

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
CREATE TABLE items(item_id int, item_cat varchar, val int, item text);
COMMIT;
```

Create una tabella condivisa denominata`item_description`, con una chiave shard composta da `item_id` e `item_cat` colonne, e collocatela con la `items` tabella dell'esempio precedente.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
SET LOCAL rds_aurora.limitless_create_table_collocate_with='items';
CREATE TABLE item_description(item_id int, item_cat varchar, color_id int);
COMMIT;
```

Crea una tabella di riferimento denominata. `colors`

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE colors(color_id int primary key, color varchar);
COMMIT;
```

Per reimpostare la variabile di `rds_aurora.limitless_create_table_mode` sessione su`standard`, utilizzate la seguente istruzione:

```
RESET rds_aurora.limitless_create_table_mode;
```

Dopo aver reimpostato questa variabile, le tabelle vengono create come tabelle standard, che è l'impostazione predefinita. Per ulteriori informazioni sulle tabelle standard, vedereConversione di tabelle standard in tabelle illimitate.

Visualizzazioni della tabella del database Aurora Postgre Limitless SQL

È possibile trovare informazioni sulle tabelle di Limitless Database utilizzando le seguenti viste.

rds_aurora.limitless_tables

La `rds_aurora.limitless_tables` vista contiene informazioni sulle tabelle illimitate e sui relativi tipi.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | standard    | active       | standard    |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     |
 HASH (order_id)
         2 |     18579 | public      | customer    | active       | sharded     |
 HASH (cust_id)
(4 rows)
```

rds_aurora.limitless_table_collocations

La vista contiene informazioni sulle tabelle suddivise `rds_aurora.limitless_table_collocations` collocate. Ad esempio, le `customers` tabelle `orders` e sono collocate e hanno le stesse caratteristiche. `collocation_id` Le `followers` tabelle `users` e sono collocate e hanno la stessa posizione. `collocation_id`

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
          16002 | public      | orders
          16002 | public      | customers
          16005 | public      | users
          16005 | public      | followers
(4 rows)
```

rds_aurora.limitless_table_collocation_distribution

Mostra la distribuzione delle chiavi per ogni collocazione`rds_aurora.limitless_table_collocation_distributions`.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
          16002 |             6 | -9223372036854775808 | -4611686018427387904
          16002 |             5 | -4611686018427387904 |                    0
          16002 |             4 |                    0 |  4611686018427387904
          16002 |             3 |  4611686018427387904 |  9223372036854775807
          16005 |             6 | -9223372036854775808 | -4611686018427387904
          16005 |             5 | -4611686018427387904 |                    0
          16005 |             4 |                    0 |  4611686018427387904
          16005 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Conversione di tabelle standard in tabelle illimitate

È possibile convertire le tabelle standard in tabelle frammentate o di riferimento. Durante la conversione, i dati vengono spostati dalla tabella standard alla tabella distribuita, quindi la tabella standard di origine viene eliminata. I dati vengono spostati utilizzando il `INSERT INTO SELECT FROM` comando.

Indice

- Creazione di tabelle condivise
- Creazione di tabelle collocate
- Creazione di tabelle di riferimento

Creazione di tabelle condivise

È possibile creare tabelle ripartite eseguendo la `rds_aurora.limitless_alter_table_type_sharded` procedura su tabelle standard. Questa procedura richiede una tabella standard e un elenco di colonne, quindi distribuisce la tabella data utilizzando l'elenco di colonne come chiave condivisa. La procedura viene eseguita in modo sincrono e acquisisce un `ACCESS EXCLUSIVE` blocco sulla tabella.

Al termine della procedura, la tabella standard di origine viene eliminata e diventa disponibile una tabella condivisa con lo stesso nome.

La `rds_aurora.limitless_alter_table_type_sharded` procedura utilizza la seguente sintassi:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('schema.table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn']);
```

La procedura richiede i seguenti parametri:

- `schema`— Lo schema del database che contiene la tabella da condividere. Se lo schema non è specificato, la procedura utilizza il`search_path`.
- `table`— La tabella da dividere.
- `shard_key`*n*— Una matrice di colonne della tabella da utilizzare come chiave shard.

  I valori delle chiavi dello shard sono stringhe letterali e pertanto fanno distinzione tra maiuscole e minuscole. Se una chiave shard contiene una virgoletta singola ('), usa un'altra virgoletta singola

per evitarla. Ad esempio, se viene denominata una colonna della tabella`customer's id`, usala `customer''s id` come chiave shard. Non è necessario che le barre rovesciate (\) e le virgolette doppie («) siano sfuggite.

> **Note**
>
> Tutte le chiavi primarie e uniche devono includere la chiave shard. Ciò significa che la chiave shard è un sottoinsieme della chiave primaria o unica.
> Nelle tabelle suddivise, il CHECK vincolo non supporta le espressioni.
> Per ulteriori informazioni, consulta Vincoli.

Per creare una tabella condivisa

L'esempio seguente mostra come creare la tabella `customer` condivisa con la chiave shard. `customer_id`

1. Create la tabella standard.

```
CREATE TABLE customer (customer_id INT PRIMARY KEY NOT NULL, zipcode INT, email
 VARCHAR);
```

2. Converte la tabella standard in una tabella condivisa.

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.customer',
 ARRAY['customer_id']);

postgres=> \d

                  List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
(6 rows)
```

Creazione di tabelle collocate

Se due o più tabelle vengono suddivise utilizzando la stessa chiave shard, è possibile allineare (collocare) in modo esplicito tali tabelle. Quando vengono collocate due o più tabelle, le righe di tali tabelle con gli stessi valori della chiave dello shard vengono posizionate sullo stesso frammento. La collocazione aiuta a limitare alcune operazioni a un singolo shard, il che si traduce in prestazioni migliori.

Si utilizza la `rds_aurora.limitless_alter_table_type_sharded` procedura con la seguente sintassi:

```
postgres=> CALL
 rds_aurora.limitless_alter_table_type_sharded('schema.collocated_table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn'], 'schema.sharded_table');
```

La procedura richiede i seguenti parametri:

- `schema`— Lo schema del database che contiene le tabelle da collocare. Se lo schema non è specificato, la procedura utilizza il`search_path`.
- `collocated_table`— La tabella da collocare.
- `shard_key`*n*— Una matrice di colonne della tabella da usare come chiave shard.

  È necessario utilizzare la stessa chiave shard della tabella condivisa originale, inclusi gli stessi nomi di colonna e tipi di colonna.
- `sharded_table`— La tabella condivisa con cui stai collocando il. `collocated_table`

Per creare una tabella collocata

1. Creare la prima tabella condivisa seguendo la procedura riportata in. Creazione di tabelle condivise
2. Crea la tabella standard per la tabella collocata.

   ```
   CREATE TABLE mytable2 (customer_id INT PRIMARY KEY NOT NULL, column1 INT, column2
    VARCHAR);
   ```
3. Converte la tabella standard in una tabella collocata.

   ```
   postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.mytable2',
   ```

```
ARRAY['customer_id'], 'public.customer');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | mytable2     | partitioned table | postgres_limitless
 public | mytable2_fs1 | foreign table     | postgres_limitless
 public | mytable2_fs2 | foreign table     | postgres_limitless
 public | mytable2_fs3 | foreign table     | postgres_limitless
 public | mytable2_fs4 | foreign table     | postgres_limitless
 public | mytable2_fs5 | foreign table     | postgres_limitless
(12 rows)
```

Creazione di tabelle di riferimento

È possibile creare tabelle di riferimento eseguendo la `rds_aurora.limitless_alter_table_type_reference` procedura su tabelle standard. Questa procedura replica una determinata tabella in tutti gli shard del gruppo di shard DB e modifica il tipo di tabella in riferimento. La procedura viene eseguita in modo sincrono e acquisisce un `ACCESS EXCLUSIVE` blocco sulla tabella.

Al termine della procedura, la tabella standard di origine viene eliminata e diventa disponibile una tabella di riferimento con lo stesso nome.

La `rds_aurora.limitless_alter_table_type_reference` procedura utilizza la seguente sintassi:

```
postgres=> CALL rds_aurora.limitless_alter_table_type_reference('schema.table');
```

La stored procedure richiede i seguenti parametri:

- `schema`— Lo schema del database che contiene la tabella da replicare. Se lo schema non è specificato, la procedura utilizza il`search_path`.

- `table`— La tabella da replicare.

> **Note**
>
> La tabella standard da cui si crea la tabella di riferimento deve avere una chiave primaria.
> Nelle tabelle di riferimento, il CHECK vincolo non supporta le espressioni.
> La funzione precedente,`limitless_table_alter_type_reference`, è obsoleta.

Per creare una tabella di riferimento

L'esempio seguente mostra come creare la tabella `zipcodes` di riferimento.

1. Create la tabella standard.

```
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);
```

2. Converte la tabella standard in una tabella di riferimento.

```
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcodes');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | zipcodes     | foreign table     | postgres_limitless
(7 rows)
```

   L'output mostra la tabella `customer` frammentata e la tabella `zipcodes` di riferimento.

Schemi di esempio di Aurora Postgree Limitless Database SQL

Forniamo i seguenti schemi di esempio per Aurora SQL Postgre Limitless Database:

- Schema di esempio Limitless E-Commerce
- pgbench senza limiti

È possibile utilizzare questi schemi per creare rapidamente un database di esempio e caricare i dati nelle tabelle del database Aurora SQL Postgre Limitless. Per ulteriori informazioni, consulta il repository. GitHub

Caricamento dei dati nel database Aurora SQL Postgre Limitless

È possibile caricare i dati nelle tabelle del database Aurora Postgre SQL Limitless utilizzando il `COPY` comando o l'utilità di caricamento dei dati.

> **Note**
>
> È possibile caricare i dati in tabelle standard, frammentate e di riferimento.

Indice

- Utilizzo del COPY comando con Aurora SQL Postgre Limitless Database
  - Utilizzo del COPY comando per caricare i dati nel database Aurora SQL Postgre Limitless
    - Suddivisione dei dati in più file
  - Utilizzo del COPY comando per copiare i dati di Limitless Database in un file
- Utilizzo dell'utilità di caricamento dati Aurora PostgreSQL Limitless Database
  - Limitazioni
  - Prerequisiti
  - Preparazione del database di origine
  - Preparazione del database di destinazione
  - Creazione di credenziali del database
    - Crea le credenziali del database di origine
    - Crea le credenziali del database di destinazione
  - Configurazione dell'autenticazione del database e dell'accesso alle risorse tramite uno script
    - Script di installazione per l'utilità di caricamento dei dati
    - Output dallo script di configurazione dell'utilità di caricamento dei dati
    - Eliminazione delle risorse non riuscite
  - Configurazione manuale dell'autenticazione del database e dell'accesso alle risorse
    - Creazione del file gestito dal cliente AWS KMS key
    - Creazione dei segreti del database
    - Creazione del ruolo IAM
    - Aggiornamento del file gestito dal cliente AWS KMS key
    - Aggiungere le politiche di autorizzazione dei IAM ruoli

- Caricamento di dati da un cluster Aurora Postgre SQL DB o RDS per un'istanza di Postgre DB SQL
- Monitoraggio del caricamento dei dati
  - Elenco dei lavori di caricamento dei dati
  - Visualizzazione dei dettagli dei lavori di caricamento dei dati utilizzando l'ID del lavoro
  - Monitoraggio del gruppo di CloudWatch log Amazon
  - Monitoraggio RDS degli eventi
- Annullamento del caricamento dei dati

Utilizzo del COPY comando con Aurora SQL Postgre Limitless Database

È possibile utilizzare la funzionalità\ copy dell'psqlutilità per importare ed esportare dati dal database Aurora Postgre Limitless SQL

Utilizzo del COPY comando per caricare i dati nel database Aurora SQL Postgre Limitless

Aurora Postgre SQL Limitless Database è compatibile con la funzionalità\ copy dell'utilità per l'importazione dei psql dati.

In Limitless Database come in Aurora SQL Postgre, quanto segue non è supportato:

- SSHAccesso diretto alle istanze DB: non è possibile copiare un file di dati (ad esempio in formato.csv) sull'host dell'istanza DB ed eseguirlo dal file. COPY
- Utilizzo di file locali sull'istanza DB: utilizza e. COPY ... FROM STDIN COPY ... TO STDOUT

Il COPY comando in Postgre SQL ha opzioni per lavorare con i file locali (FROM/TO) e trasmettere dati utilizzando una connessione tra il client e il server (). STDIN/STDOUT Per ulteriori informazioni, consultate la documentazione di COPYPostgre. SQL

Il \copy comando dell'SQLpsqlutilità Postgre funziona con i file locali sul computer su cui viene eseguito il client. psql Richiama il rispettivo COPY ... FROM STDOUT comando COPY ... FROM STDIN o sul server remoto (ad esempio, Limitless Database) a cui ci si connette. Legge STDIN o scrive dati dal file locale su di esso. STDOUT

Suddivisione dei dati in più file

I dati vengono archiviati su più shard in Aurora SQL Postgre Limitless Database. Per velocizzare il caricamento dei dati\copy, è possibile suddividerli in più file. Quindi importa in modo indipendente per ogni file di dati eseguendo \copy comandi separati in parallelo.

Ad esempio, avete un file di dati di input in CSV formato con 3 milioni di righe da importare. Puoi dividere il file in blocchi contenenti ciascuno 200.000 righe (15 blocchi):

```
split -l200000 data.csv data_ --additional-suffix=.csv -d
```

Ciò si traduce in file. data_00.csv data_14.csv È quindi possibile importare i dati utilizzando 15 \copy comandi paralleli, ad esempio:

```
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table from '/tmp/data_00.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_01.csv';" postgres_limitless &
...
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_13.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_14.csv';" postgres_limitless
```

Utilizzando questa tecnica, la stessa quantità di dati viene importata circa 10 volte più velocemente rispetto all'utilizzo di un singolo `\copy` comando.

Utilizzo del COPY comando per copiare i dati di Limitless Database in un file

È possibile utilizzare il comando\ copy per copiare i dati da una tabella illimitata a un file, come mostrato nell'esempio seguente:

```
postgres_limitless=> \copy test_table TO '/tmp/test_table.csv' DELIMITER ',' CSV
 HEADER;
```

Utilizzo dell'utilità di caricamento dati Aurora PostgreSQL Limitless Database

Aurora fornisce un'utilità per caricare i dati direttamente in Limitless Database da un cluster Aurora PostgreSQL DB o da un'istanza DB RDS per PostgreSQL.

Per utilizzare l'utilità di caricamento dei dati, è necessario eseguire le seguenti operazioni:

1. Prerequisiti
2. Preparazione del database di origine
3. Preparazione del database di destinazione
4. Creazione di credenziali del database
5. Una delle seguenti:
   - Configurazione dell'autenticazione del database e dell'accesso alle risorse tramite uno script (consigliato)
   - Configurazione manuale dell'autenticazione del database e dell'accesso alle risorse
6. Caricamento di dati da un cluster Aurora Postgre SQL DB o RDS per un'istanza di Postgre DB SQL

Limitazioni

L'utilità di caricamento dei dati presenta le seguenti limitazioni:

- I seguenti tipi di dati non sono supportati: `enum` `ARRAYBOX`,`CIRCLE`,`LINE`,`LSEG`,`PATH`,`PG_LSN`,`PG_SNAPSHOT`,`POLYGON`,`TSQUERY`,`TSVECTOR`, e`TXID_SNAPSHOT`.
- Gli zeri iniziali (`0`) vengono rimossi dal tipo di `VARBIT` dati durante il caricamento.
- La migrazione dei dati non riesce quando sono presenti chiavi primarie composite nelle tabelle di origine.
- La migrazione dei dati non riesce quando sono presenti chiavi esterne nelle tabelle di destinazione.
- Il caricamento di dati dai cluster DB RDS for PostgreSQL Multi-AZ non è supportato.

Prerequisiti

L'utilità di caricamento dei dati presenta i seguenti prerequisiti:

- Il database di origine utilizza Aurora PostgreSQL o RDS per PostgreSQL versione 11.x e successive.
- Il database di origine si trova nello stesso gruppo di shard DB e nel gruppo di shard DB di destinazione. Account AWS Regione AWS
- Il cluster o l'istanza DB di origine si trova nello `available` stato.
- Le tabelle del database di origine e del database illimitato hanno gli stessi nomi di tabella, nomi di colonna e tipi di dati di colonna.
- Le tabelle di origine e di destinazione hanno chiavi primarie che utilizzano le stesse colonne e gli stessi ordini di colonne.
- È necessario disporre di un ambiente per la connessione a un database illimitato per eseguire i comandi di caricamento dei dati. I comandi disponibili sono i seguenti:
  - `rds_aurora.limitless_data_load_start`
  - `rds_aurora.limitless_data_load_cancel`
- Per CDC:
  - Sia il database di origine che il gruppo di shard DB di destinazione devono utilizzare lo stesso gruppo di sottorete DB, lo stesso gruppo di sicurezza VPC e la stessa porta del database. Queste configurazioni riguardano le connessioni di rete sia al database di origine che ai router del gruppo di shard DB.
  - È necessario abilitare la replica logica sul database di origine. L'utente del database di origine deve disporre dei privilegi per leggere la replica logica.

Preparazione del database di origine

Per accedere al database di origine per il caricamento dei dati, è necessario consentire il traffico di rete in entrata. Esegui questa procedura.

Per consentire il traffico di rete verso il database di origine

1. Accedi a AWS Management Console e apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Vai alla pagina dei gruppi di sicurezza.
3. Scegli l'ID del gruppo di sicurezza per il gruppo di sicurezza utilizzato dal cluster o dall'istanza DB di origine.

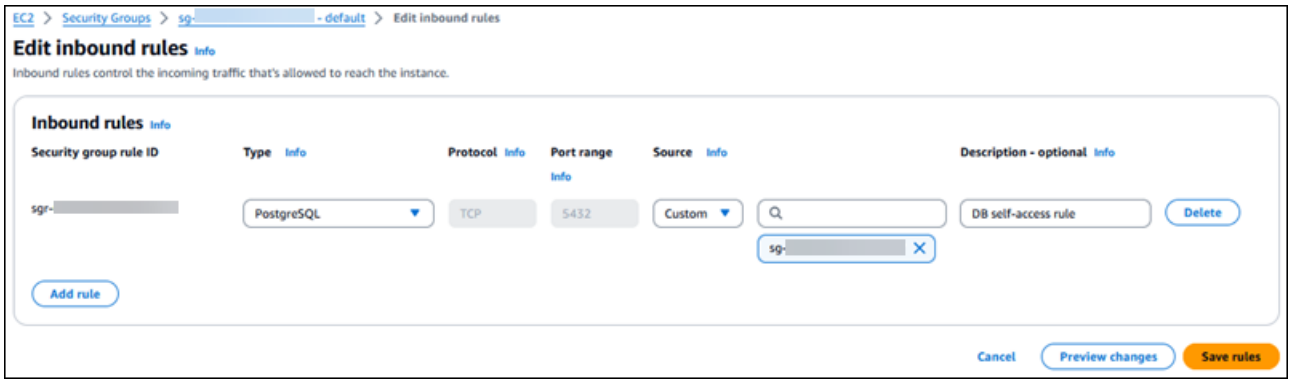
   Ad esempio, l'ID del gruppo di sicurezza è`sg-056a84f1712b77926`.

4. Nella scheda Regole in entrata:

   a. Sceglere Edit inbound rules (Modifica regole in entrata).

   b. Aggiungi una nuova regola in entrata per il cluster o l'istanza DB di origine:

      - Intervallo di porte: porta del database per il database di origine, in genere `5432`
      - ID del gruppo di sicurezza, `sg-056a84f1712b77926` in questo esempio

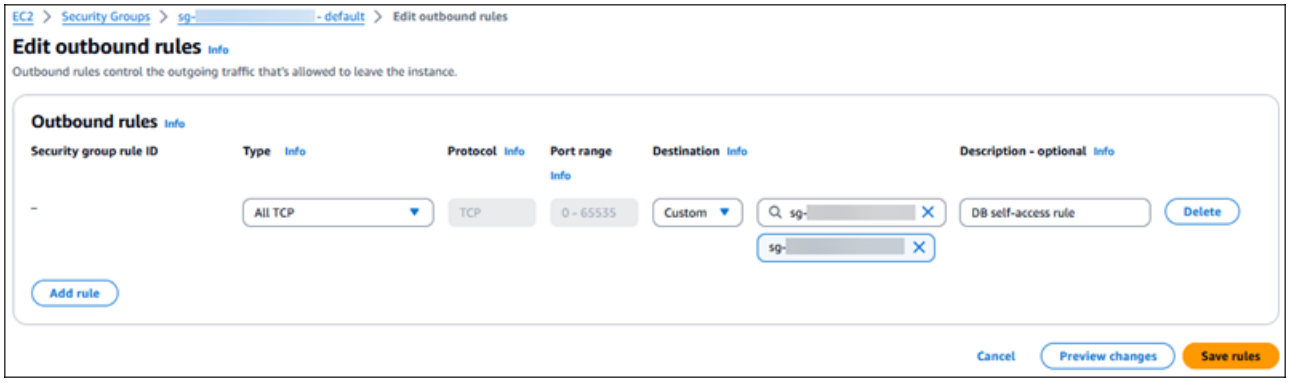


5. Nella scheda Regole in uscita:

   a. Scegli Edit outbound rules (Modifica regole in uscita).

   b. Aggiungi una nuova regola in uscita per il cluster o l'istanza DB di origine:

      - Porta del database: `All traffic` (include le porte`0-65535`)
      - ID del gruppo di sicurezza, `sg-056a84f1712b77926` in questo esempio



6. Accedi a AWS Management Console e apri la console Amazon VPC all'indirizzo. https://console.aws.amazon.com/vpc/

7. Vai alla ACLs pagina Rete.

8. Aggiungi la configurazione ACL di rete predefinita come indicato in ACL di rete predefinito.

Preparazione del database di destinazione

Segui le procedure Creazione di tabelle del database Aurora SQL Postgre Limitless per creare le tabelle di destinazione nel gruppo di shard DB.

Le tabelle di destinazione devono avere gli stessi schemi, nomi di tabella e chiavi primarie delle tabelle di origine.

Creazione di credenziali del database

È necessario creare utenti del database nei database di origine e di destinazione e concedere loro i privilegi necessari. Per ulteriori informazioni, vedere CREATE USER e GRANT nella documentazione di PostgreSQL.

Crea le credenziali del database di origine

All'utente del database di origine viene passato il comando per avviare il caricamento. Questo utente deve disporre dei privilegi per eseguire la replica dal database di origine.

1. Utilizza l'utente principale del database (o un altro utente con il `rds_superuser` ruolo) per creare un utente del database di origine con privilegi`LOGIN`.

   ```
   CREATE USER source_db_username WITH PASSWORD 'source_db_user_password';
   ```

2. Assegna il `rds_superuser` ruolo all'utente del database di origine.

   ```
   GRANT rds_superuser to source_db_username;
   ```

3. Se utilizzi la `full_load_and_cdc` modalità, concedi il `rds_replication` ruolo all'utente del database di origine. Il ruolo `rds_replication` fornisce le autorizzazioni per gestire gli slot logici e per eseguire lo streaming dei dati utilizzando gli slot logici.

   ```
   GRANT rds_replication to source_db_username;
   ```

Crea le credenziali del database di destinazione

L'utente del database di destinazione deve disporre dell'autorizzazione per scrivere nelle tabelle di destinazione nel gruppo di shard DB.

1. Utilizzate l'utente principale del database (o un altro utente con il `rds_superuser` ruolo) per creare un utente del database di destinazione con `LOGIN` privilegi.

```
CREATE USER destination_db_username WITH PASSWORD 'destination_db_user_password';
```

2. Assegna il `rds_superuser` ruolo all'utente del database di destinazione.

```
GRANT rds_superuser to destination_db_username;
```

Configurazione dell'autenticazione del database e dell'accesso alle risorse tramite uno script

Lo script di configurazione crea un ruolo gestito dal cliente AWS KMS key, un AWS Identity and Access Management (IAM) ruolo e due AWS Secrets Manager segreti.

Eseguite i seguenti passaggi per utilizzare lo script di configurazione:

1. Assicurati di averlo AWS CLI installato e configurato con Account AWS le tue credenziali.
2. Installa il processore da riga di `jq` comandoJSON. Per ulteriori informazioni, vedere jqlang/jq.
3. Copia il file data_loading_script.zip sul tuo computer ed estrai il file da esso. `data_load_aws_setup_script.sh`
4. Modifica lo script per sostituire le variabili segnaposto con i valori appropriati per quanto segue:
   - Il tuo Account AWS
   - La Regione AWS
   - Credenziali del database di origine
   - Credenziali del database di destinazione
5. Apri un nuovo terminale sul tuo computer ed esegui il seguente comando:

```
bash ./data_load_aws_setup_script.sh
```

Script di installazione per l'utilità di caricamento dei dati

Forniamo qui il testo del `data_load_aws_setup_script.sh` file come riferimento.

```
#!/bin/bash
# Aurora Limitless data loading - AWS resources setup script #
# Set up the account credentials in advance. #
# Update the following script variables. #

###################################
#### Start of variable section ####

ACCOUNT_ID="12-digit_AWS_account_ID"
REGION="AWS_Region"
DATE=$(date +'%m%d%H%M%S')
RANDOM_SUFFIX="${DATE}"
SOURCE_SECRET_NAME="secret-source-${DATE}"
```

```
SOURCE_USERNAME="source_db_username"
SOURCE_PASSWORD="source_db_password"
DESTINATION_SECRET_NAME="secret-destination-${DATE}"
DESTINATION_USERNAME="destination_db_username"
DESTINATION_PASSWORD="destination_db_password"
DATA_LOAD_IAM_ROLE_NAME="aurora-data-loader-${RANDOM_SUFFIX}"
TMP_WORK_DIR="./tmp_data_load_aws_resource_setup/"

#### End of variable section ####
################################

# Main logic start
echo "DATE - [${DATE}]"
echo "RANDOM_SUFFIX - [${RANDOM_SUFFIX}]"
echo 'START!'

mkdir -p $TMP_WORK_DIR

# Create the symmetric KMS key for encryption and decryption.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_key_response.txt"
aws kms create-key --region $REGION | tee $TMP_FILE_PATH
KMS_KEY_ARN=$(cat $TMP_FILE_PATH | jq -r '.KeyMetadata.Arn')
aws kms create-alias \
    --alias-name alias/"${DATA_LOAD_IAM_ROLE_NAME}-key" \
    --target-key-id $KMS_KEY_ARN \
    --region $REGION

# Create the source secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_source_secret_response.txt"
aws secretsmanager create-secret \
    --name $SOURCE_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$SOURCE_USERNAME\",\"password\":\"$SOURCE_PASSWORD
\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
SOURCE_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the destination secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_destination_secret_response.txt"
aws secretsmanager create-secret \
    --name $DESTINATION_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
```

```
    --secret-string "{\"username\":\"$DESTINATION_USERNAME\",\"password\":
\"$DESTINATION_PASSWORD\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
DESTINATION_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the RDS trust policy JSON file.
# Use only rds.amazonaws.com for RDS PROD use cases.
TRUST_POLICY_PATH="${TMP_WORK_DIR}rds_trust_policy.json"
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}' > $TRUST_POLICY_PATH

# Create the IAM role.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_iam_role_response.txt"
aws iam create-role \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --assume-role-policy-document "file://${TRUST_POLICY_PATH}" \
    --tags Key=assumer,Value=aurora_limitless_table_data_load \
    --region $REGION \
    | tee $TMP_FILE_PATH
IAM_ROLE_ARN=$(cat $TMP_FILE_PATH | jq -r '.Role.Arn')

# Create the permission policy JSON file.
PERMISSION_POLICY_PATH="${TMP_WORK_DIR}data_load_permission_policy.json"
permission_json_policy=$(cat &lt;&lt;EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "$SOURCE_SECRET_ARN",
                "$DESTINATION_SECRET_ARN"
            ]
        },
        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "$KMS_KEY_ARN"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
```

```
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $permission_json_policy > $PERMISSION_POLICY_PATH

# Add the inline policy.
aws iam put-role-policy \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --policy-name aurora-limitless-data-load-policy \
    --policy-document "file://${PERMISSION_POLICY_PATH}" \
    --region $REGION

# Create the key policy JSON file.
KEY_POLICY_PATH="${TMP_WORK_DIR}data_load_key_policy.json"
key_json_policy=$(cat &lt;&lt;EOF
{
    "Id": "key-aurora-limitless-data-load-$RANDOM_SUFFIX",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "$IAM_ROLE_ARN"
            },
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
```

```
            "Resource": "*"
        }
    ]
}
EOF
)
echo $key_json_policy > $KEY_POLICY_PATH

# Add the key policy.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_put_key_policy_response.txt"
sleep 10 # sleep 10 sec for IAM role ready
aws kms put-key-policy \
    --key-id $KMS_KEY_ARN \
    --policy-name default \
    --policy "file://${KEY_POLICY_PATH}" \
    --region $REGION \
    | tee $TMP_FILE_PATH

echo 'DONE!'

echo "ACCOUNT_ID : [${ACCOUNT_ID}]"
echo "REGION : [${REGION}]"
echo "RANDOM_SUFFIX : [${RANDOM_SUFFIX}]"
echo "IAM_ROLE_ARN : [${IAM_ROLE_ARN}]"
echo "SOURCE_SECRET_ARN : [${SOURCE_SECRET_ARN}]"
echo "DESTINATION_SECRET_ARN : [${DESTINATION_SECRET_ARN}]"

# Example of a successful run:
# ACCOUNT_ID : [012345678912]
# REGION : [ap-northeast-1]
# RANDOM_SUFFIX : [0305000703]
# IAM_ROLE_ARN : [arn:aws:iam::012345678912:role/aurora-data-loader-0305000703]
# SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-source-0305000703-yQDtow]
# DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-destination-0305000703-5d5Jy8]

# If you want to manually clean up failed resource,
# please remove them in the following order:
# 1. IAM role.
 # aws iam delete-role-policy --role-name Test-Role --policy-name ExamplePolicy --
region us-east-1
# aws iam delete-role --role-name Test-Role --region us-east-1
# 2. Source and destination secrets.
```

```
# aws secretsmanager delete-secret --secret-id MyTestSecret --force-delete-without-
recovery --region us-east-1
# 3. KDM key.
# aws kms schedule-key-deletion --key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab --pending-window-in-days 7
 --region us-east-1
```

Output dallo script di configurazione dell'utilità di caricamento dei dati

L'esempio seguente mostra l'output di un'esecuzione riuscita dello script.

```
% bash ./data_load_aws_setup_script.sh
DATE - [0305000703]
RANDOM_SUFFIX - [0305000703]
START!
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:ap-
northeast-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-03-05T00:07:49.852000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow",
    "Name": "secret-source-0305000703",
    "VersionId": "a017bebe-a71b-4220-b923-6850c2599c26"
}
{
```

```
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
destination-0305000703-5d5Jy8",
    "Name": "secret-destination-0305000703",
    "VersionId": "32a1f989-6391-46b1-9182-f65d242f5eb6"
}
{
    "Role": {
        "Path": "/",
        "RoleName": "aurora-data-loader-0305000703",
        "RoleId": "AROAYPX63ITQOYORQSC6U",
        "Arn": "arn:aws:iam::123456789012:role/aurora-data-loader-0305000703",
        "CreateDate": "2024-03-05T00:07:54+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "rds.amazonaws.com"
                        ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "assumer",
                "Value": "aurora_limitless_table_data_load"
            }
        ]
    }
}
DONE!
ACCOUNT_ID : [123456789012]
REGION : [ap-northeast-1]
RANDOM_SUFFIX : [0305000703]
IAM_ROLE_ARN : [arn:aws:iam::123456789012:role/aurora-data-loader-0305000703]
SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow]
DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:123456789012:secret:secret-destination-0305000703-5d5Jy8]
```

Eliminazione delle risorse non riuscite

Se desideri pulire manualmente le risorse guaste, rimuovile nel seguente ordine:

1. IAMruolo, ad esempio:

```
aws iam delete-role-policy \
--role-name Test-Role \
--policy-name ExamplePolicy

aws iam delete-role \
--role-name Test-Role
```

2. Segreti di origine e destinazione, ad esempio:

```
aws secretsmanager delete-secret \
--secret-id MyTestSecret \
--force-delete-without-recovery
```

3. KMSchiave, ad esempio:

```
aws kms schedule-key-deletion \
--key-id arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab
 \
--pending-window-in-days 7
```

Quindi puoi riprovare lo script.

Configurazione manuale dell'autenticazione del database e dell'accesso alle risorse

Il processo manuale per configurare l'autenticazione del database e l'accesso alle risorse prevede i seguenti passaggi:

1. Creazione del file gestito dal cliente AWS KMS key
2. Aggiungere le politiche di autorizzazione dei IAM ruoli
3. Creazione dei segreti del database
4. Creazione del ruolo IAM
5. Aggiornamento del file gestito dal cliente AWS KMS key

Questo processo è facoltativo ed esegue le stesse attività di cui sopraConfigurazione dell'autenticazione del database e dell'accesso alle risorse tramite uno script. Si consiglia di utilizzare lo script.

Creazione del file gestito dal cliente AWS KMS key

Segui le procedure descritte in Creazione di chiavi di crittografia simmetriche per creare una chiave gestita dal clienteKMS. Puoi anche utilizzare una chiave esistente se soddisfa questi requisiti.

Per creare una chiave gestita dal cliente KMS

1. Accedi a AWS Management Console e apri la AWS KMS console su https://console.aws.amazon.com /kms.
2. Vai alla pagina delle chiavi gestite dal cliente.
3. Scegliere Create key (Crea chiave).
4. Nella pagina Configura chiave:
    a. Per Tipo di chiave, seleziona Symmetric.
    b. Per Utilizzo della chiave, seleziona Crittografa e decrittografa.
    c. Scegli Next (Successivo).
5. Nella pagina Aggiungi etichette, inserisci un alias come**`limitless`**, quindi scegli Avanti.
6. Nella pagina Definisci le autorizzazioni amministrative chiave, assicurati che la casella di controllo Consenti agli amministratori chiave di eliminare questa chiave sia selezionata, quindi scegli Avanti.
7. Nella pagina Definisci le autorizzazioni per utilizzare le chiavi scegli Avanti.

8. Nella pagina Review (Rivedi), scegliere Finish (Fine).

   La politica chiave verrà aggiornata in un secondo momento.

Registra gli Amazon Resource Names (ARN) della KMS chiave da utilizzareAggiungere le politiche di autorizzazione dei IAM ruoli.

Per informazioni sull'utilizzo di per AWS CLI creare la KMS chiave gestita dal cliente, consulta create-key e create-alias.

Creazione dei segreti del database

Per consentire all'utilità di caricamento dei dati di accedere alle tabelle del database di origine e di destinazione, è necessario creare due segreti in AWS Secrets Manager: uno per il database di origine e uno per il database di destinazione. Questi segreti memorizzano i nomi utente e le password per accedere ai database di origine e di destinazione.

Segui le procedure in Creare un AWS Secrets Manager segreto per creare i segreti della coppia chiave-valore.

Per creare i segreti del database

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Nella pagina Scegli il tipo di segreto:

   a. Per Tipo segreto, seleziona Altro tipo di segreto.
   b. Per le coppie chiave/valore, scegliete la scheda Testo semplice.
   c. Immettete il JSON codice seguente, dove *sourcedbreader* e *sourcedbpassword* sono le credenziali dell'utente del database di origine. Crea le credenziali del database di origine

      ```
      {
          "username":"sourcedbreader",
          "password":"sourcedbpassword"
      }
      ```

   d. Per Chiave di crittografia, scegli la KMS chiave in cui hai creatoCreazione del file gestito dal cliente AWS KMS key, ad esempio`limitless`.
   e. Scegli Next (Successivo).

4. Nella pagina Configura segreto, inserisci un nome segreto, ad esempio**`source_DB_secret`**, quindi scegli Avanti.
5. Nella pagina Configura rotazione - opzionale, scegli Avanti.
6. Nella pagina Review (Rivedi), scegli Store (Archivia).
7. Ripeti la procedura per il segreto del database di destinazione:

   a. Immettere il JSON codice seguente, dove *`destinationdbwriter`* e *`destinationdbpassword`* sono le credenziali dell'utente del database di Crea le credenziali del database di destinazione destinazione.

   ```
   {
       "username":"destinationdbwriter",
       "password":"destinationdbpassword"
   }
   ```

   b. Inserisci un nome segreto, ad esempio**`destination_DB_secret`**.

Registra ARNs i segreti da utilizzareAggiungere le politiche di autorizzazione dei IAM ruoli.

Creazione del ruolo IAM

Il caricamento dei dati richiede l'accesso alle AWS risorse. Per fornire l'accesso, è necessario creare il `aurora-data-loader` IAM ruolo seguendo le procedure riportate in Creazione di un ruolo per delegare le autorizzazioni a un IAM utente.

Per creare il ruolo IAM

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo. https://console.aws.amazon.com/iam/
2. Vai alla pagina Ruoli.
3. Scegliere Crea ruolo.
4. Nella pagina Seleziona entità attendibile:

   a. Per il tipo di entità affidabile, seleziona Criteri di attendibilità personalizzati.

   b. Inserisci il seguente JSON codice per la politica di fiducia personalizzata:

   ```
   {
       "Version": "2012-10-17",
   ```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

c. Scegli Next (Successivo).

5. Nella pagina Add permissions (Aggiungi autorizzazioni), scegli Next (Successivo).
6. Nella pagina Nome, rivedi e crea:

   a. Per il nome del ruolo, inserisci **`aurora-data-loader`** o un altro nome che preferisci.

   b. Scegli Aggiungi tag e inserisci il seguente tag:

   - Chiave: **`assumer`**
   - Value (Valore): **`aurora_limitless_table_data_load`**

> **Important**
>
> L'Aurora Postgre SQL Limitless Database può assumere solo un IAM ruolo con questo tag.

   c. Scegliere Crea ruolo.

Aggiornamento del file gestito dal cliente AWS KMS key

Segui le procedure in Modifica di una politica chiave per aggiungere il IAM ruolo `aurora-data-loader` alla politica chiave predefinita.

Per aggiungere il IAM ruolo alla politica chiave

1. Accedi a AWS Management Console e apri la AWS KMS console su https://console.aws.amazon.com/kms.

2. Vai alla pagina delle chiavi gestite dal cliente.
3. Scegli la KMS chiave che hai creatoCreazione del file gestito dal cliente AWS KMS key, ad esempio`limitless`.
4. Nella scheda Politica chiave, per Utenti chiave, scegli Aggiungi.
5. Nella finestra Aggiungi utenti chiave, seleziona, ad esempioCreazione del ruolo IAM, il nome del IAM ruolo in cui hai creato aurora-data-loader.
6. Scegli Aggiungi.

Aggiungere le politiche di autorizzazione dei IAM ruoli

È necessario aggiungere politiche di autorizzazione al IAM ruolo creato. Ciò consente all'utilità di caricamento dei dati del database Aurora Postgre SQL Limitless di accedere alle AWS risorse correlate per creare connessioni di rete e recuperare i segreti delle credenziali DB di origine e destinazione.

Per ulteriori informazioni, consulta Modifica un ruolo.

Per aggiungere le politiche di autorizzazione

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Vai alla pagina Ruoli.
3. Scegli, ad esempioCreazione del ruolo IAM, il IAM ruolo in cui hai creato aurora-data-loader.
4. Nella scheda Autorizzazioni, per Politiche di autorizzazione scegli Aggiungi autorizzazioni, quindi Crea politica in linea.
5. Nella pagina Specificare le autorizzazioni, scegli l'editor. JSON
6. Copia e incolla il seguente modello nell'JSONeditor, sostituendo i segnaposto con i segreti e ARNs la chiave del tuo database. KMS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
```

```
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:123456789012:secret:source_DB_secret-
ABC123",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:destination_DB_secret-456DEF"
            ]
        },        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/aa11bb22-####-####-
####-fedcba123456"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
```

```
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Verifica la presenza di errori e correggili.
8. Scegli Next (Successivo).
9. Nella pagina Rivedi e crea, inserisci il nome di una politica**`data_loading_policy`**, ad esempio, quindi scegli Crea politica.

Caricamento di dati da un cluster Aurora Postgre SQL DB o RDS per un'istanza di Postgre DB SQL

Dopo aver completato la configurazione delle risorse e dell'autenticazione, connettiti all'endpoint del cluster e richiama la `rds_aurora.limitless_data_load_start` stored procedure da un database illimitato, ad esempio. `postgres_limitless` Il database illimitato è un database del gruppo di shard DB in cui desideri migrare i dati.

Questa funzione si connette in modo asincrono in background al database di origine specificato nel comando, legge i dati dall'origine e li carica sugli shard. Per prestazioni migliori, i dati vengono caricati utilizzando thread paralleli. La funzione recupera un'istantanea della point-in-time tabella eseguendo un `SELECT` comando per leggere i dati delle tabelle fornite nel comando.

È possibile caricare i dati in tabelle frammentate, di riferimento e standard.

È possibile caricare dati a livello di database, schema o tabella nelle `rds_aurora.limitless_data_load_start` chiamate.

- Database: è possibile caricare un database alla volta in ogni chiamata, senza limiti allo schema o al numero di tabelle all'interno del database.
- Schema: è possibile caricare un massimo di 15 schemi in ogni chiamata, senza limiti al numero di tabelle all'interno di ogni schema.
- Tabella: è possibile caricare un massimo di 15 tabelle in ogni chiamata.

> **Note**
>
> Questa funzionalità non utilizza le RDS istantanee di Amazon o point-in-time l'isolamento del database. Per garantire la coerenza tra le tabelle, consigliamo di clonare il database di origine e di utilizzare il database clonato come origine.

La stored procedure utilizza la seguente sintassi:

```
CALL rds_aurora.limitless_data_load_start('source_type',
    'source_DB_cluster_or_instance_ID',
    'source_database_name',
    'streaming_mode',
    'data_loading_IAM_role_arn',
    'source_DB_secret_arn',
    'destination_DB_secret_arn',
    'ignore_primary_key_conflict_boolean_flag',
    'is_dry_run',
    (optional parameter) schemas/tables => ARRAY['name1', 'name2', ...]);
```

I parametri di input sono i seguenti:

- `source_type`— Il tipo di fonte: `aurora_postgresql` o `rds_postgresql`
- `source_DB_cluster_or_instance_ID`— L'identificatore del cluster Aurora Postgre SQL DB di origine o RDS per l'identificatore di istanza Postgre DB SQL
- `source_database_name`— Il nome del database di origine, ad esempio *`postgres`*
- `streaming_mode`— Se includere l'acquisizione dei dati di modifica (CDC): `full_load` o `full_load_and_cdc`
- `data_loading_IAM_role_arn`— Il IAM ruolo di Amazon Resource Name (ARN) per `aurora-data-loader`
- `source_DB_secret_arn`— Il segreto del database di origine ARN
- `destination_DB_secret_arn`— Il segreto del DB di destinazione ARN
- `ignore_primary_key_conflict_boolean_flag`— Se continuare se si verifica un conflitto tra le chiavi primarie:
  - Se impostato su`true`, il caricamento dei dati ignora le nuove modifiche per le righe con un conflitto di chiave primaria.
  - Se impostato su`false`, il caricamento dei dati sovrascrive le righe esistenti nelle tabelle di destinazione quando si verifica un conflitto di chiavi primarie.
- `is_dry_run`— Se verificare che il processo di caricamento dei dati possa connettersi ai database di origine e di destinazione:
  - Se impostato su`true`, verifica le connessioni senza caricare dati
  - Se impostato su`false`, carica i dati

- (opzionale) `schemas` o `tables` — Una matrice di schemi o tabelle da caricare. Puoi specificare uno dei seguenti:
  - Un elenco di tabelle nel formato `tables => ARRAY['schema1.table1', 'schema1.table2', 'schema2.table1', ...]`
  - Un elenco di schemi nel formato `schemas => ARRAY['schema1', 'schema2', ...]`

  Se non si include questo parametro, viene migrato l'intero database di origine specificato.

Il parametro di output è l'ID del lavoro con un messaggio.

L'esempio seguente mostra come utilizzare la `rds_aurora.limitless_data_load_start` stored procedure per caricare dati da un cluster Aurora SQL Postgre DB.

```
CALL rds_aurora.limitless_data_load_start('aurora_postgresql',
    'my-db-cluster',
    'postgres',
    'full_load_and_cdc',
    'arn:aws:iam::123456789012:role/aurora-data-loader-8f2c66',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-source-8f2c66-EWrr0V',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-destination-8f2c66-
d04fbD',
    'true',
    'false',
    tables => ARRAY['public.customer', 'public.order', 'public.orderdetails']);

INFO: limitless data load job id 1688761223647 is starting.
```

Monitoraggio del caricamento dei dati

Aurora Postgre SQL Limitless Database offre diversi modi per monitorare i processi di caricamento dei dati:

- Elenco dei lavori di caricamento dei dati
- Visualizzazione dei dettagli dei lavori di caricamento dei dati utilizzando l'ID del lavoro
- Monitoraggio del gruppo di CloudWatch log Amazon
- Monitoraggio RDS degli eventi

Elenco dei lavori di caricamento dei dati

È possibile connettersi all'endpoint del cluster e utilizzare la `rds_aurora.limitless_data_load_jobs` vista per elencare i lavori di caricamento dei dati.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_jobs LIMIT 6;

    job_id     |  status   | message |     source_db_identifier      | source_db_name |
 full_load_complete_time |
   progress_details                                                                  |
     start_time       |   last_updated_time    |  streaming_mode   | source_engine_type
 | ignore_primary_key_conflict | is_dryrun
--------------+-----------+---------+-------------------------------+---------------
+-------------------------
+-------------------------------------------------------------------------------------------------------------------------------
+-----------------------+-----------------------+------------------
+--------------------+-----------------------------+-----------
 1725697520693 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:48:15+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "9
 of 9 tables loaded", "COMPLETED_AT": "2024/09/07 08:48:15+00", "RECORDS_MIGRATED":
 600003}} | 2024-09-07 08:47:13+00 | 2024-09-07 08:48:15+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696114225 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:24:20+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "3
 of 3 tables loaded", "COMPLETED_AT": "2024/09/07 08:24:20+00", "RECORDS_MIGRATED":
 200001}} | 2024-09-07 08:23:56+00 | 2024-09-07 08:24:20+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696067630 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:23:45+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "6
 of 6 tables loaded", "COMPLETED_AT": "2024/09/07 08:23:45+00", "RECORDS_MIGRATED":
```

```
 400002}} | 2024-09-07 08:23:10+00 | 2024-09-07 08:23:45+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725694221753 | CANCELED  |         | persistent-kdm-auto-source-01 | postgres
    |                         | {}

           | 2024-09-07 07:31:18+00 | 2024-09-07 07:51:49+00 | full_load_and_cdc |
 aurora_postgresql  | t                           | f
 1725691698210 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:51+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:51+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:42+00 | 2024-09-07 07:10:52+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725691695049 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:48+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:48+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:41+00 | 2024-09-07 07:10:48+00 | full_load         |
 aurora_postgresql  | t                           | f
(6 rows)
```

I Job record vengono eliminati dopo 90 giorni.

Visualizzazione dei dettagli dei lavori di caricamento dei dati utilizzando l'ID del lavoro

Se conosci un ID di lavoro, puoi connetterti all'endpoint del cluster e utilizzare la `rds_aurora.limitless_data_load_job_details` vista per visualizzare i dettagli di quel processo di caricamento dei dati, inclusi il nome della tabella, lo stato del lavoro e il numero di righe caricate. È possibile ottenere l'ID del lavoro nelle risposte alle funzioni di avvio del caricamento dei dati o dalla `rds_aurora.limitless_data_load_jobs` vista.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_job_details WHERE
 job_id='1725696114225';

job_id        | destination_table_name | destination_schema_name | start_time
      | status    | full_load_rows | full_load_total_rows | full_load_complete_time |
 cdc_insert | cdc_update | cdc_delete
--------------+------------------------+-------------------------
+-------------------------+-----------+----------------+---------------------
+-------------------------+------------+------------+------------
1725696114225 | standard_1             | public                  | 2024-09-07
 08:23:57+00 | COMPLETED | 100000         | 100000              | 2024-09-07
 08:24:08+00  | 0          | 0          | 0
```

```
1725696114225 | standard_2             | public                  | 2024-09-07
 08:24:08+00 | COMPLETED | 100000         | 100000               | 2024-09-07
 08:24:17+00  | 0          | 0          | 0
1725696114225 | standard_3             | public                  | 2024-09-07
 08:24:18+00 | COMPLETED | 1              | 1                    | 2024-09-07
 08:24:20+00  | 0          | 0          | 0
1725696114225 | standard_4             | public                  | 2024-09-07
 08:23:58+00 | PENDING   | 0              | 0                    |
    | 0          | 0          | 0
(4 rows)
```

I Job record vengono eliminati dopo 90 giorni.

Monitoraggio del gruppo di CloudWatch log Amazon

Dopo che lo stato del processo di caricamento dei dati cambia a`RUNNING`, puoi controllare l'avanzamento del runtime utilizzando Amazon CloudWatch Logs.

Per monitorare i flussi di CloudWatch log

Accedi a AWS Management Console e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.

1. Passa a Registri, quindi a Gruppi di log.
2. Scegli il gruppo di log/aws/rds/aurora-limitless-database.
3. Cerca il flusso di log del tuo processo di caricamento dei dati tramite job_id.

   Il flusso di log ha lo schema Data-Load-Job -. *job_id*
4. Scegliete il flusso di registro per vedere gli eventi del registro.

Ogni flusso di log mostra gli eventi contenenti lo stato del lavoro e il numero di righe caricate nelle tabelle di destinazione del database Aurora Postgre SQL Limitless. Se un processo di caricamento dei dati non riesce, viene creato anche un registro degli errori che mostra lo stato dell'errore e il motivo.

I Job record vengono eliminati dopo 90 giorni.

Monitoraggio RDS degli eventi

Il processo di caricamento dei dati pubblica anche RDS eventi, ad esempio quando un processo riesce, fallisce o viene annullato. È possibile visualizzare gli eventi dal database di destinazione.

Per ulteriori informazioni, consulta Eventi del gruppo DB Shard.

Annullamento del caricamento dei dati

Per annullare un processo di caricamento dei dati, chiamate la `rds_aurora.limitless_data_load_cancel` stored procedure, con l'ID del lavoro come parametro di input. Questa stored procedure viene richiamata dallo stesso database del gruppo di shard DB in cui è stato avviato lo specifico processo di caricamento dei dati. Per esempio:

```
CALL rds_aurora.limitless_data_load_cancel(12345);

INFO: limitless data load job with id 12345 is canceling without rollback.
```

Non è possibile annullare un processo di caricamento dati che non esiste o non è in esecuzione nello stesso gruppo di shard DB.

L'utilità di caricamento dati Aurora Postgre SQL Limitless Database lascia i dati caricati nelle tabelle di destinazione senza rollback, come mostra la risposta. Se non desideri conservare i dati caricati, ti consigliamo di troncare le tabelle di destinazione.

Interrogazione del database Aurora SQL Postgre Limitless

Aurora Postgre SQL Limitless Database è compatibile con la sintassi Postgre per le query. SQL Puoi interrogare il tuo Limitless Database utilizzando `psql` o qualsiasi altra utilità di connessione che funzioni con Postgre. SQL Per eseguire le query, ti connetti all'endpoint illimitato come mostrato in. Connessione al cluster Aurora PostgreSQL Limitless Database DB

Tutte le SQL `SELECT` query Postgre sono supportate in Aurora Postgre Limitless Database. SQL Tuttavia, le interrogazioni vengono eseguite su due livelli:

1. Router a cui il client invia la query
2. Shards in cui si trovano i dati

Le prestazioni dipendono dall'interrogazione del database in modo da consentirgli di ottenere un elevato grado di elaborazione simultanea di diverse query su frammenti diversi. Le query vengono prima analizzate nel livello di transazione distribuito (router). Prima di pianificare l'esecuzione della query, è prevista una fase di analisi per identificare la posizione di tutte le relazioni che partecipano alla query. Se tutte le relazioni sono tabelle condivise con una chiave shard filtrata sullo stesso shard o tabelle di riferimento, la pianificazione delle query viene ignorata sul livello del router e trasmessa completamente allo shard per la pianificazione e l'esecuzione. Questo processo riduce il numero di round trip tra diversi nodi (router e shard) e si traduce in prestazioni migliori nella maggior parte dei casi. Per ulteriori informazioni, consulta Query a shard singolo nel database Aurora PostgreSQL Limitless.

> **Note**
>
> In alcuni casi specifici, ad esempio in un prodotto cartesiano (cross join), la query offre prestazioni migliori recuperando i dati separatamente dallo shard.

Per ulteriori informazioni sui piani di esecuzione delle query, vedere in. EXPLAIN Riferimento al database Aurora SQL Postgre Limitless Per informazioni generali sulle query, consulta Interrogazioni nella documentazione di SQL Postgre.

Argomenti

- Query a shard singolo nel database Aurora PostgreSQL Limitless
- Interrogazioni distribuite nel database SQL Aurora Postgre Limitless

- Tracciamento delle query distribuito nei log di Postgre nel database Aurora SQL Postgre Limitless SQL
- Deadlock distribuiti nel database Aurora Postgre Limitless SQL

Query a shard singolo nel database Aurora PostgreSQL Limitless

Una query a shard singolo è una query che può essere eseguita direttamente su uno shard mantenendo la semantica di SQL ACID. Quando viene rilevata una query di questo tipo dal pianificatore di query sul router, il planner la rileva e procede a inviare l'intera query SQL allo shard corrispondente.

Questa ottimizzazione riduce il numero di round trip di rete dal router allo shard, migliorando le prestazioni. Attualmente questa ottimizzazione viene eseguita per`INSERT`, `SELECTUPDATE`, e le `DELETE` interrogazioni.

Argomenti

- Esempi di query a frammento singolo
- Restrizioni per le query a shard singolo
- Unioni completamente qualificate (esplicite)
- Impostazione di una chiave shard attiva

Esempi di query a frammento singolo

Negli esempi seguenti, abbiamo la tabella condivisa`customers`, con la chiave shard`customer_id`, e la tabella di riferimento. `zipcodes`

SELECT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM customers WHERE
 customer_id = 100;

                        QUERY PLAN
-----------------------------------------------------------
 Foreign Scan
   Output: customer_id, other_id, customer_name, balance
   Remote SQL:  SELECT customer_id,
     other_id,
     customer_name,
```

```
     balance
    FROM public.customers
   WHERE (customer_id = 100)
 Single Shard Optimized
(9 rows)
```

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM orders
    LEFT JOIN zipcodes ON orders.zipcode_id = zipcodes.zipcode_id
    WHERE customer_id = 11;

                                                QUERY PLAN

-------------------------------------------------------------------------------------------------
 Foreign Scan
   Output: customer_id, order_id, zipcode_id, customer_name, balance,
 zipcodes.zipcode_id, zipcodes.city
   Remote SQL:  SELECT orders.customer_id,
     orders.order_id,
     orders.zipcode_id,
     orders.customer_name,
     orders.balance,
     zipcodes.zipcode_id,
     zipcodes.city
    FROM (public.orders
      LEFT JOIN public.zipcodes ON ((orders.zipcode_id = zipcodes.zipcode_id)))
   WHERE (orders.customer_id = 11)
 Single Shard Optimized
(13 rows)
```

INSERT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) INSERT INTO customers
    (customer_id, other_id, customer_name, balance)
    VALUES (1, 10, 'saikiran', 1000);

                      QUERY PLAN
------------------------------------------------------
 Insert on public.customers
   ->  Result
         Output: 1, 10, 'saikiran'::text, '1000'::real
 Single Shard Optimized
(4 rows)
```

UPDATE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) UPDATE orders SET balance = balance +
 100
    WHERE customer_id = 100;

                                        QUERY PLAN

-------------------------------------------------------------------------------------------
 Update on public.orders
   Foreign Update on public.orders_fs00002 orders_1
   ->  Foreign Update
         Remote SQL:  UPDATE public.orders SET balance = (balance + (100)::double
 precision)
   WHERE (customer_id = 100)
 Single Shard Optimized
(6 rows)
```

DELETE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) DELETE FROM orders
    WHERE customer_id = 100 and balance = 0;

                             QUERY PLAN
---------------------------------------------------------------------
 Delete on public.orders
   Foreign Delete on public.orders_fs00002 orders_1
   ->  Foreign Delete
         Remote SQL:  DELETE FROM public.orders
   WHERE ((customer_id = 100) AND (balance = (0)::double precision))
 Single Shard Optimized
(6 rows)
```

Restrizioni per le query a shard singolo

Le query a shard singolo presentano le seguenti restrizioni:

Funzioni

Se una query a shard singolo contiene una funzione, la query è idonea per l'ottimizzazione a shard singolo solo se si applica una delle seguenti condizioni:

- La funzione è immutabile. Per ulteriori informazioni, consulta Volatilità della funzione.
- La funzione è mutabile, ma è registrata nella vista. `rds_aurora.limitless_distributed_functions` Per ulteriori informazioni, consulta Distribuzione delle funzioni.

Visualizzazioni

Se una query contiene una o più viste, l'ottimizzazione a shard singolo è disabilitata per la query se presenta una delle seguenti condizioni:

- Qualsiasi vista ha l'`security_barrier`attributo.
- Gli oggetti utilizzati nella query richiedono più privilegi utente. Ad esempio, una query contiene due visualizzazioni che vengono eseguite da due utenti diversi.

```
CREATE VIEW v1 AS SELECT customer_name FROM customers c WHERE c.customer_id =  1;
CREATE VIEW v2 WITH (security_barrier) AS SELECT customer_name FROM customers c
 WHERE c.customer_id =  1;

postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v1;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Output: customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 c  (cost=0.00..24.12 rows=6 width=32)
           Output: c.customer_name
           Filter: (c.customer_id = 1)
         Query Identifier: -6005737533846718506
   Remote SQL:  SELECT customer_name
    FROM ( SELECT c.customer_name
             FROM public.customers c
            WHERE (c.customer_id = 1)) v1
 Query Identifier: -5754424854414896228
(12 rows)


postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v2;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Foreign Scan on public.customers_fs00001 c  (cost=100.00..128.41 rows=7 width=32)
   Output: c.customer_name
   Remote Plans from Shard postgres_s3:
```

```
         Seq Scan on public.customers_ts00001 customers  (cost=0.00..24.12 rows=6
 width=32)
           Output: customers.customer_name
           Filter: (customers.customer_id = 1)
         Query Identifier: 4136563775490008117
   Remote SQL: SELECT customer_name FROM public.customers WHERE ((customer_id = 1))
 Query Identifier: 5056054318010163757
(9 rows)
```

Istruzioni PREPARE ed EXECUTE

Aurora PostgreSQL Limitless Database supporta l'ottimizzazione a shard singolo per le istruzioni preparate e. `SELECT` `UPDATE` `DELETE`

Tuttavia, se si utilizzano istruzioni preparate per `PREPARE` e `EXECUTE` con `plan_cache_mode` set to, il pianificatore di query rifiuta l'ottimizzazione a `'force_generic_plan'` shard singolo per quella query.

PL/pgSQL

Interrogazioni con PL/pgSQL variables are run as implicitly prepared statements. If a query contains any PL/pgSQL variabili, il pianificatore di query rifiuta l'ottimizzazione a shard singolo.

L'ottimizzazione è supportata nelle variabili. PL/pgSQL block if the statement doesn't contain any PL/pgSQL

Unioni completamente qualificate (esplicite)

L'ottimizzazione a shard singolo si basa sull'eliminazione delle partizioni. L'ottimizzatore PostgreSQL elimina le partizioni in base a condizioni costanti. Se Aurora PostgreSQL Limitless Database rileva che tutte le partizioni e le tabelle rimanenti si trovano sullo stesso shard, contrassegna la query come idonea per l'ottimizzazione a shard singolo. Tutte le condizioni di filtro devono essere esplicite affinché l'eliminazione delle partizioni funzioni. Aurora PostgreSQL Limitless Database non è in grado di eliminare le partizioni senza uno o più predicati di join o filtrare i predicati sulle chiavi shard di ogni tabella condivisa nell'istruzione.

Supponiamo di aver partizionato le tabelle, e in base alla colonna. `customers` `orders` `order_details` `customer_id` In questo schema, l'applicazione cerca di conservare tutti i dati di un cliente su un singolo shard.

Considera la query seguente:

```
SELECT * FROM
    customers c, orders o, order_details od
WHERE c.customer_id = o.customer_id
    AND od.order_id = o.order_id
    AND c.customer_id = 1;
```

Questa query recupera tutti i dati di un cliente ()`c.customer_id = 1`. I dati di questo cliente si trovano su un singolo shard, ma Aurora PostgreSQL Limitless Database non qualifica questa query come query a shard singolo. Il processo di ottimizzazione per la query è il seguente:

1. L'ottimizzatore può eliminare le partizioni per `customers` e in `orders` base alla seguente condizione:

   ```
   c.customer_id = 1
   c.customer_id = o.customer_id
   o.customer_id =  1 (transitive implicit condition)
   ```

2. L'ottimizzatore non può eliminare alcuna partizione per`order_details`, perché non esiste una condizione costante sulla tabella.
3. L'ottimizzatore conclude di aver letto tutte le partizioni da. `order_details` Pertanto, la query non può essere qualificata per l'ottimizzazione a shard singolo.

Per renderla una query a shard singolo, aggiungiamo la seguente condizione di unione esplicita:

```
o.customer_id = od.customer_id
```

La query modificata ha il seguente aspetto:

```
SELECT * FROM
    customers c, orders o,  order_details od
WHERE c.customer_id = o.customer_id
     AND o.customer_id = od.customer_id
     AND od. order_id = o. order_id
 AND c.customer_id =  1;
```

Ora l'ottimizzatore può eliminare le partizioni per. `order_details` La nuova query diventa una query a frammento singolo e si qualifica per l'ottimizzazione.

Impostazione di una chiave shard attiva

Questa funzionalità consente di impostare una singola chiave shard durante l'interrogazione del database, facendo sì che tutte `SELECT` le query DML vengano aggiunte con la chiave shard come predicato costante. Questa funzionalità è utile se hai effettuato la migrazione ad Aurora PostgreSQL Limitless Database e hai denormalizzato lo schema aggiungendo chiavi shard alle tabelle.

È possibile aggiungere automaticamente un predicato di chiave shard alla logica SQL esistente, senza modificare la semantica delle query. L'aggiunta di un predicato di chiave shard attiva viene eseguita solo per le tabelle compatibili.

La funzionalità active shard key utilizza la `rds_aurora.limitless_active_shard_key` variabile, che ha la seguente sintassi:

```
SET [session | local] rds_aurora.limitless_active_shard_key = '{"col1_value",
 "col2_value", ...}';
```

Alcune considerazioni sulle chiavi shard attive e sulle chiavi esterne:

- Una tabella condivisa può avere un vincolo di chiave esterna se le tabelle principale e secondaria sono collocate e la chiave esterna è un superset della chiave shard.
- Una tabella condivisa può avere un vincolo di chiave esterna rispetto a una tabella di riferimento.
- Una tabella di riferimento può avere un vincolo di chiave esterna rispetto a un'altra tabella di riferimento.

Supponiamo di avere una `customers` tabella condivisa sulla colonna. `customer_id`

```
BEGIN;
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE customers(customer_id int PRIMARY KEY, name text , email text);
COMMIT;
```

Con un set di chiavi shard attivo, le query hanno le seguenti trasformazioni.

SELECT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
```

```
SELECT * FROM customers;

-- This statement is changed to:
SELECT * FROM customers WHERE customer_id = '123'::int;
```

INSERT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
INSERT INTO customers(name, email) VALUES('Alex', 'alex@example.com');

-- This statement is changed to:
INSERT INTO customers(customer_id, name, email) VALUES('123'::int, 'Alex',
 'alex@example.com');
```

UPDATE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
UPDATE customers SET email = 'alex_new_email@example.com';

-- This statement is changed to:
UPDATE customers SET email = 'alex_new_email@example.com' WHERE customer_id =
 '123'::int;
```

DELETE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
DELETE FROM customers;

-- This statement is changed to:
DELETE FROM customers WHERE customer_id = '123'::int;
```

Join

Quando si eseguono operazioni di join su tabelle con una chiave shard attiva, il predicato della chiave shard viene aggiunto automaticamente a tutte le tabelle coinvolte nel join. Questa aggiunta automatica del predicato della chiave shard si verifica solo quando tutte le tabelle della query appartengono allo stesso gruppo di collocazione. Se la query riguarda tabelle di gruppi di collocazione diversi, viene invece generato un errore.

Supponiamo di avere `orders` anche delle `order_details` tabelle che sono collocate con la `customers` tabella.

```
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_collocate_with='customers';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE orders (id int , customer_id int, total_amount int, date date);
CREATE TABLE order_details (id int , order_id int, customer_id int, product_name
 VARCHAR(100), price int);
COMMIT;
```

Recupera le ultime 10 fatture d'ordine per un cliente il cui ID cliente è 10.

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    order_details.order_id = orders.order_id AND
    customers.customer_id = 10
    order by order_date limit 10;
```

Questa interrogazione viene trasformata nella seguente:

```
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    orders.order_id = order_details.order_id AND
    customers.customer_id = 10 AND
    order_details.customer_id = 10 AND
    orders.customer_id = 10 AND
    ORDER BY "order_date" LIMIT 10;
```

Tabelle compatibili con Active Shard Key

Il predicato della chiave shard viene aggiunto solo alle tabelle compatibili con la chiave shard attiva. Una tabella è considerata compatibile se ha lo stesso numero di colonne nella chiave shard specificato nella variabile. `rds_aurora.limitless_active_shard_key` Se la query riguarda tabelle incompatibili con la chiave shard attiva, il sistema genera un errore invece di procedere con la query.

Per esempio:

```
-- Compatible table
SET rds_aurora.limitless_active_shard_key = '{"10"}';

-- The following query works because the customers table is sharded on one column.
```

```
SELECT * FROM customers;

-- Incompatible table
SET rds_aurora.limitless_active_shard_key = '{"10","20"}';

-- The following query raises a error because the customers table isn't sharded on
 two columns.
 SELECT * FROM customers;
```

Interrogazioni distribuite nel database SQL Aurora Postgre Limitless

Le query distribuite vengono eseguite su un router e su più di uno shard. La richiesta viene ricevuta da uno dei router. Il router crea e gestisce la transazione distribuita, che viene inviata agli shard partecipanti. Gli shard creano una transazione locale con il contesto fornito dal router e la query viene eseguita.

Quando la transazione viene confermata, il router utilizza un protocollo di commit ottimizzato a due fasi, se necessario, e Multi Version Concurrency Control (MVCC) basato sul tempo per fornire la ACIDsemantica in un sistema di database distribuito.

Time-based MVCC registra l'ora di commit per ogni transazione e utilizza l'ora di inizio della transazione per generare l'ora dello snapshot dei dati. Per determinare se una transazione è stata confermata (visibile) in base all'istantanea di un lettore, il database confronta l'ora di commit con l'ora dell'istantanea. Se il tempo di commit è inferiore a quello dell'istantanea del lettore, è visibile, altrimenti invisibile. Con questo protocollo, ti aspetterai sempre di vedere dati fortemente coerenti su Aurora SQL Postgre Limitless Database.

Tracciamento delle query distribuito nei log di Postgre nel database Aurora SQL Postgre Limitless SQL

Il tracciamento distribuito delle query è uno strumento per tracciare e correlare le query nei log di Postgre SQL su Aurora Postgre Limitless Database. SQL In Aurora PostgreSQL, si utilizza l'ID della transazione per identificare una transazione. Ma in Aurora Postgre SQL Limitless Database, l'ID della transazione può essere ripetuto su vari router. Pertanto, ti consigliamo di utilizzare l'ID di tracciamento invece in Limitless Database.

I casi d'uso principali sono i seguenti:

- Utilizzate la `rds_aurora.limitless_get_last_trace_id()` funzione per trovare l'ID di tracciamento univoco dell'ultima query eseguita nella sessione corrente. Quindi cerca nel gruppo di log del cluster DB in Amazon CloudWatch Logs utilizzando quell'ID di tracciamento per trovare tutti i log correlati.

  Puoi utilizzare i `log_min_error_statement` parametri `log_min_messages` e insieme per controllare il volume dei log stampati e stampare una dichiarazione che contenga l'ID di tracciamento.
- Utilizzate il `log_min_duration_statement` parametro per determinare il tempo di esecuzione oltre il quale tutte le query stampano la durata di esecuzione e l'ID di tracciamento. Questa fase di esecuzione può quindi essere ricercata nel gruppo di log del cluster DB in CloudWatch Logs per determinare i nodi critici e le attività di ottimizzazione del pianificatore.

  Il `log_min_duration_statement` parametro abilita l'ID di tracciamento per tutti i nodi indipendentemente dai valori e dai parametri. `log_min_messages` `log_min_error_statement`

Argomenti

- ID di tracciamento
- Utilizzo del tracciamento delle query
- Esempi di log

ID di tracciamento

Al centro di questa funzionalità c'è un identificatore univoco noto come ID di tracciamento. L'ID di tracciamento è una stringa di 31 cifre aggiunta alle righe di registro dei STATEMENT log di Postgre, che funge da identificatore preciso per correlare SQL i log relativi all'esecuzione

di una specifica query. Alcuni esempi sono `1126253375719408504000000000011` e `112625337571940849500000000090`.

L'ID di tracciamento è composto da quanto segue:

- Identificatore della transazione: le prime 20 cifre, che identificano in modo univoco la transazione.
- Identificatore di comando: le prime 30 cifre, che indicano una singola query all'interno di una transazione.

  Se vengono eseguite più di 4.294.967.294 query all'interno di un blocco di transazione esplicito, il comando Identifier si riferisce a. 1 Quando ciò accade, vieni avvisato dal seguente messaggio nel registro di Postgre: `LOG` SQL

  ```
  wrapping around the tracing ID back to 1 after running 4294967294 (4.2 billion or
   2^32-2) queries inside of an explicit transaction block
  ```
- Identificatore del tipo di nodo: l'ultima cifra, che indica se il nodo funziona come router coordinatore (1) o nodo partecipante (). `0`

I seguenti esempi illustrano i componenti dell'ID di tracciamento:

- `1126253375719408504000000000011`:
  - Identificatore della transazione: `1126253375719408504`
  - Identificatore di comando: `1126253375719408504000000000001` indica il primo comando nel blocco di transazione
  - Identificatore del tipo di nodo: 1 indica un router coordinatore
- `112625337571940849500000000090`:
  - Identificatore della transazione — `1126253375719408495`
  - Identificatore di comando: `1126253375719408495000000000009` indica il nono comando nel blocco di transazione
  - Identificatore del tipo di nodo: `0` indica un nodo partecipante

Utilizzo del tracciamento delle query

Eseguite le seguenti attività per utilizzare il tracciamento delle query:

1. Assicurati che il tracciamento sia abilitato.

Puoi controllare usando il seguente comando:

```
SHOW rds_aurora.limitless_log_distributed_trace_id;
```

È abilitato di default (on). Se non è abilitato, impostalo usando il seguente comando:

```
SET rds_aurora.limitless_log_distributed_trace_id = on;
```

2. Controlla il volume dei log stampati configurando il livello di gravità del registro.

   Il volume del registro è controllato dal `log_min_messages` parametro. Il `log_min_error_statement` parametro viene utilizzato per stampare la `STATEMENT` riga con l'ID di tracciamento. Entrambi sono impostati come impostazione `ERROR` predefinita. È possibile effettuare il controllo utilizzando i seguenti comandi:

   ```
   SHOW log_min_messages;
   SHOW log_min_error_statement;
   ```

   Per aggiornare il livello di gravità e stampare la `STATEMENT` riga per la sessione corrente, utilizzate i seguenti comandi con uno di questi livelli di gravità:

```
SET log_min_messages = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO | NOTICE
 | WARNING | ERROR | LOG | FATAL | PANIC';
SET log_min_error_statement = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO
 | NOTICE | WARNING | ERROR | LOG | FATAL | PANIC';
```

   Per esempio:

   ```
   SET log_min_messages = 'WARNING';
   SET log_min_error_statement = 'WARNING';
   ```

3. Abilita la stampa dell'ID di tracciamento nei registri al di sopra di un periodo di esecuzione specifico.

   Il `log_min_duration_statement` parametro può essere modificato impostando il tempo minimo di esecuzione della query al di sopra del quale la query stampa una riga di registro con la durata dell'esecuzione, insieme alla traccia nel cluster IDs DB. Questo parametro è impostato

come impostazione `-1` predefinita, il che significa che è disabilitato. È possibile verificare utilizzando il seguente comando:

```
SHOW log_min_duration_statement;
```

Se lo si modifica, vengono `0` stampati la durata e l'ID di tracciamento nei log per ogni query nel cluster DB. È possibile impostarlo `0` per la sessione corrente utilizzando il seguente comando:

```
SET log_min_duration_statement = 0;
```

4. Ottieni l'ID di tracciamento.

   Dopo aver eseguito una query (anche all'interno di un blocco di transazione esplicito), chiama la `rds_aurora.limitless_get_last_trace_id` funzione per ottenere l'ID di tracciamento dell'ultima esecuzione della query:

   ```
   SELECT * FROM rds_aurora.limitless_get_last_trace_id();
   ```

   Questa funzione restituisce l'identificatore della transazione e l'identificatore del comando. Non restituisce l'identificatore del tipo di nodo.

   ```
   => SELECT * FROM customers;
    customer_id | fname | lname
   -------------+-------+-------
   (0 rows)

   => SELECT * FROM rds_aurora.limitless_get_last_trace_id();
    transaction_identifier |       command_identifier
   ------------------------+--------------------------------
    10104661421959001813   | 10104661421959001813000000001
   (1 row)
   ```

   La funzione restituisce una riga vuota per le query non distribuite, in quanto non dispongono di un ID di tracciamento.

   ```
   => SET search_path = public;
   SET

   => SELECT * FROM rds_aurora.limitless_get_last_trace_id();
    transaction_identifier | command_identifier
   ```

```
------------------------+--------------------
                        |
(1 row)
```

> **Note**
>
> Per le ANALYZE query VACUUM e, l'istruzione di durata non viene registrata con l'ID di tracciamento. Pertanto, `limitless_get_last_trace_id()` non restituisce l'ID di tracciamento. Se VACUUM o ANALYZE è un'operazione di lunga durata, è possibile utilizzare la seguente query per ottenere l'ID di traccia per tale operazione:
>
> ```
> SELECT * FROM rds_aurora.limitless_stat_activity
> WHERE distributed_tracing_id IS NOT NULL;
> ```
>
> Se il server si arresta prima che tu riesca a trovare l'ultimo ID di tracciamento, dovrai cercare manualmente nei SQL log di Postgre per trovare la traccia contenuta IDs nei log immediatamente prima dell'errore.

5. Cerca l'ID di tracciamento nei log del cluster DB utilizzando. CloudWatch

   Utilizza CloudWatch Insights per interrogare il gruppo di log del cluster DB, come illustrato negli esempi seguenti.

   - Esegui una query per un particolare identificatore di transazione e tutti i comandi vengono eseguiti al suo interno:

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813/
     | sort @timestamp desc
     ```

   - Interrogazione per un particolare identificatore di comando:

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813000000001/
     | sort @timestamp desc
     ```

6. Esamina tutti i log del cluster DB prodotti dalla query distribuita.

Esempi di log

Gli esempi seguenti mostrano l'uso del tracciamento delle query.

Correlazione dei log per le interrogazioni soggette a errori

In questo esempio il `TRUNCATE` comando viene eseguito sulla `customers` tabella quando tale tabella non esiste.

Senza tracciamento delle query

File di SQL registro Postgre sul router del coordinatore:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:CONTEXT: remote
 SQL command: truncate public.customers;
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:STATEMENT:
 truncate customers;
```

File di SQL registro Postgre su un partecipante condiviso:

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:STATEMENT:
 truncate customers;
```

Questi log sono tipici. Mancano degli identificatori precisi necessari per correlare facilmente le query all'interno del cluster DB.

Con tracciamento delle query

File di SQL registro Postgre sul router del coordinatore:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:CONTEXT: remote SQL
 command: truncate public.customers;
```

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT: /* tid =
 11262533757194085027000000000011 */ truncate customers;
```

File di SQL registro Postgre su un partecipante condiviso:

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR:  failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT:  /* tid
 = 11262533757194085027000000000010 */ truncate customers;
```

In presenza del tracciamento delle interrogazioni, a ogni riga di registro viene aggiunto un identificatore univoco di 31 cifre. Qui, `1126253375719408502700000000011` e `1126253375719408502700000000010` rappresentano la traccia IDs per i nodi coordinatore e partecipante, rispettivamente.

- Identificatore della transazione: `11262533757194085027`
- Identificatore del comando: — `1126253375719408502700000000001`
- Identificatore del tipo di nodo: l'ultima cifra, `1` o`0`, indica rispettivamente un router coordinatore e un nodo partecipante.

Registri correlati per trovare il tempo di esecuzione della query su vari nodi

In questo esempio, il `log_min_duration_statement` parametro è stato aggiornato `0` per stampare la durata di tutte le query.

Senza tracciamento delle query

```
2024-01-15 07:28:46 UTC:[local]:postgres@postgres_limitless:[178322]:LOG: duration:
 12.779 ms statement: select * from customers;
```

Con tracciamento delle interrogazioni

File di SQL registro Postgre sul router del coordinatore:

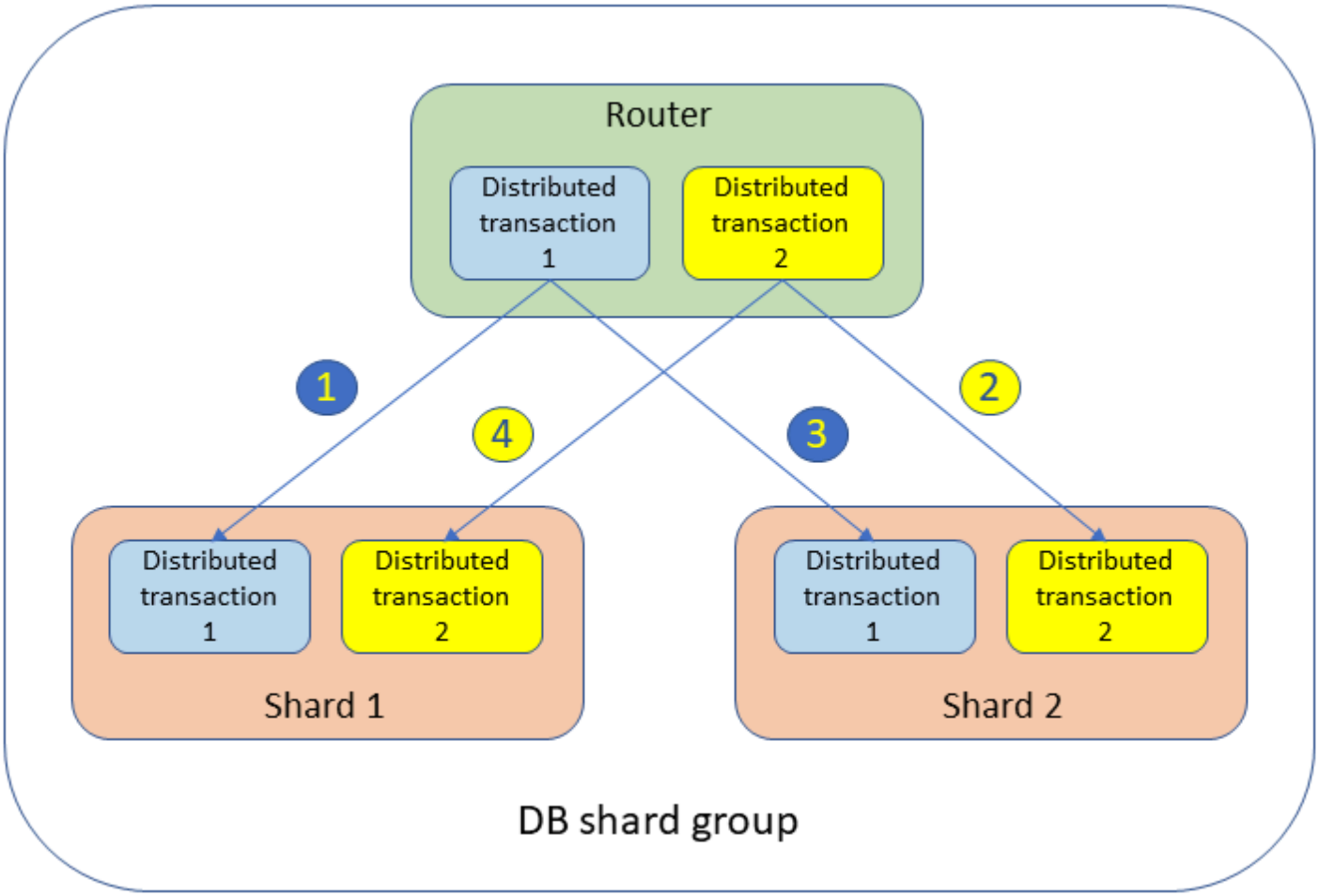
```
2024-01-15 07:32:08 UTC:[local]:postgres@postgres_limitless:[183877]:LOG: duration:
 12.618 ms statement: /* tid = 045766956624049708840000000011 */ select * from
 customers;
```

File di SQL registro Postgre su un partecipante condiviso:

```
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.279 ms statement: /* tid = 045766956624049708840000000010 */ START
 TRANSACTION ISOLATION LEVEL READ COMMITTED
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.249 ms parse <unnamed>: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.398 ms bind <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.019 ms execute <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.073 ms statement: /* tid = 045766956624049708840000000010 */ COMMIT
 TRANSACTION
```

Deadlock distribuiti nel database Aurora Postgre Limitless SQL

In un gruppo di shard DB, possono verificarsi dei deadlock tra transazioni distribuite tra router e shard diversi. Ad esempio, vengono eseguite due transazioni distribuite simultanee che si estendono su due shard, come illustrato nella figura seguente.



Le transazioni bloccano le tabelle e creano eventi di attesa nei due shard come segue:

1. Transazione distribuita 1:

```
UPDATE table SET value = 1 WHERE key = 'shard1_key';
```

   Questo contiene un blocco sullo shard 1.

2. Transazione distribuita 2:

```
UPDATE table SET value = 2 WHERE key = 'shard2_key';
```

Questo contiene un blocco sullo shard 2.

3. Transazione distribuita 1:

```
UPDATE table SET value = 3 WHERE key = 'shard2_key';
```

La transazione distribuita 1 è in attesa sullo shard 2.

4. Transazione distribuita 2:

```
UPDATE table SET value = 4 WHERE key = 'shard1_key';
```

La transazione distribuita 2 è in attesa sullo shard 1.

In questo scenario, né lo shard 1 né lo shard 2 vedono il problema: la transazione 1 è in attesa della transazione 2 sullo shard 2 e la transazione 2 è in attesa della transazione 1 sullo shard 1. Da un punto di vista globale, la transazione 1 è in attesa della transazione 2 e la transazione 2 è in attesa della transazione 1. Questa situazione in cui due transazioni su due frammenti diversi sono in attesa l'una dell'altra viene chiamata deadlock distribuito.

Aurora Postgre SQL Limitless Database è in grado di rilevare e risolvere automaticamente i deadlock distribuiti. Un router del gruppo di shard DB riceve una notifica quando una transazione attende troppo a lungo per acquisire una risorsa. Il router che riceve la notifica inizia a raccogliere le informazioni necessarie da tutti i router e gli shard all'interno del gruppo di shard DB. Il router procede quindi a terminare le transazioni che partecipano a un deadlock distribuito, finché le altre transazioni nel gruppo di shard DB non possono procedere senza essere bloccate l'una dall'altra.

Viene visualizzato il seguente errore quando la transazione faceva parte di un deadlock distribuito ed è stata quindi interrotta dal router:

```
ERROR: aborting transaction participating in a distributed deadlock
```

Il parametro del cluster `rds_aurora.limitless_distributed_deadlock_timeout` DB imposta il tempo di attesa di ogni transazione su una risorsa prima di notificare al router la verifica di un deadlock distribuito. È possibile aumentare il valore del parametro se il carico di lavoro è meno soggetto a situazioni di deadlock. L'impostazione predefinita è `1000` millisecondi (1 secondo).

Il ciclo di deadlock distribuito viene pubblicato nei SQL log di Postgre quando viene trovato e risolto un deadlock tra nodi. Le informazioni su ogni processo che fa parte del deadlock includono quanto segue:

- Nodo coordinatore che ha avviato la transazione
- ID della transazione virtuale (xid) della transazione sul nodo coordinatore, nel formato *backend_id*/*backend_local_xid*
- ID di sessione distribuito della transazione

Gestione del database Aurora SQL Postgre Limitless

I seguenti argomenti descrivono come gestire i cluster Aurora Postgre SQL Limitless Database DB.

Argomenti

- Considerazioni sulle dimensioni di database e tabelle
- Recupero dello spazio di archiviazione mediante aspirazione

Considerazioni sulle dimensioni di database e tabelle

Per Aurora Postgre SQL Limitless Database, su ogni shard una tabella condivisa è suddivisa in un numero di slice di tabella che varia a seconda del numero di shard disponibili nel gruppo di shard DB. Ogni slice di tabella può crescere fino a 32 TiB, ma ogni shard ha una capacità massima di 128 TiB. Le tabelle di riferimento hanno un limite di dimensione di 32 TiB per l'intero gruppo di shard DB.

> **Note**
>
> La capacità massima di ogni nodo (router o shard) è di 128 TiB, poiché è la capacità massima per un cluster SQL Aurora Postgre DB.

Il numero massimo di relazioni per database (incluse tabelle, viste e indici) in Aurora Postgre e Aurora Postgre Limitless Database è SQL 1.431.650.303. SQL

Per ulteriori informazioni, consulta l'Appendice K. Limiti di Postgre nella documentazione di Postgre e. SQL SQL Limiti di dimensione Amazon Aurora

Recupero dello spazio di archiviazione mediante aspirazione

Postgre SQL Multiversion Concurrency Control (MVCC) aiuta a preservare l'integrità dei dati salvando una copia interna delle righe aggiornate o eliminate fino al completamento o al rollback di una transazione. Queste copie, chiamate anche tuple, possono causare il sovraccarico della tabella se non vengono pulite regolarmente. SQLLe istanze Postgre ordinano le transazioni in base alla transazione IDs e Postgre SQL utilizza la tecnologia basata sull'ID delle transazioni per controllare la visibilità delle tuple e garantire l'isolamento MVCC delle transazioni. Ogni transazione stabilisce un'istantanea dei dati e ogni tupla ha una versione. Sia l'istantanea che la versione sono basate sull'ID della transazione.

Per ripulire i dati, l'`VACUUM`utilità svolge quattro funzioni chiave in Postgre: SQL

- `VACUUM`— Rimuove le versioni di riga scadute, rendendo lo spazio disponibile per il riutilizzo.
- `VACUUM FULL`— Fornisce una deframmentazione completa rimuovendo le versioni con riga morta e compattando le tabelle, riducendo le dimensioni e aumentando l'efficienza.
- `VACUUM FREEZE`— Protegge dai problemi di configurazione degli ID delle transazioni contrassegnando le versioni di riga precedenti come bloccate.
- `VACUUM ANALYZE`— Rimuove le versioni dead row e aggiorna le statistiche di pianificazione delle interrogazioni del database. È una combinazione delle `ANALYZE` funzioni `VACUUM` e. Per ulteriori informazioni su come `ANALYZE` funziona Aurora Postgre SQL Limitless Database, vedere. ANALYZE

Allo stesso modoMVCC, l'aspirapolvere in Aurora SQL Postgre è basato sull'ID della transazione. Se c'è una transazione in corso quando inizia l'operazione di vacuuming, le righe che sono ancora visibili in quella transazione non vengono rimosse.

Per ulteriori informazioni sull'`VACUUM`utilità, consulta VACUUMla documentazione di SQL Postgre. Per ulteriori informazioni sul `VACUUM` supporto in Aurora Postgre SQL Limitless Database, vedere. VACUUM

Argomenti

- AUTOVACUUM
- Aspirazione basata sul tempo nel database Aurora Postgre Limitless SQL
- Utilizzo delle statistiche del database per l'aspirazione

- Differenze nel comportamento di aspirazione tra Aurora Postgre e Aurora Postgre Limitless Database SQL SQL

AUTOVACUUM

Aurora Postgre SQL utilizza le AUTOVACUUM utilità VACUUM and per rimuovere le tuple non necessarie. Il meccanismo alla base di un AUTOVACUUM manuale VACUUM è lo stesso; l'unica differenza è l'automazione.

AUTOVACUUMin Aurora Postgre e Aurora SQL SQL Postgre Limitless Database è una combinazione delle utilità e. VACUUM ANALYZE AUTOVACUUMdetermina quali database e tabelle pulire, in base a una regola predefinita, come la percentuale di tuple morte e il numero di inserti.

Ad esempio, AUTOVACUUM «si riattiva» periodicamente per eseguire la pulizia. L'intervallo è controllato dal parametro. autovacuum_naptime Il valore predefinito è 1 minuto. I valori predefiniti AUTOVACUUM e i parametri di VACUUM configurazione sono gli stessi per Aurora Postgre SQL Limitless Database e per Aurora Postgre. SQL

Il AUTOVACUUM demone, se abilitato, emette automaticamente i ANALYZE comandi ogni volta che il contenuto di una tabella è sufficientemente modificato. In Aurora Postgre SQL Limitless Database, AUTOVACUUM problemi sia sui router che ANALYZE sugli shard.

Per ulteriori informazioni sui parametri di archiviazione dei AUTOVACUUM daemon e delle tabelle associati aAUTOVACUUM, consulta Il demone autovacuum e i parametri di archiviazione nella documentazione di Postgre. SQL

Aspirazione basata sul tempo nel database Aurora Postgre Limitless SQL

Aurora Postgre SQL Limitless Database è un sistema distribuito, il che significa che più istanze possono essere coinvolte in una transazione. Pertanto, la visibilità basata sull'ID della transazione non si applica. Al contrario, Aurora Postgre SQL Limitless Database utilizza la visibilità basata sul tempo, poiché le transazioni IDs non sono «unificate» tra le istanze, ma il tempo può essere «unificato» tra le istanze. Ogni snapshot della transazione e ogni versione della tupla obbediscono all'ora anziché all'ID della transazione. Per essere più specifici, un'istantanea della transazione ha un'ora di inizio dell'istantanea e una tupla ha un'ora di creazione (quando si verifica un INSERT o) e un'ora di cancellazione (quando UPDATE accade un). DELETE

Per mantenere la coerenza dei dati tra le istanze del gruppo di shard DB, Aurora Postgre SQL Limitless Database deve assicurarsi che l'vacuuming non rimuova le tuple ancora visibili a qualsiasi

transazione attiva nel gruppo di shard DB. Pertanto, anche l'aspirazione in Aurora Postgre SQL Limitless Database è basata sul tempo. Altri aspetti `VACUUM` rimangono invariati, incluso il fatto che per essere eseguito `VACUUM` su una particolare tabella, un utente deve avere accesso a quella tabella.

> **Note**
>
> Ti consigliamo vivamente di non lasciare le transazioni aperte per lunghi periodi di tempo. L'aspirapolvere basato sul tempo consuma più memoria rispetto all'aspirapolvere basato sull'ID delle transazioni.

L'esempio seguente illustra come funziona l'aspirazione basata sul tempo.

1. Una tabella dei clienti è distribuita su quattro frammenti.
2. La transazione 1 inizia con una lettura ripetibile e si rivolge a un solo shard (shard 1). Questa transazione rimane aperta.

   La transazione 1 è più vecchia di qualsiasi altra transazione iniziata dopo di essa.
3. La transazione 2 inizia più tardi ed elimina tutte le tuple da una tabella, quindi esegue il commit.
4. Se il `AUTOVACUUM` nostro manuale `VACUUM` tenta di ripulire le tuple morte (morte a causa della transazione 2), non rimuove nulla.

   Questo vale non solo per gli shard 1, ma anche per gli shard 2—4, perché la transazione 1 potrebbe ancora aver bisogno di accedere a queste tuple. Sono ancora visibili nella transazione 1 a causa di. MVCC

L'ultimo passaggio viene eseguito tramite la sincronizzazione, in modo che tutti gli shard siano a conoscenza della transazione 1, anche se la transazione 1 non li riguarda tutti.

Utilizzo delle statistiche del database per l'aspirazione

Per ottenere informazioni sulle tuple che potresti dover ripulire, usa la vista limitless_stat_all_tables, che funziona in modo simile a pg_stat_all_tables. L'esempio seguente interroga la vista.

```
SELECT * FROM rds_aurora.limitless_stat_all_tables WHERE relname LIKE '%customer%';
```

Allo stesso modo, per le statistiche del database usa limitless_stat_database invece di pg_stat_database e limitless_stat_activity invece di pg_stat_activity.

Per controllare l'utilizzo del disco della tabella, usa la funzione limitless_stat_relation_sizes, che funziona in modo simile a pg_relation_size. L'esempio seguente interroga la funzione.

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customer');
```

Per tenere traccia dell'avanzamento di un'VACUUMoperazione su Aurora Postgre SQL Limitless Database, usa la vista limitless_stat_progress_vacuum invece di pg_stat_progress_vacuum.
L'esempio seguente interroga la vista.

```
SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;
```

Per ulteriori informazioni, consulta Visualizzazioni del database Aurora PostgreSQL Limitless e Funzioni del database Aurora SQL Postgre Limitless.

Differenze nel comportamento di aspirazione tra Aurora Postgre e Aurora Postgre Limitless Database SQL SQL

Alcune altre differenze tra Aurora Postgre e Aurora SQL SQL Postgre Limitless Database nel funzionamento dell'aspirazione sono le seguenti:

- Aurora Postgre SQL esegue `VACUUM` operazioni sulla transazione IDs fino alla transazione più vecchia in corso. Se non ci sono transazioni in corso nel database, `VACUUM` esegue l'operazione fino all'ultima transazione.
- Aurora Postgre SQL Limitless Database sincronizza l'istantanea temporale più vecchia ogni 10 secondi. Pertanto, `VACUUM` potrebbe non eseguire l'operazione su nessuna transazione eseguita negli ultimi 10 secondi.

Per informazioni sul supporto per `VACUUM` Aurora Postgre SQL Limitless Database, vedere in. VACUUM Riferimento al database Aurora SQL Postgre Limitless

Monitoraggio del database Aurora PostgreSQL Limitless

Puoi utilizzare Amazon CloudWatch, Enhanced Monitoring e Performance Insights per monitorare Aurora PostgreSQL Limitless Database. Sono inoltre disponibili nuove funzioni e visualizzazioni statistiche ed eventi di attesa per Aurora PostgreSQL Limitless Database che è possibile utilizzare per il monitoraggio e la diagnostica.

Argomenti

- Monitoraggio del database Aurora PostgreSQL Limitless con Amazon CloudWatch
- Monitoraggio del database Aurora PostgreSQL Limitless con Database Insights CloudWatch
- Monitoraggio del database Aurora Postgre SQL Limitless con Amazon Logs CloudWatch
- Monitoraggio del database Aurora Postgre SQL Limitless con monitoraggio avanzato
- Monitoraggio del database Aurora PostgreSQL Limitless con Performance Insights
- Monitoraggio del database Aurora Postgre SQL Limitless con Amazon Protection GuardDuty RDS
- Funzioni e visualizzazioni per Aurora SQL Postgre Limitless Database
- Attendi gli eventi per Aurora SQL Postgre Limitless Database

Monitoraggio del database Aurora PostgreSQL Limitless con Amazon CloudWatch

CloudWatch le metriche per Aurora PostgreSQL Limitless Database sono riportate nelle seguenti dimensioni:

- DBShardGroup (Gruppo)
- DBShardGroupRouterAggregation
- DBShardGroupInstance
- DBClusterIdentifier (Identificatore)

Per ulteriori informazioni sulle metriche, consulta. CloudWatch Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch

DBShardMetriche di gruppo

Per visualizzare le **`DBShardGroup`** metriche per Aurora PostgreSQL Limitless Database CloudWatch nella console, scegli RDS, quindi scegli Gruppo. DBShard

Puoi tenere traccia delle seguenti metriche: CloudWatch

- `DBShardGroupACUUtilization`— Utilizzo dell'unità di capacità Aurora (ACU) come percentuale calcolata dividendo `DBShardGroupCapacity` per. `DBShardGroupMaxACU`
- `DBShardGroupCapacity`— Numero di ACUs istanze utilizzate dal writer del gruppo di shard DB.
- `DBShardGroupComputeRedundancyCapacity`— Numero di istanze ACUs consumate dalle istanze in standby del gruppo di shard DB.
- `DBShardGroupMaxACU`— Numero massimo di ACUs configurazioni per il gruppo di shard DB.
- `DBShardGroupMinACU`— Numero minimo di file ACUs richiesti dal gruppo di shard DB.

La chiave `DBShardGroupIdentifier` della dimensione è disponibile per l'aggregazione delle `DBShardGroup` metriche.

DBShardGroupRouterAggregation metriche

Per visualizzare le **`DBShardGroupRouterAggregation`** metriche per Aurora PostgreSQL Limitless Database CloudWatch nella console, scegli RDS, quindi scegli. DBShardGroupRouterAggregation

Puoi tenere traccia delle seguenti metriche: CloudWatch

- `CommitThroughput`— Il numero medio di operazioni di commit al secondo su tutti i nodi del router del gruppo di shard DB.
- `DatabaseConnections`— La somma di tutte le connessioni su tutti i nodi del router nel gruppo di shard DB.

DBShardGroupInstance metriche

A DBShardGroupInstanceè la singola istanza DB all'interno di ogni sottogruppo di shard o router.

Per visualizzare le **`DBShardGroupInstance`** metriche per Aurora PostgreSQL Limitless Database CloudWatch nella console, scegli RDS, quindi scegli. DBShardGroupInstance

Puoi tenere traccia delle seguenti metriche: CloudWatch

- `ACUUtilization`— La percentuale calcolata come `ServerlessDatabaseCapacity` metrica divisa per il valore ACU massimo assegnato al sottocluster.
- `AuroraReplicaLag`- Per i cluster Limitless con ridondanza di calcolo, si tratta del ritardo nella replica degli aggiornamenti dall'istanza principale del sottocluster.
- `AuroraReplicaLagMaximum`— Per i cluster Limitless con ridondanza di calcolo, questo è il ritardo massimo durante la replica degli aggiornamenti dall'istanza principale nel sottocluster. Quando le repliche di lettura vengono eliminate o rinominate, potrebbe verificarsi un picco temporaneo del ritardo di replica dovuto al riciclo della vecchia risorsa. Utilizzate questa metrica per scoprire se si è verificato un failover a causa di un elevato ritardo di replica su uno dei suoi lettori.
- `AuroraReplicaLagMinimum`— Per i cluster Limitless con ridondanza di calcolo, questo è il ritardo minimo durante la replica degli aggiornamenti dall'istanza principale del sottocluster.
- `BufferCacheHitRatio`— La percentuale di dati e indici serviti dalla cache di memoria di un'istanza (rispetto al volume di archiviazione).
- `CommitLatency`— La durata media del completamento delle operazioni di commit da parte del motore e dello storage per un particolare nodo (router o shard).
- `CommitThroughput`— Il numero medio di operazioni di commit al secondo.
- `CPUUtilization`— Utilizzo della CPU come percentuale del valore ACU massimo assegnato al subcluster.
- `FreeableMemory`— La quantità di memoria inutilizzata disponibile quando il gruppo di shard viene scalato alla sua capacità massima. Ciò è determinato dall'assegnazione del gruppo ACUs

di shard. Per ogni unità ACU la capacità attuale è inferiore alla capacità massima, questo valore aumenta di circa 2 GiB. Pertanto, questa metrica non si avvicina allo zero finché il gruppo di shard DB non viene scalato fino al limite massimo.

- `MaximumUsedTransactionIDs`— L'età dell'ID di transazione non sottovalutato più vecchio, nelle transazioni. Se questo valore raggiunge 2.146.483.648 (2^31 - 1.000.000), il database viene forzato in modalità di sola lettura, per evitare il wraparound dell'ID transazione. Per ulteriori informazioni, consulta Impedire gli errori wraparound dell'ID transazione nella documentazione di PostgreSQL.
- `NetworkReceiveThroughput`— La quantità di throughput di rete ricevuta dai client da ciascuna istanza del gruppo di shard DB. Questo throughput non include il traffico di rete tra le istanze del gruppo di shard DB e il volume del cluster.
- `NetworkThroughput`— Il throughput di rete aggregato (sia trasmesso che ricevuto) tra client e router e router e shard nel gruppo di shard DB. Questo throughput non include il traffico di rete tra le istanze del gruppo di shard DB e il volume del cluster.
- `NetworkTransmitThroughput`— La quantità di throughput di rete inviata ai client da ciascuna istanza del gruppo di shard DB. Questo throughput non include il traffico di rete tra le istanze del gruppo di shard DB e il volume del cluster.
- `ReadIOPS`— Il numero medio di operazioni di input/output in lettura su disco al secondo (IOPS).
- `ReadLatency`— Il tempo medio impiegato per operazione (lettura) del disco. input/output (I/O
- `ReadThroughput`— Il numero medio di byte letti dal disco al secondo.
- `ServerlessDatabaseCapacity`— La capacità attuale dello shard DB o del sottocluster del router all'interno del gruppo di shard DB.
- `StorageNetworkReceiveThroughput`— La quantità di throughput di rete ricevuta dal sottosistema di archiviazione Aurora da ciascuna istanza del gruppo di shard DB.
- `StorageNetworkThroughput`— Il throughput di rete aggregato trasmesso e ricevuto dal sottosistema di archiviazione Aurora da ciascuna istanza del gruppo di shard DB.
- `StorageNetworkTransmitThroughput`— La quantità di throughput di rete inviata al sottosistema di archiviazione Aurora da ciascuna istanza del gruppo di shard DB.
- `SwapUsage`— La quantità di spazio di swap utilizzata dal gruppo di shard DB.
- `TempStorageIOPS`— Il numero medio di operazioni di I/O eseguite sullo storage locale collegato all'istanza DB. Include operazioni di I/O in lettura e scrittura.

`TempStorageIOPS`può essere utilizzato `TempStorageThroughput` per diagnosticare i rari casi in cui l'attività di rete per i trasferimenti tra le istanze DB e i dispositivi di archiviazione locali è responsabile di aumenti imprevisti della capacità.

- `TempStorageThroughput`— La quantità di dati trasferiti da e verso l'archiviazione locale associata a un router o a uno shard.
- `WriteIOPS`— Il numero medio di IOPS di scrittura su disco.
- `WriteLatency`— Il tempo medio impiegato per operazione di I/O di scrittura su disco.
- `WriteThroughput`— Il numero medio di byte scritti su disco al secondo.

Le seguenti chiavi dimensionali sono disponibili per l'aggregazione delle `DBShardGroupInstance` metriche:

- `DBClusterIdentifier`— Il cluster Aurora PostgreSQL DB.
- `DBShardGroupIdentifier`— Il gruppo di shard DB a cui appartiene l'istanza.
- `DBShardGroupSubClusterType`— Il tipo di nodo, `Distributed Transaction Router` (router) o `Data Access Shard` (shard).
- `DBShardGroupSubClusterIdentifier`— Il nome del router o dello shard a cui appartiene l'istanza.

Di seguito sono riportati alcuni esempi di metriche aggregate CloudWatch :

- Totale `CPUUtilization` di tutte le istanze che appartengono a un particolare shard o router in un gruppo di shard DB.
- Totale `CPUUtilization` di tutte le istanze in un gruppo di shard DB.

DBClusterMetriche degli identificatori

Per visualizzare le **`DBClusterIdentifier`** metriche per Aurora PostgreSQL Limitless Database CloudWatch nella console, scegli RDS, quindi scegli Identifier. DBCluster

Quando usi Aurora PostgreSQL Limitless Database, potresti avere più (più) input/output (I/O operazioni di quante ne avresti per un cluster Aurora DB. Puoi tenere traccia delle seguenti CloudWatch metriche per il tuo cluster Limitless Database:

- `VolumeReadIops`— Il numero di operazioni di I/O di lettura fatturate da un volume di cluster, riportate a intervalli di 5 minuti.
- `VolumeWriteIops`— Il numero di operazioni di I/O del disco di scrittura sul volume del cluster, riportate a intervalli di 5 minuti.

Aurora PostgreSQL Limitless Database utilizza la configurazione dello storage del cluster. Aurora I/O-Optimized ConAurora I/O-Optimized, paghi un unico prezzo mensile per tutte le operazioni di I/O, anziché pagare per ogni milione di richieste di I/O. Per ulteriori informazioni, consulta Configurazioni dell'archiviazione per i cluster database Amazon Aurora.

Potresti anche utilizzare più spazio di archiviazione rispetto a quello che utilizzeresti per un cluster Aurora DB. È possibile tenere traccia delle seguenti CloudWatch metriche per lo storage:

- `BackupRetentionPeriodStorageUsed`— L'utilizzo totale fatturato dello storage di backup continuo fatturato del cluster Aurora PostgreSQL Limitless Database.
- `SnapshotStorageUsed`— L'utilizzo totale fatturato dello storage di snapshot del cluster Aurora PostgreSQL Limitless Database.
- `TotalBackupStorageBilled`— La somma dei costi per la conservazione automatica dei backup e le istantanee del cluster DB.

  Per ulteriori informazioni sui costi dello storage di backup, vedereInformazioni sull'utilizzo dello storage di backup Amazon Aurora.
- `VolumeBytesUsed`— La quantità di storage utilizzata dal cluster Aurora PostgreSQL Limitless Database, riportata a intervalli di 5 minuti.

Monitoraggio del database Aurora PostgreSQL Limitless con Database Insights CloudWatch

La modalità Standard di Database Insights è richiesta come parte dell'abilitazione di Aurora PostgreSQL Limitless Database. È possibile utilizzarla per monitorare il carico del database (DB Load) delle istanze DB di Limitless Database in tempo reale. DB Load misura il livello di attività della sessione in un database. È possibile utilizzare Database Insights per analizzare e risolvere i problemi delle prestazioni delle istanze DB di Aurora PostgreSQL Limitless Database su larga scala.

Per ulteriori informazioni su Database Insights, consulta quanto segue. CloudWatch

- Monitoraggio dei database Amazon Aurora con Database Insights CloudWatch
- CloudWatch Database Insights nella Guida per CloudWatch l'utente di Amazon
- Inizia a usare CloudWatch Database Insights nella Amazon CloudWatch User Guide
-

Per informazioni sull'attivazione della modalità Avanzata o della modalità Standard di Database Insights, consulta i seguenti argomenti.

Argomenti

- Attivazione della modalità avanzata di Database Insights per Aurora PostgreSQL Limitless Database
- Attivazione della modalità Standard di Database Insights per Aurora PostgreSQL Limitless Database

Attivazione della modalità avanzata di Database Insights per Aurora PostgreSQL Limitless Database

Per attivare la modalità avanzata di Database Insights per il database Aurora PostgreSQL Limitless, utilizzare le seguenti procedure.

Attivazione della modalità avanzata di Database Insights durante la creazione di un cluster DB per Aurora PostgreSQL Limitless Database

Attiva la modalità avanzata di Database Insights quando crei un database per Aurora PostgreSQL Limitless Database.

Console

Nella console, puoi attivare la modalità avanzata di Database Insights quando crei un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità avanzata di Database Insights durante la creazione di un cluster DB utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona la modalità Avanzata. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15-24 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegliere Crea database.

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la creazione di un cluster DB, chiamate il create-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--db-cluster-identifier`— L'identificatore per il cluster DB.
- `--database-insights-mode advanced`per attivare la modalità avanzata di Database Insights.
- `--engine`— Il cluster DB deve utilizzare il motore `aurora-postgresql` DB.
- `--engine-version`— Il cluster DB deve utilizzare una delle versioni del motore DB:
  - `16.4-limitless`
  - `16.6-limitless`

- `--storage-type`— Il cluster DB deve utilizzare la configurazione di archiviazione del cluster `aurora-iopt1` DB.
- `--cluster-scalability-type`— specifica la modalità di scalabilità del cluster Aurora DB. Se impostato su`sulimitless`, il cluster funziona come un database Aurora PostgreSQL Limitless. Se impostato su `standard` (impostazione predefinita), il cluster utilizza la normale creazione di istanze DB.

  > **Note**
  >
  > Non è possibile modificare questa impostazione dopo aver creato il cluster DB.

- `--master-username`— Il nome dell'utente principale per il cluster DB.
- `--master-user-password`— La password per l'utente principale.
- `--enable-performance-insights`per attivare Performance Insights for Database Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione dei dati per il cluster DB. Per attivare Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.
- `--monitoring-interval`— L'intervallo, in secondi, tra i punti in cui vengono raccolte le metriche di Enhanced Monitoring per il cluster DB. Questo valore non può essere. `0`
- `--monitoring-role-arn`— Amazon Resource Name (ARN) per il ruolo IAM che consente a RDS di inviare metriche di Enhanced Monitoring ad Amazon Logs. CloudWatch
- `--enable-cloudwatch-logs-exports`— È necessario esportare i log in Logs. `postgresql` CloudWatch

L'esempio seguente abilita la modalità avanzata di Database Insights durante la creazione di un cluster DB.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode advanced \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
```

```
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 465 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Per Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode advanced ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 465 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Per attivare la modalità avanzata di Database Insights quando crei un', specifica i seguenti parametri per l'operazione Create DBCluster Amazon RDS API.

- `DatabaseInsightsMode` Da a `advanced`
- `Engine` Da a `aurora-postgresql`
- `EngineVersion`a una versione del motore disponibile per Limitless Database
- `StorageType` Da a `aurora-iopt1`
- `ClusterScalabilityType` Da a `limitless`
- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` Da a `True`
- `PerformanceInsightsRetentionPeriod`per almeno giorni `465`
- `MonitoringInterval`a un valore che non lo è `0`

- `MonitoringRoleArn`all'Amazon Resource Name (ARN) per il ruolo IAM che consente a RDS di inviare metriche di Enhanced Monitoring ad Amazon Logs CloudWatch

Attivazione della modalità avanzata di Database Insights durante la modifica di un cluster DB per Aurora PostgreSQL Limitless Database

Attiva Database Insights quando modifichi un database per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Per abilitare Database Insights, ogni istanza DB nel cluster ADb deve avere le stesse impostazioni di Performance Insights e Enhanced Monitoring.

Console

Nella console, è possibile attivare la modalità avanzata di Database Insights quando si modifica un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità avanzata di Database Insights quando si modifica un cluster DB utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli un cluster DB e scegli Modifica.
4. Nella sezione Database Insights, seleziona la modalità Avanzata. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Il periodo di conservazione deve essere di 15-24 mesi per la modalità avanzata di Database Insights.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegli Continue (Continua).

6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se si sceglie Applica durante la successiva finestra di manutenzione programmata, il database ignora questa impostazione e attiva immediatamente la modalità avanzata di Database Insights.
7. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per attivare la modalità avanzata di Database Insights durante la modifica di un cluster DB, chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--database-insights-mode advanced`per attivare la modalità avanzata di Database Insights.
- `--db-cluster-identifier`— L'identificatore per il cluster DB.
- `--enable-performance-insights`per attivare Performance Insights for Database Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione dei dati per l'. Per attivare la modalità avanzata di Database Insights, il periodo di conservazione deve essere di almeno 465 giorni.

L'esempio seguente abilita la modalità avanzata di Database Insights durante la modifica di un cluster DB.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Per attivare la modalità avanzata di Database Insights quando modifichi un cluster DB, specifica i seguenti parametri per il funzionamento dell'API Modify DBCluster Amazon RDS.

- `DatabaseInsightsMode` Da a `advanced`
- `EnablePerformanceInsights` Da a `True`
- `PerformanceInsightsRetentionPeriod`per almeno giorni `465`

Attivazione della modalità Standard di Database Insights per Aurora PostgreSQL Limitless Database

Per attivare la modalità Standard di Database Insights per il database Aurora PostgreSQL Limitless, utilizzare le seguenti procedure.

Attivazione della modalità Standard di Database Insights durante la creazione di un cluster DB per Aurora PostgreSQL Limitless Database

Attiva la modalità Standard di Database Insights quando crei un database per Aurora PostgreSQL Limitless Database.

Console

Nella console, puoi attivare la modalità Standard di Database Insights quando crei un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità Standard di Database Insights durante la creazione di un cluster DB utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegliere Crea database.
4. Nella sezione Database Insights, seleziona la modalità Standard. Quindi, scegli le seguenti opzioni:

    - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Per creare un cluster DB per Aurora PostgreSQL Limitless Database, il periodo di conservazione deve essere di almeno 31 giorni.

- AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.

5. Scegliere Crea database.

AWS CLI

Per attivare la modalità Standard di Database Insights durante la creazione di un cluster DB, chiamate il create-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--db-cluster-identifier`— L'identificatore per il cluster DB.
- `--database-insights-mode standard`per attivare la modalità Standard di Database Insights.
- `--engine`— Il cluster DB deve utilizzare il motore `aurora-postgresql` DB.
- `--engine-version`— Il cluster DB deve utilizzare una delle versioni del motore DB:
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type`— Il cluster DB deve utilizzare la configurazione di archiviazione del cluster `aurora-iopt1` DB.
- `--cluster-scalability-type`— specifica la modalità di scalabilità del cluster Aurora DB. Se impostato su`limitless`, il cluster funziona come un database Aurora PostgreSQL Limitless. Se impostato su `standard` (impostazione predefinita), il cluster utilizza la normale creazione di istanze DB.

  > **Note**
  >
  > Non è possibile modificare questa impostazione dopo aver creato il cluster DB.

- `--master-username`— Il nome dell'utente principale per il cluster DB.
- `--master-user-password`— La password per l'utente principale.
- `--enable-performance-insights`per attivare Performance Insights for Database Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione dei dati per il cluster DB. Per creare un cluster DB per Aurora PostgreSQL Limitless Database, il periodo di conservazione deve essere di almeno 31 giorni.

- `--monitoring-interval`— L'intervallo, in secondi, tra i punti in cui vengono raccolte le metriche di Enhanced Monitoring per il cluster DB. Questo valore non può essere. `0`
- `--monitoring-role-arn`— Amazon Resource Name (ARN) per il ruolo IAM che consente a RDS di inviare metriche di Enhanced Monitoring ad Amazon Logs. CloudWatch
- `--enable-cloudwatch-logs-exports`— È necessario esportare i log in Logs. `postgresql` CloudWatch

L'esempio seguente abilita la modalità Standard di Database Insights durante la creazione di un cluster DB.

Per Linux, macOS o Unix:

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode standard \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 31 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Per Windows:

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode standard ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 31 ^
--monitoring-interval 5 ^
```

```
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Per attivare la modalità Standard di Database Insights quando crei un', specifica i seguenti parametri per l'operazione Create DBCluster Amazon RDS API.

- `DatabaseInsightsMode` Da a `standard`
- `Engine` Da a `aurora-postgresql`
- `EngineVersion`a una versione del motore disponibile per Limitless Database
- `StorageType` Da a `aurora-iopt1`
- `ClusterScalabilityType` Da a `limitless`
- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` Da a `True`
- `PerformanceInsightsRetentionPeriod`per almeno giorni `31`
- `MonitoringInterval`a un valore che non lo è `0`
- `MonitoringRoleArn`all'Amazon Resource Name (ARN) per il ruolo IAM che consente a RDS di inviare metriche di Enhanced Monitoring ad Amazon Logs CloudWatch

Attivazione della modalità Standard di Database Insights durante la modifica di un cluster DB per Aurora PostgreSQL Limitless Database

Attiva Database Insights quando modifichi un database per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Per abilitare Database Insights, ogni istanza DB in un deve avere le stesse impostazioni di Performance Insights e Enhanced Monitoring.

Console

Nella console, è possibile attivare la modalità Standard di Database Insights quando si crea un cluster DB. Le impostazioni per Database Insights si applicano a tutte le istanze DB del cluster DB.

Per attivare la modalità Standard di Database Insights quando si modifica un cluster DB utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli un cluster DB e scegli Modifica.
4. Nella sezione Database Insights, seleziona la modalità Standard. Quindi, scegli le seguenti opzioni:
   - Retention (Conservazione) – Quantità di tempo per cui conservare i dati di Performance Insights. Per creare un cluster DB per Aurora PostgreSQL Limitless Database, il periodo di conservazione deve essere di almeno 31 giorni.
   - AWS KMS key: specificare la chiave KMS. Performance Insights crittografa tutti i dati potenzialmente sensibili con la chiave KMS. I dati vengono crittografati mentre sono in transito o inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
5. Scegli Continue (Continua).
6. In Scheduling of Modifications (Pianificazione delle modifiche), scegliere Apply immediately (Applica immediatamente). Se scegli Applica durante la successiva finestra di manutenzione programmata, il database ignora questa impostazione e attiva immediatamente la modalità Standard di Database Insights.
7. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per attivare la modalità Standard di Database Insights durante la modifica di un cluster DB, chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:

- `--db-cluster-identifier`— L'identificatore per il cluster DB.
- `--database-insights-mode standard`per attivare la modalità Standard di Database Insights.
- `--enable-performance-insights`per attivare Performance Insights for Database Insights.
- `--performance-insights-retention-period`— Il periodo di conservazione dei dati per il cluster DB. Per attivare la modalità Standard di Database Insights, il periodo di conservazione deve essere di almeno 31 giorni.

L'esempio seguente abilita la modalità Standard di Database Insights durante la modifica di un cluster DB.

Per Linux, macOS o Unix:

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 31
```

Per Windows:

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 31
```

RDS API

Per attivare la modalità Standard di Database Insights quando modifichi un cluster DB, specifica i seguenti parametri per il funzionamento dell'API Modify DBCluster Amazon RDS.

- `DatabaseInsightsMode` Da a `standard`
- `EnablePerformanceInsights` Da a `True`
- `PerformanceInsightsRetentionPeriod`per almeno giorni `31`

Monitoraggio del database Aurora Postgre SQL Limitless con Amazon Logs CloudWatch

L'esportazione dei log di Postgre in SQL CloudWatch Logs è richiesta come parte dell'abilitazione di Aurora Postgre Limitless Database. SQL È possibile accedere e analizzare questi log in CloudWatch Logs Insights, in modo analogo all'accesso ai SQL log di Postgre per un cluster Aurora Postgre DB standard. SQL Per ulteriori informazioni, consulta Analisi dei log PostgreSQL con Logs Insights CloudWatch .

Il nome del gruppo di log per il cluster DB è lo stesso di Aurora Postgre: SQL

```
/aws/rds/cluster/DB_cluster_ID/postgresql
```

Il nome del gruppo di log per il gruppo di shard DB assume la forma seguente:

```
/aws/rds/cluster/DB_cluster_ID/DB_shard_group_ID/postgresql
```

Esistono flussi di log per ogni nodo (router o shard). I loro nomi hanno la forma seguente:

```
[DistributedTransactionRouter|
DataAccessShard]/node_cluster_serial_ID-node_instance_serial_ID/n
```

Per esempio:

- Router — `DistributedTransactionRouter/6-6.2`
- Frammento — `DataAccessShard/22-22.0`

> **Note**
>
> Non è possibile visualizzare i file di SQL registro di Postgre per il gruppo di shard DB direttamente nella RDS console o RDS API come è possibile per il cluster DB. AWS CLIÈ necessario utilizzare CloudWatch Logs Insights per visualizzarli.

Monitoraggio del database Aurora Postgre SQL Limitless con monitoraggio avanzato

Il monitoraggio avanzato è necessario come parte dell'abilitazione del database Aurora SQL Postgre Limitless. Puoi usarlo per monitorare il sistema operativo delle tue istanze DB di Limitless Database in tempo reale.

Aurora pubblica le metriche di monitoraggio avanzato nei registri. CloudWatch Alcune delle metriche chiave disponibili includono le connessioni al database, l'utilizzo dello storage e la latenza delle query. Queste possono aiutare a identificare i punti deboli in termini di prestazioni.

Per ulteriori informazioni sulle metriche di Enhanced Monitoring, consulta. Parametri del sistema operativo per Aurora

Monitoraggio del database Aurora PostgreSQL Limitless con Performance Insights

Usa Performance Insights per monitorare il tuo cluster Aurora PostgreSQL Limitless Database. Performance Insights funziona in modo simile per Aurora PostgreSQL Limitless Database e per i cluster Aurora DB standard. Tuttavia, tieni traccia delle metriche a livello di gruppo di shard per Aurora PostgreSQL Limitless Database.

Le due metriche principali di Performance Insights da monitorare sono le seguenti:

- Caricamento del database: misura il livello di attività nel database. Il parametro chiave in Performance Insights è `DBLoad`, che viene raccolto ogni secondo.

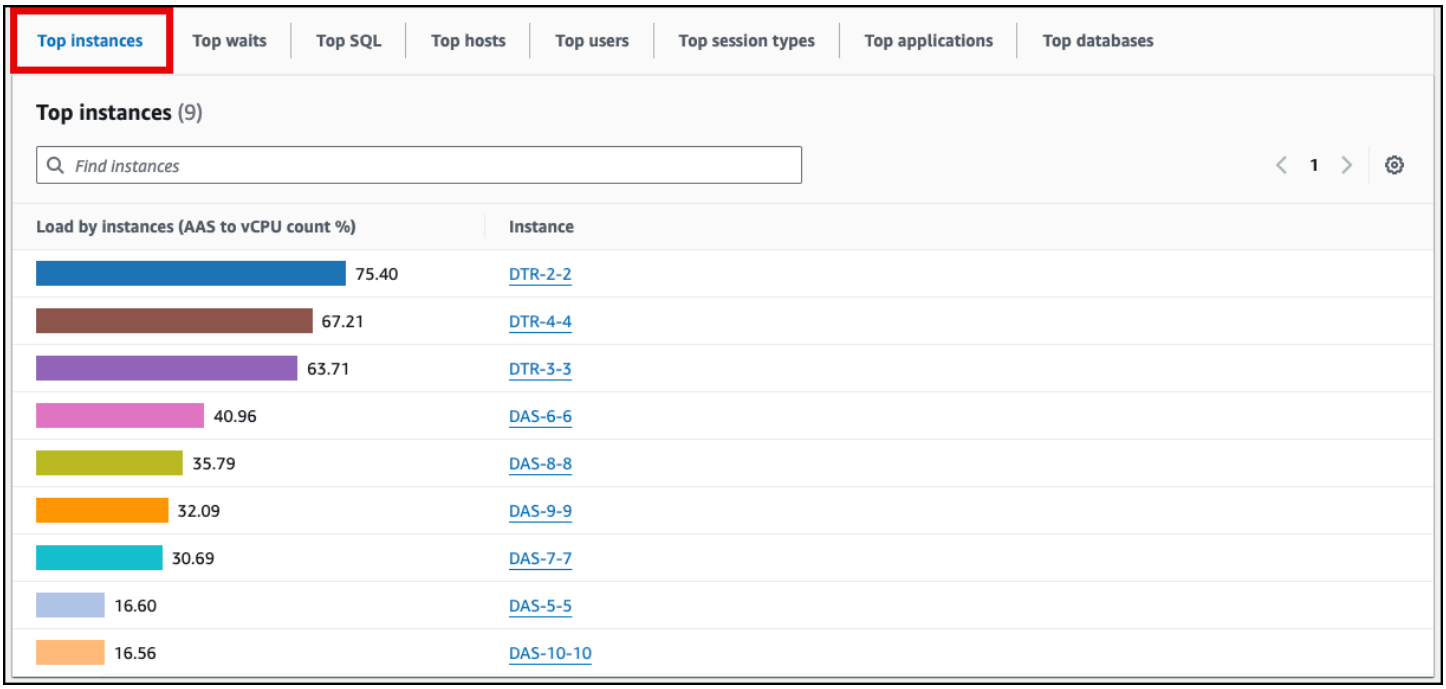
  L'unità per la `DBLoad` metrica in Performance Insights è la media delle sessioni attive (AAS). Per ottenere le sessioni attive medie (AAS), Performance Insights esegue il campionamento del numero di sessioni che eseguono contemporaneamente una query. L'AAS è il numero totale di sessioni diviso per il numero totale di campioni per un periodo di tempo specifico. Per ulteriori informazioni su `DBLoad` e AAS, vedereCaricamento database.
- CPU massima: la massima potenza di calcolo disponibile per il database. Per vedere se le sessioni attive superano la CPU massima, esamina la loro relazione con la `Max vCPU` linea. Il `Max vCPU` valore è determinato dal numero di core vCPU (CPU virtuale) per l'istanza DB. Per ulteriori informazioni su `Max vCPU`, consulta Massimo CPU.

Inoltre, puoi «suddividere» la `DBLoad` metrica in dimensioni, che sono sottocategorie della metrica. Le dimensioni più utili sono le seguenti:

- Istanze principali: mostra il carico database relativo per le istanze (shard e router) in ordine decrescente.
- Eventi di attesa: consente alle istruzioni SQL di attendere che si verifichino eventi specifici prima di poter continuare l'esecuzione. Gli eventi di attesa indicano dove il lavoro è ostacolato.
- Top SQL: mostra quali query contribuiscono maggiormente al caricamento del DB.

Per ulteriori informazioni sulle dimensioni di Performance Insights, vedereDimensioni.

La figura seguente mostra la dimensione Top instances per un gruppo di shard DB.



Argomenti

- [Analisi del carico del DB per Aurora SQL Postgre Limitless Database utilizzando la dashboard Performance Insights]

Analisi del carico del DB per Aurora SQL Postgre Limitless Database utilizzando la dashboard Performance Insights

Con Performance Insights, puoi tenere traccia delle metriche a livello di gruppo di shard e a livello di istanza per un database Aurora Postgre Limitless. SQL Quando si analizza il carico del DB per un database Aurora SQL Postgre Limitless, è possibile confrontare il carico del DB per ogni shard e router con il valore massimo v. CPU

> **Note**
>
> Aurora Postgre SQL Limitless Database ha sempre Performance Insights e Enhanced Monitoring abilitati. Il periodo minimo di conservazione dei dati di Performance Insights per Limitless Database è di 31 giorni (1 mese).

La visualizzazione Assoluta mostra il numero medio di sessioni attive (AAS) e la stima vCPU. La vista relativa mostra il rapporto tra e AAS il valore stimato di vCPU.

Argomenti

- Analisi del carico relativo del DB per Aurora SQL Postgre Limitless Database utilizzando la dashboard Performance Insights
- Analisi del carico del DB in base alle attese per Aurora Postgre Limitless SQL Database utilizzando la dashboard Performance Insights
- Analisi della distribuzione del carico per Aurora SQL Postgre Limitless Database utilizzando la dashboard Performance Insights

Analisi del carico relativo del DB per Aurora SQL Postgre Limitless Database utilizzando la dashboard Performance Insights

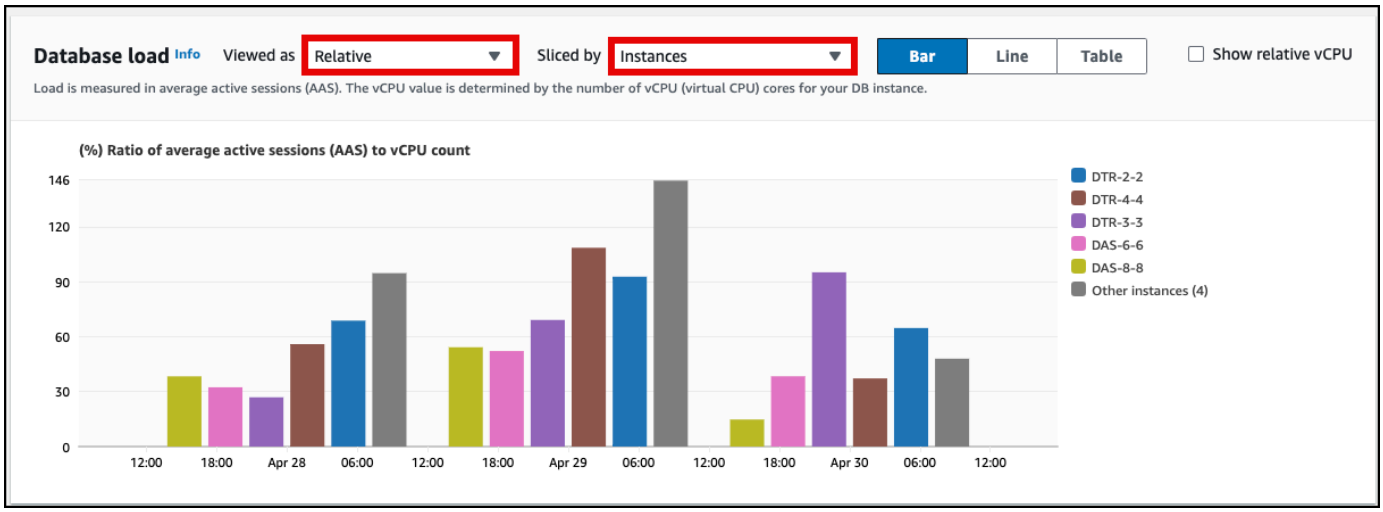
Potresti voler migliorare le prestazioni del tuo database Aurora Postgre SQL Limitless monitorando il carico relativo del DB. Per analizzare il carico relativo del DB per istanza per il database Aurora Postgre SQL Limitless, utilizzare la procedura seguente.

Per analizzare il carico relativo del DB per Aurora Postgre SQL Limitless Database utilizzando la console

1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

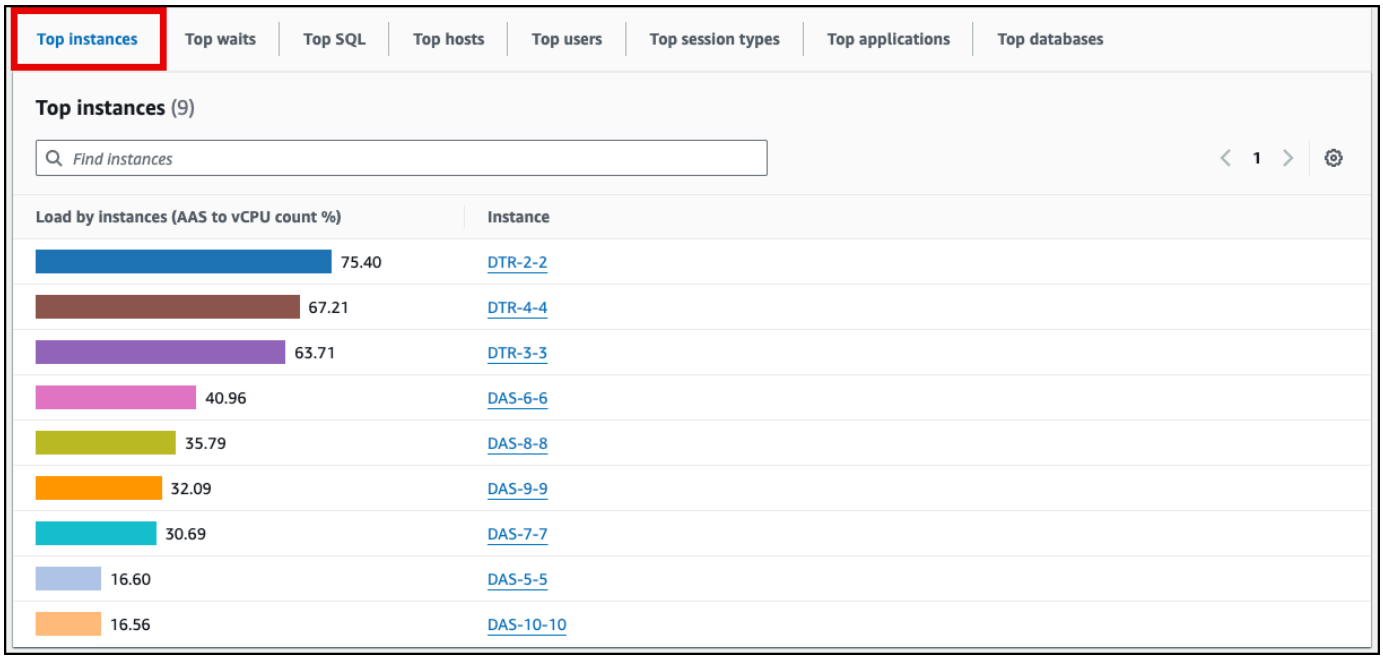
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un database Aurora SQL Postgre Limitless. Viene visualizzata la dashboard Performance Insights per quel database Aurora SQL Postgre Limitless.
4. Nella sezione Caricamento del database (caricamento DB), scegli Istanze per Sliced by. Per vedere il rapporto tra le sessioni attive medie (AAS) e i v CPU core per tutte le istanze del tuo database Aurora Postgre SQL Limitless, scegli Relativo per Visualizzato come.

   Il grafico delle sessioni attive medie mostra il carico del DB per le istanze nel database Aurora SQL Postgre Limitless.



5. Per visualizzare le istanze principali, scegli la scheda Prime istanze.

   Nell'esempio seguente, l'istanza con il carico DB più elevato è. `DTR-2-2`



6. (Facoltativo) Per analizzare il carico del DB per un'istanza nel tuo database Aurora Postgre SQL Limitless, scegli il nome dell'istanza nella colonna Istanze. Per visualizzare il carico del DB**DTR-2-2**, scegliete **DTR-2-2** nella colonna Istanze.

> **Note**
>
> È possibile visualizzare le metriche di Performance Insights solo per le istanze in un database Aurora Postgre Limitless. SQL

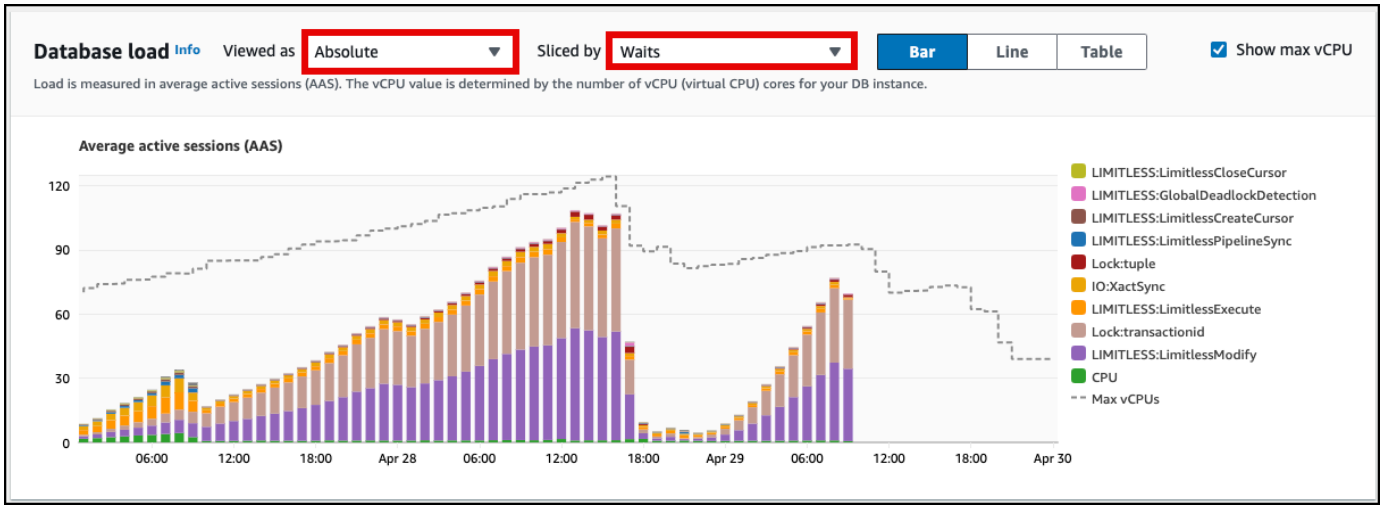
Analisi del carico del DB in base alle attese per Aurora Postgre Limitless SQL Database utilizzando la dashboard Performance Insights

Potresti voler migliorare le prestazioni del tuo database Aurora Postgre SQL Limitless monitorando gli eventi di attesa. Per analizzare gli eventi di caricamento del DB in base all'attesa per il database Aurora Postgre SQL Limitless, utilizzare la procedura seguente.

Per analizzare il carico del DB in base alle attese di Aurora SQL Postgre Limitless Database utilizzando la console

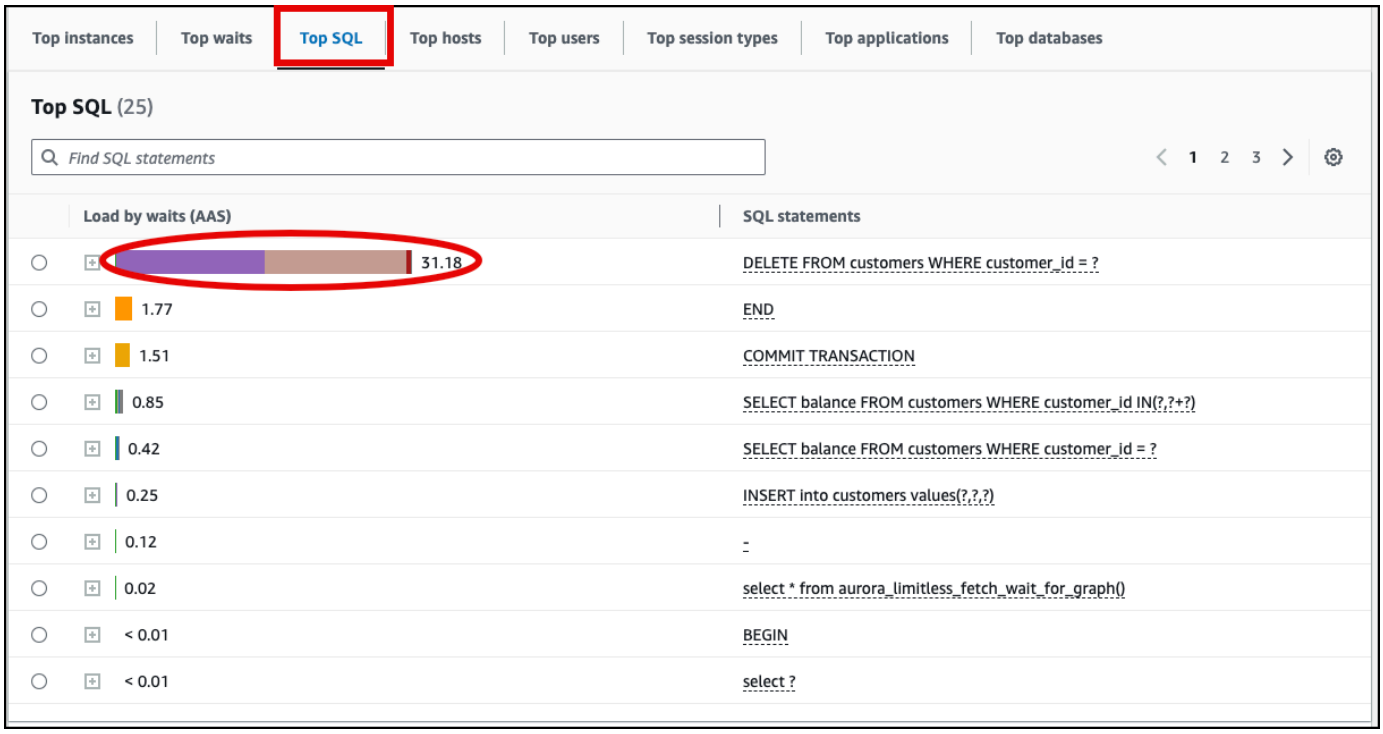
1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un database Aurora SQL Postgre Limitless. Viene visualizzata la dashboard Performance Insights per quel database Aurora SQL Postgre Limitless.
4. Nella sezione Caricamento del database (caricamento DB), scegli Waits for Sliced by. Per visualizzare il numero di v AAS e la stimaCPU, scegli Absolute per Viewed as.

   Il grafico delle sessioni attive medie mostra il carico del DB per le istanze nel database Aurora SQL Postgre Limitless.



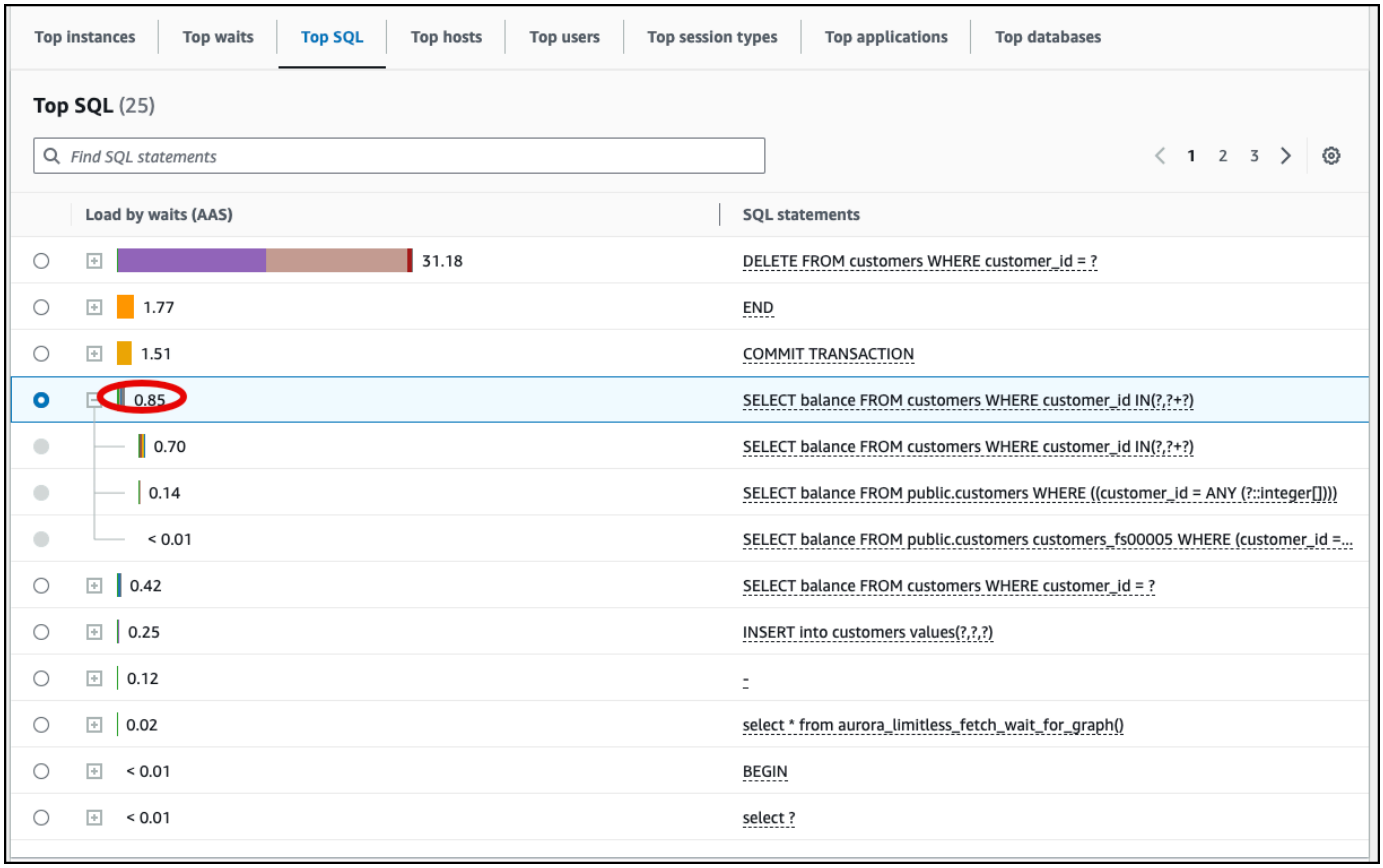
5. Scorri verso il basso fino alla scheda Top SQL.

   Nell'esempio seguente, l'SQListruzione con il carico di attesa più elevato è l'DELETEistruzione.



6. Scegliete l'SQListruzione per espanderla nelle istruzioni che lo compongono.

   Nell'esempio seguente, l'SELECTistruzione ha 3 istruzioni componenti.



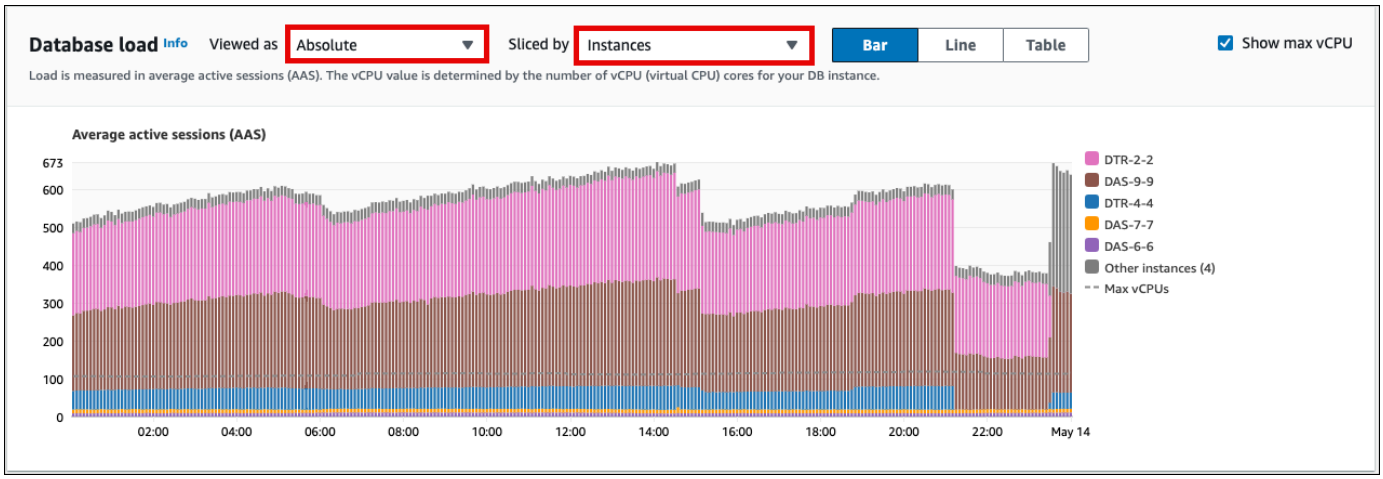
Analisi della distribuzione del carico per Aurora SQL Postgre Limitless Database utilizzando la dashboard Performance Insights

Potresti voler bilanciare la distribuzione del carico per le istanze sul tuo database Aurora SQL Postgre Limitless. Per analizzare la distribuzione del carico delle istanze su un database Aurora SQL Postgre Limitless, utilizzare la procedura seguente.

Per analizzare la distribuzione del carico delle istanze su un database Aurora SQL Postgre Limitless utilizzando la console

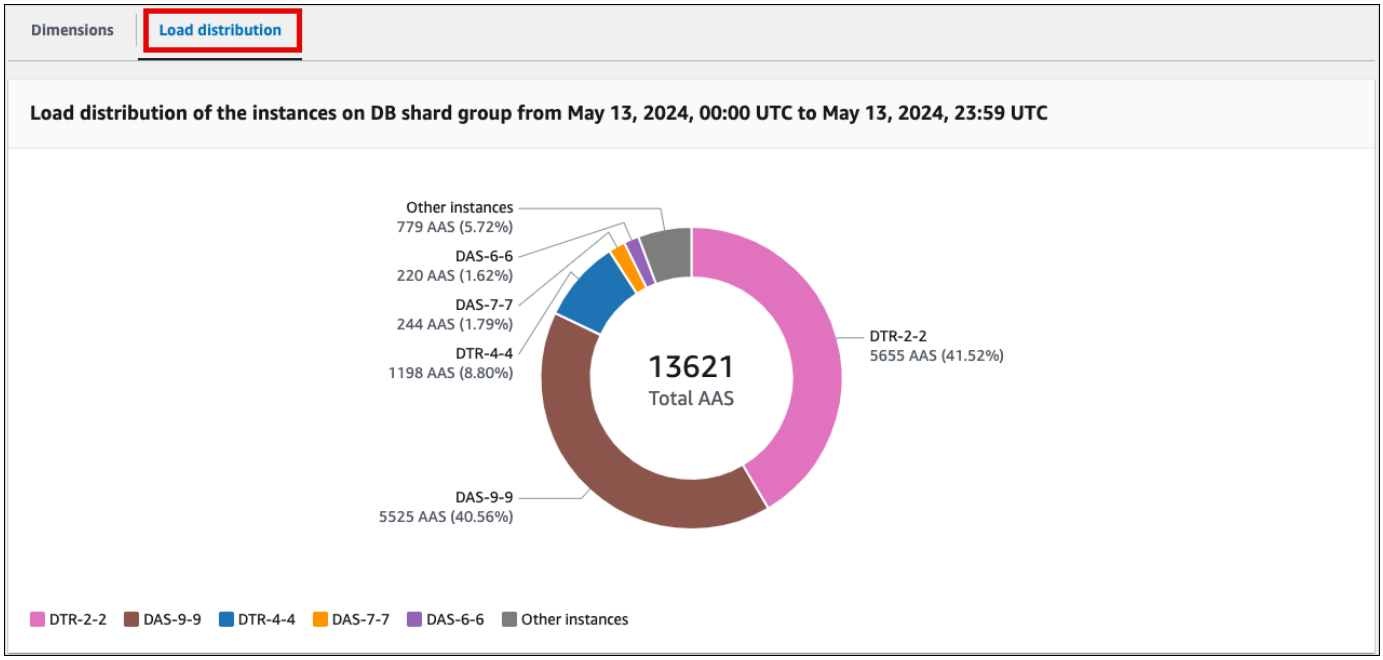
1. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione scegli Approfondimenti sulle prestazioni.
3. Scegli un database Aurora SQL Postgre Limitless. Viene visualizzata la dashboard Performance Insights per quel database Aurora SQL Postgre Limitless.
4. Nella sezione Caricamento del database (caricamento DB), scegli Istanze per Sliced by. Per visualizzare il numero AAS e la stima di v CPU per tutte le istanze nel database Aurora SQL Postgre Limitless, scegli Assoluto per Visualizzato come.

Il grafico delle sessioni attive medie mostra il carico del DB per le istanze nel database Aurora SQL Postgre Limitless.



5. Per visualizzare un grafico della distribuzione del carico delle istanze nel database Aurora SQL Postgre Limitless, scegli la scheda Distribuzione del carico.

   Nell'esempio seguente, l'istanza con il carico DB più elevato è. `DTR-2-2`



Monitoraggio del database Aurora Postgre SQL Limitless con Amazon Protection GuardDuty RDS

Amazon GuardDuty è un servizio di rilevamento delle minacce che aiuta a proteggere account, contenitori, carichi di lavoro e dati all'interno del tuo AWS ambiente. Utilizzando modelli di machine learning (ML) e funzionalità di rilevamento di anomalie e minacce, monitora GuardDuty continuamente diverse fonti di log e attività di runtime per identificare e dare priorità ai potenziali rischi per la sicurezza e alle attività dannose nel tuo ambiente.

GuardDuty RDSLa protezione analizza e profila gli eventi di accesso per potenziali minacce di accesso ai database Amazon Aurora. Quando attivi la RDS protezione, GuardDuty consuma gli eventi di RDS accesso dai tuoi database Aurora. RDS La protezione monitora questi eventi e li profila per potenziali minacce interne o attori esterni.

Per ulteriori informazioni sulla GuardDuty RDS protezione in Aurora, consulta. Monitoraggio delle minacce con Amazon GuardDuty RDS Protection per Amazon Aurora

Per ulteriori informazioni sull'attivazione della GuardDuty RDS protezione, consulta la sezione GuardDuty RDSProtezione nella Amazon GuardDuty User Guide.

Funzioni e visualizzazioni per Aurora SQL Postgre Limitless Database

Aurora Postgre SQL Limitless Database ha aggiunto funzioni e visualizzazioni. Si basano sulle funzioni e sulle viste corrispondenti di Aurora Postgre. SQL

> **Note**
>
> Alcune statistiche possono restituire risultati incoerenti se ci sono transazioni in corso.

Argomenti

- Funzioni del database Aurora SQL Postgre Limitless
- Visualizzazioni del database Aurora PostgreSQL Limitless

Funzioni del database Aurora SQL Postgre Limitless

La tabella seguente mostra le nuove funzioni per Aurora SQL Postgre Limitless Database.

> **Note**
>
> Le funzioni elencate in questa tabella si trovano nello schema. `rds_aurora` Quando usi una funzione Limitless Database, assicurati di includere il nome completo dell'oggetto:`rds_aurora.` *`object_name`*.

Funzione Aurora SQL Postgre Limitless Database	Funzione Aurora Postgre corrispondente SQL
limitless_backend_dsid	pg_backend_pid
sessione_annullamento_illimitata	pg_cancel_backend
limitless_stat_clear_snapshot	pg_stat_clear_snapshot
dimensione_stat_database_limitate	pg_database_size
limitless_stat_get_snapshot_timestamp	pg_stat_get_snapshot_timestamp

Funzione Aurora SQL Postgre Limitless Database	Funzione Aurora Postgre corrispondente SQL
stat_prepared_xacts senza limiti	pg_prepared_xacts
dimensioni_relazioni_stat_limitate	pg_indexes_size, pg_relation_size, pg_table_size, pg_total_relation_size
limitless_stat_reset	pg_stat_reset
limitless_stat_statements_reset	pg_stat_statements_reset
limitless_stat_system_waits	aurora_stat_system_waits
limitless_terminate_session	pg_terminate_backend
rapporto di attesa illimitato	aurora_wait_report

Gli esempi seguenti forniscono dettagli sulle funzioni del database Aurora Postgre SQL Limitless. Per ulteriori informazioni sulle funzioni di Postgre, consulta SQL Funzioni e operatori nella documentazione di Postgre. SQL

limitless_backend_dsid

La `limitless_backend_dsid` funzione restituisce l'ID di sessione distribuito per la sessione corrente. Una sessione distribuita viene eseguita su un router in un gruppo di shard DB e coinvolge processi di backend su uno o più shard nel gruppo di shard DB.

L'esempio seguente mostra come utilizzare la funzione. `limitless_backend_dsid`

```
SELECT rds_aurora.limitless_backend_dsid();

limitless_backend_dsid
------------------------
8CACD7B04D0FC2A5
(1 row)
```

limitless_cancel_session

La `limitless_cancel_session` funzione funziona in modo simile a`pg_cancel_backend`, ma tenta di annullare tutti i processi di backend relativi all'ID di sessione distribuito fornito inviando un (segnale di interruzione). `SIGINT`

Il parametro di input è il seguente:

- `distributed_session_id`(testo) — L'ID della sessione distribuita da annullare.

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `pid`(text) — L'ID del processo di backend.
- `success`(boolean) — Indica se l'annullamento è andato a buon fine.

L'esempio seguente mostra come utilizzare la `limitless_cancel_session` funzione.

```
SELECT * FROM rds_aurora.limitless_cancel_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_stat_clear_snapshot

La `limitless_stat_clear_snapshot` funzione elimina l'istantanea delle statistiche correnti o le informazioni memorizzate nella cache su tutti i nodi.

L'esempio seguente mostra come utilizzare la funzione. `limitless_stat_clear_snapshot`

```
SELECT rds_aurora.limitless_stat_clear_snapshot();
```

limitless_stat_database_size

La `limitless_stat_database_size` funzione restituisce le dimensioni di un database nel gruppo di shard DB.

Il parametro di input è il seguente:

- `dbname`(name) — Il database per il quale ottenere le dimensioni.

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`
- `db_size`— La dimensione del database in questo sottocluster, in byte.

L'esempio seguente mostra come utilizzare la `limitless_stat_database_size` funzione.

```
SELECT * FROM rds_aurora.limitless_stat_database_size('postgres_limitless');

 subcluster_id | subcluster_type | db_size
---------------+-----------------+----------
             1 | router          |  8895919
             2 | router          |  8904111
             3 | shard           | 21929391
             4 | shard           | 21913007
             5 | shard           | 21831087
(5 rows)
```

limitless_stat_get_snapshot_timestamp

La `limitless_stat_get_snapshot_timestamp` funzione restituisce il timestamp dell'istantanea delle statistiche corrente o se non è stata scattata alcuna istantanea delle statistiche. NULL Un'istantanea viene scattata la prima volta che si accede alle statistiche cumulative in una transazione, se è impostata su. `stats_fetch_consistency` `snapshot` Restituisce una vista consolidata dei timestamp delle istantanee da tutti i nodi. Le `subcluster_type` colonne `subcluster_id` e mostrano da quale nodo provengono i dati.

L'esempio seguente mostra come utilizzare la `limitless_stat_get_snapshot_timestamp` funzione.

```
SELECT * FROM rds_aurora.limitless_stat_get_snapshot_timestamp();

 subcluster_id | subcluster_type | snapshot_timestamp
---------------+-----------------+--------------------
             1 | router          |
             2 | router          |
             3 | shard           |
             4 | shard           |
             5 | shard           |
```

```
(5 rows)
```

limitless_stat_prepared_xacts

La `limitless_stat_prepared_xacts` funzione restituisce informazioni sulle transazioni su tutti i nodi attualmente preparati per il commit in due fasi. Per ulteriori informazioni, consulta pg_prepared_xacts nella documentazione di Postgre. SQL

L'esempio seguente mostra come usare la funzione. `limitless_stat_prepared_xacts`

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_prepared_xacts;

 subcluster_id | subcluster_type | transaction_id |             gid              |
         prepared            |  owner_id  |    database_id
---------------+-----------------+----------------+------------------------------
+-------------------------------+------------+--------------------
 8             | shard           |        5815978 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659603+00 | auroraperf | postgres_limitless
 12            | shard           |        4599138 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659637+00 | auroraperf | postgres_limitless
(2 rows)
```

limitless_stat_relation_sizes

La `limitless_stat_relation_sizes` funzione restituisce le diverse dimensioni di una tabella nel gruppo di shard DB.

I parametri di input sono i seguenti:

- `relnspname`(name) — Il nome dello schema contenente la tabella.
- `relname`(name) — Il nome della tabella.

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`
- `main_size`— La dimensione in byte del fork di dati principale in questo nodo.
- `fsm_size`— La dimensione in byte della mappa dello spazio libero per la tabella in questo nodo.
- `vm_size`— La dimensione in byte della mappa di visibilità per la tabella in questo nodo.

- `init_size`— La dimensione in byte dell'inizializzazione della tabella in questo nodo.
- `toast_size`— La dimensione in byte della tabella toast associata alla tabella in questo fork.
- `index_size`— La dimensione in byte di tutti gli indici della tabella in questo nodo.
- `total_size`— La dimensione in byte di tutti i segmenti della tabella in questo nodo.

L'esempio seguente mostra come utilizzare la `limitless_stat_relation_sizes` funzione (alcune colonne vengono omesse).

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customers');

 subcluster_id | subcluster_type | main_size | fsm_size | vm_size | toast_size |
 table_size | total_size
---------------+-----------------+-----------+----------+---------+------------
+------------+------------
             1 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             2 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             3 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             4 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             5 | shard           |   3981312 |  4227072 | 1409024 |    1409024 |
 11026432 |   11026432
(5 rows)
```

limitless_stat_reset

La `limitless_stat_reset` funzione azzera tutti i contatori delle statistiche per il database corrente (0). Se `track_functions` è abilitata, la `stats_reset` colonna in alto `limitless_stat_database` mostra l'ultima volta che le statistiche sono state reimpostate per il database. Per impostazione predefinita, `limitless_stat_reset` può essere eseguita solo da un superutente. È possibile concedere l'autorizzazione ad altri utenti utilizzando il `EXECUTE` privilegio.

L'esempio seguente mostra come utilizzare la `limitless_stat_reset` funzione.

```
SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

 tup_inserted | tup_deleted
```

```
--------------+-------------
          896 |           0
(1 row)

SELECT rds_aurora.limitless_stat_reset();

limitless_stat_reset
---------------------
(1 row)

SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

tup_inserted | tup_deleted
-------------+-------------
           0 |           0
(1 row)
```

limitless_stat_statements_reset

La `limitless_stat_statements_reset` funzione elimina le statistiche raccolte finora in base `limitless_stat_statements` ai parametri specificati,, e. `username` `dbname` `distributed_query_id` `queryid` Se uno qualsiasi dei parametri non viene specificato, per ciascuno di essi viene utilizzato il valore predefinito `""` o `0` (non valido) e le statistiche che corrispondono agli altri parametri vengono ripristinate. Se non viene specificato alcun parametro o tutti i parametri specificati sono `""` o `0` (non validi), la funzione scarta tutte le statistiche. Se tutte le statistiche nella `limitless_stat_statements` vista vengono scartate, la funzione reimposta anche le statistiche nella vista. `limitless_stat_statements_info`

I parametri di input sono i seguenti:

- `username`(nome) — L'utente che ha richiesto l'istruzione.
- `dbname`(name) — Il database in cui è stata eseguita la query.
- `distributed_query_id`(bigint) — L'ID della query principale proveniente dal nodo coordinatore. Questa colonna indica `NULL` se è la query principale. Il nodo coordinatore invia l'ID della query distribuita ai nodi partecipanti. Pertanto, per i nodi partecipanti, i valori per l'ID della query distribuito e l'ID della query sono diversi.
- `queryid`(bigint) — L'ID della query dell'istruzione.

L'esempio seguente mostra come utilizzare la `limitless_stat_statements_reset` funzione per reimpostare tutte le statistiche raccolte da`limitless_stat_statements`.

```
SELECT rds_aurora.limitless_stat_statements_reset();
```

limitless_stat_system_waits

La `limitless_stat_system_waits` funzione restituisce una visualizzazione consolidata dei dati degli eventi di attesa`aurora_stat_system_waits`, che riporta l'attività di attesa a livello di sistema in un'istanza, da tutti i nodi. Le `subcluster_type` colonne `subcluster_id` e mostrano da quale nodo provengono i dati.

L'esempio seguente mostra come utilizzare la `limitless_stat_system_waits` funzione.

```
postgres_limitless=> SELECT *
FROM rds_aurora.limitless_stat_system_waits() lssw, pg_catalog.aurora_stat_wait_event()
 aswe
WHERE lssw.event_id=aswe.event_id and aswe.event_name='LimitlessTaskScheduler';

 subcluster_id | subcluster_type | type_id | event_id  | waits  |  wait_time   |
 event_name
---------------+-----------------+---------+-----------+--------+--------------
+------------------------
             1 | router          |      12 | 201326607 | 677068 | 616942216307 |
 LimitlessTaskScheduler
             2 | router          |      12 | 201326607 | 678586 | 616939897111 |
 LimitlessTaskScheduler
             3 | shard           |      12 | 201326607 | 756640 | 616965545172 |
 LimitlessTaskScheduler
             4 | shard           |      12 | 201326607 | 755184 | 616958057620 |
 LimitlessTaskScheduler
             5 | shard           |      12 | 201326607 | 757522 | 616963183539 |
 LimitlessTaskScheduler
(5 rows)
```

limitless_terminate_session

La `limitless_terminate_session` funzione funziona in modo simile a`pg_terminate_backend`, ma tenta di terminare tutti i processi di backend relativi all'ID di sessione distribuito fornito inviando un (segnale di fine). `SIGTERM`

Il parametro di input è il seguente:

- `distributed_session_id`(testo) — L'ID della sessione distribuita da terminare.

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `pid`(text) — L'ID del processo di backend.
- `success`(boolean) — Indica se il processo è stato terminato con successo.

L'esempio seguente mostra come utilizzare la `limitless_terminate_session` funzione.

```
SELECT * FROM rds_aurora.limitless_terminate_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_wait_report

La `limitless_wait_report` funzione restituisce l'attività dell'evento di attesa in un periodo di tempo da tutti i nodi. Le `subcluster_type` colonne `subcluster_id` e mostrano da quale nodo provengono i dati.

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`

Il resto delle colonne è uguale a quello in`aurora_wait_report`.

L'esempio seguente mostra come utilizzare la `limitless_wait_report` funzione.

```
postgres_limitless=> select * from rds_aurora.limitless_wait_report();

 subcluster_id | subcluster_type | type_name | event_name | waits | wait_time |
 ms_per_wait | waits_per_xact | ms_per_xact
---------------+-----------------+-----------+------------+-------+-----------
+-------------+--------------- +-------------
             1 | router          | Client    | ClientRead |    57 | 741550.14 |
 13009.652 |           0.19 |    2505.237
             5 | shard           | Client    | ClientRead |    54 | 738897.68 |
 13683.290 |           0.18 |    2496.276
             4 | shard           | Client    | ClientRead |    54 | 738859.53 |
 13682.584 |           0.18 |    2496.147
```

```
            2 | router          | Client    | ClientRead |    53 | 719223.64 |
13570.257 |           0.18 |    2429.810
            3 | shard           | Client    | ClientRead |    54 | 461720.40 |
8550.378 |           0.18 |    1559.86
```

Visualizzazioni del database Aurora PostgreSQL Limitless

La tabella seguente mostra le nuove viste per Aurora PostgreSQL Limitless Database.

> **Note**
>
> Le viste elencate in questa tabella si trovano nello schema. `rds_aurora` Quando utilizzi una vista Limitless Database, assicurati di includere il nome completo dell'oggetto:`rds_aurora`. *`object_name`*.

Visualizzazione del database Aurora PostgreSQL Limitless	Vista Aurora PostgreSQL corrispondente
limitless_database	pg_database
limitless_locks	pg_locks
attività_stat_illimitata	pg_stat_activity
limitless_stat_all_indexes	pg_stat_all_indexes
limitless_stat_all_tables	pg_stat_all_tables
limitless_stat_database	pg_stat_database
stat_progress_vacuum senza limiti	pg_stat_progress_vacuum
dichiarazioni_stat_limitless	pg_stat_statements
limitless_stat_subclusters	Nessuno
limitless_stat_statements_info	pg_stat_statements_info
limitless_statio_all_indexes	pg_statio_all_indexes
limitless_statio_all_tables	pg_statio_all_tables
tabelle_illimitate	pg_tables
table_collocazioni senza limiti	Nessuno

Visualizzazione del database Aurora PostgreSQL Limitless	Vista Aurora PostgreSQL corrispondente
distribuzioni_table_collocazione_limitate	Nessuno

Gli esempi seguenti forniscono dettagli sulle viste del database Aurora PostgreSQL Limitless. Per ulteriori informazioni sulle viste di PostgreSQL, vedere Visualizzazione delle statistiche nella documentazione di PostgreSQL.

> **Note**
>
> Alcune visualizzazioni statistiche possono restituire risultati incoerenti se hai transazioni in corso.

limitless_database

Questa vista contiene informazioni sui database disponibili nel gruppo di shard DB. Per esempio:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, oid, datname, datacl
 FROM rds_aurora.limitless_database;

 subcluster_id | subcluster_type |  oid  |      datname       |
                                 datacl

---------------+-----------------+-------+--------------------
+-------------------------------------------------------------------------------------------
 2             | router          |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 2             | router          |     5 | postgres           |
 2             | router          | 16384 | rdsadmin           |
 {rdsadmin=CTc/rdsadmin,rds_aurora_limitless_metadata_admin=c/
rdsadmin,rds_aurora_limitless_heat_mgmt_admin=c/rdsadmin}
 2             | router          | 16477 | postgres_limitless |
 2             | router          |     1 | template1          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 6             | shard           |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
```

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster (nodo)
- `subcluster_type`(testo) — Il tipo di sottocluster (nodo), router o shard

Le altre colonne sono le stesse di. `pg_database`

limitless_locks

Questa vista contiene una riga per processo per nodo. Fornisce l'accesso alle informazioni sui blocchi mantenuti dai processi attivi nel server di database.

Example di creare un lucchetto con due transazioni

In questo esempio, eseguiamo due transazioni contemporaneamente su due router.

```
# Transaction 1 (run on router 1)
BEGIN;
SET search_path = public;
SELECT * FROM customers;
INSERT INTO customers VALUES (400,'foo','bar');

# Transaction 2 (run on router 2)
BEGIN;
SET search_path = public;
ALTER TABLE customers ADD COLUMN phone VARCHAR;
```

Viene eseguita la prima transazione. Le transazioni successive devono attendere il completamento della prima transazione. Pertanto la seconda transazione viene bloccata con un lucchetto. Per verificarne la causa principale, eseguiamo un comando unendoci `limitless_locks` a`limitless_stat_activity`.

```
# Run on router 2
SELECT distributed_session_id, state, usename, query, query_start
FROM rds_aurora.limitless_stat_activity
WHERE distributed_session_id in (
SELECT distributed_session_id
FROM rds_aurora.limitless_locks
WHERE relname = 'customers'
);

 distributed_session_id | state               | usename                | query
                                  | query_start
------------------------+---------------------+-------------------------
+---------------------------------- -------------+-------------------------------
```

```
 47BDE66E9A5E8477       | idle in transaction | limitless_metadata_admin | INSERT INTO
 customers VALUES (400,'foo','bar'); | 2023-04-13 17:44:45.152244+00
 2AD7F370202D0FA9       | active              | limitless_metadata_admin | ALTER TABLE
 customers ADD COLUMN phone VARCHAR; | 2023-04-13 17:44:55.113388+00
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
(6 rows)
```

Example di creare un lucchetto in modo esplicito

In questo esempio, creiamo un lucchetto in modo esplicito, quindi utilizziamo la `limitless_locks` vista per vedere i blocchi (alcune colonne vengono omesse).

```
BEGIN;
SET search_path = public;
LOCK TABLE customers IN ACCESS SHARE MODE;
SELECT * FROM rds_aurora.limitless_locks WHERE relname = 'customers';

 subcluster_id | subcluster_type | distributed_session_id | locktype |      datname
   | relnspname |  relname  | virtualtransaction |  pid  |      mode
---------------+-----------------+------------------------+----------
+--------------------+------------+ ----------+--------------------+-------
+-----------------
             1 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600787          | 59564 |
 AccessShareLock
             2 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600405          | 67130 |
 AccessShareLock
             3 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 15/473401          | 27735 |
 AccessShareLock
             4 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473524          | 27734 |
 AccessShareLock
```

```
             5 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/472935          | 27737 |
 AccessShareLock
             6 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473015          | 48660 |
 AccessShareLock
(6 rows)
```

limitless_stat_activity

Questa vista contiene una riga per processo per nodo. Può essere utilizzata per tenere traccia dello stato generale del sistema e dei processi di triage che richiedono molto tempo. Per esempio:

```
postgres=# SELECT
    subcluster_id,
    subcluster_type,
    distributed_session_id,
    distributed_session_state,
    datname,
    distributed_query_id,
    is_sso_query
FROM
    rds_aurora.limitless_stat_activity
WHERE
    distributed_session_id in ('D2470C97E3D07E06', '5A3CD7B8E5FD13FF')
    order by  distributed_session_id;

 subcluster_id | subcluster_type | distributed_session_id | distributed_session_state |
      datname       | distributed_query_id | is_sso_query
---------------+-----------------+------------------------+---------------------------
+--------------------+----------------------+-------------
 2             | router          | 5A3CD7B8E5FD13FF       | coordinator               |
 postgres_limitless |                      | f
 3             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 4             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 2             | router          | D2470C97E3D07E06       | coordinator               |
 postgres_limitless |                      | t
 3             | shard           | D2470C97E3D07E06       | participant               |
 postgres_limitless |  4058400544464210222 |
(5 rows)
```

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`
- `distributed_session_id`(testo) — L'ID della sessione distribuita a cui appartiene questo processo.
- `distributed_session_state`(testo) — Se si tratta di un coordinatore, di un partecipante o di un standalone/nondistributed processo (mostrato come`NULL`).
- `datname`(testo) — Il database a cui è connesso questo processo.
- `distributed_query_id`(bigint) — L'ID della query principale proveniente dal nodo coordinatore. Questa colonna indica `NULL` se è la query principale. Il nodo coordinatore invia l'ID della query distribuita ai nodi partecipanti. Pertanto, per i nodi partecipanti, i valori per l'ID della query distribuito e l'ID della query sono diversi.
- `is_sso_query`(testo) — Questo ci consente di sapere se la query è ottimizzata per un singolo shard o meno.

Il resto delle colonne è uguale a quello in`pg_stat_activity`.

limitless_stat_all_indexes

Questa vista contiene le statistiche di utilizzo degli indici nel gruppo di shard DB. Per esempio:

```
postgres_limitless=> SELECT schemaname, relname, indexrelname, idx_scan
  FROM rds_aurora.limitless_stat_all_indexes
  WHERE relname LIKE 'orders_ts%' ORDER BY indexrelname LIMIT 10;

 schemaname |    relname     |    indexrelname     | idx_scan
------------+----------------+---------------------+----------
 ec_sample  | orders_ts00001 | orders_ts00001_pkey |   196801
 ec_sample  | orders_ts00002 | orders_ts00002_pkey |   196703
 ec_sample  | orders_ts00003 | orders_ts00003_pkey |   196376
 ec_sample  | orders_ts00004 | orders_ts00004_pkey |   197966
 ec_sample  | orders_ts00005 | orders_ts00005_pkey |   195301
 ec_sample  | orders_ts00006 | orders_ts00006_pkey |   195673
 ec_sample  | orders_ts00007 | orders_ts00007_pkey |   194475
 ec_sample  | orders_ts00008 | orders_ts00008_pkey |   191694
 ec_sample  | orders_ts00009 | orders_ts00009_pkey |   193744
 ec_sample  | orders_ts00010 | orders_ts00010_pkey |   195421
(10 rows)
```

limitless_stat_all_tables

Questa vista contiene statistiche su tutte le tabelle nel database corrente nel gruppo di shard DB. Ciò è utile per tenere traccia delle operazioni di vacuo e delle operazioni del linguaggio DML (Data Manipulation Language). Per esempio:

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, relname,
 n_ins_since_vacuum, n_tup_ins, last_vacuum
  FROM rds_aurora.limitless_stat_all_tables
  WHERE relname LIKE 'orders_ts%' ORDER BY relname LIMIT 10;

 subcluster_id | subcluster_type |    relname     | n_ins_since_vacuum | n_tup_ins |
 last_vacuum
---------------+-----------------+----------------+--------------------+-----------
+-------------
 5             | shard           | orders_ts00001 |              34779 |    196083 |
 5             | shard           | orders_ts00002 |              34632 |    194721 |
 5             | shard           | orders_ts00003 |              34950 |    195965 |
 5             | shard           | orders_ts00004 |              34745 |    197283 |
 5             | shard           | orders_ts00005 |              34879 |    195754 |
 5             | shard           | orders_ts00006 |              34340 |    194605 |
 5             | shard           | orders_ts00007 |              33779 |    192203 |
 5             | shard           | orders_ts00008 |              33826 |    191293 |
 5             | shard           | orders_ts00009 |              34660 |    194117 |
 5             | shard           | orders_ts00010 |              34569 |    195560 |
(10 rows)
```

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`
- `relname`(name) — Il nome della tabella.

Il resto delle colonne è uguale a quello in`pg_stat_all_tables`.

limitless_stat_database

Questa vista contiene statistiche su tutti i database del gruppo di shard DB. Restituisce una riga per database per nodo. Per esempio:

```
postgres_limitless=> SELECT
    subcluster_id,
```

```
    subcluster_type,
    datname,
    blks_read,
    blks_hit
FROM
    rds_aurora.limitless_stat_database
WHERE
    datname='postgres_limitless';
 subcluster_id | subcluster_type |      datname       | blks_read | blks_hit
---------------+-----------------+--------------------+-----------+----------
             1 | router          | postgres_limitless |       484 | 34371314
             2 | router          | postgres_limitless |       673 | 33859317
             3 | shard           | postgres_limitless |      1299 | 17749550
             4 | shard           | postgres_limitless |      1094 | 17492849
             5 | shard           | postgres_limitless |      1036 | 17485098
             6 | shard           | postgres_limitless |      1040 | 17437257
(6 rows)
```

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`
- `datname`(name) — Il nome del database.

Il resto delle colonne è uguale a quello in`pg_stat_database`.

limitless_stat_progress_vacuum

Questa visualizzazione contiene informazioni sulle operazioni di aspirazione in corso. Per esempio:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;

-[ RECORD 1 ]----------+-----------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | A56D96E2A5C9F426
pid                    | 5270
datname                | postgres
nspname                | public
relname                | customer_ts2
phase                  | vacuuming heap
```

```
heap_blks_total        | 130500
heap_blks_scanned      | 100036
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 11184810
num_dead_tuples        | 0

-[ RECORD 2 ]----------+------------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | 56DF26A89EC23AB5
pid                    | 6854
datname                | postgres
nspname                | public
relname                | sales_ts1
phase                  | vacuuming heap
heap_blks_total        | 43058
heap_blks_scanned      | 24868
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 8569523
num_dead_tuples        | 0
```

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` o. `shard`
- `distributed_session_id`(testo) — L'identificatore della sessione che ha avviato l'operazione di aspirazione.
- `datname`(nome) — Il database in cui viene eseguita l'aspirazione.
- `nspname`(name) — Il nome dello schema della tabella che viene evacuata. È `null` se la tabella da cancellare non si trova nello stesso database a cui è connesso l'utente.
- `relname`(nome) — Il nome della tabella che viene sottovuotata. È `null` se la tabella da cancellare non si trova nello stesso database a cui è connesso l'utente.

Le altre colonne sono le stesse di. `pg_stat_progress_vacuum`

limitless_stat_statements

Questa visualizzazione fornisce un mezzo per tenere traccia della pianificazione e dell'esecuzione delle statistiche di tutte le istruzioni SQL eseguite su tutti i nodi.

> **Note**
>
> È necessario installare l'estensione pg_stat_statements per utilizzare la vista. `limitless_stat_statements`
>
> ```
> -- CREATE EXTENSION must be run by a superuser
> CREATE EXTENSION pg_stat_statements;
>
> -- Verify that the extension is created on all nodes in the DB shard group
> SELECT distinct node_id
>     FROM rds_aurora.limitless_stat_statements
>     LIMIT 10;
> ```

L'esempio seguente mostra l'uso della vista. `limitless_stat_statements`

```
postgres_limitless=> SELECT
 subcluster_id,
 subcluster_type,
 distributedqueryid,
 username,
 dbname,
 sso_calls
FROM
 rds_aurora.limitless_stat_statements;

 subcluster_id | subcluster_type |  distributedqueryid  |             username
       |       dbname       | sso_calls
---------------+-----------------+----------------------
+-------------------------------------+---------------------+-----------
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         1
```

```
 3             | shard           | -7975178695405682176 | postgres
       | postgres_limitless |
[...]
```

I parametri di output sono i seguenti:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.
- `subcluster_type`(testo) — Il tipo di sottocluster a cui appartiene questo processo: `router` for o. `shard`
- `distributedqueryid`(bigint) — L'ID della query principale proveniente dal nodo coordinatore. Questa colonna indica `NULL` se è la query principale. Il nodo coordinatore invia l'ID della query distribuita ai nodi partecipanti. Pertanto, per i nodi partecipanti, i valori per l'ID della query distribuito e l'ID della query sono diversi.
- `username`(nome) — L'utente che ha richiesto l'istruzione.
- `dbname`(name) — Il database in cui è stata eseguita la query.
- `sso_calls`(nome) — Il numero di volte in cui l'istruzione è stata ottimizzata per un singolo shard.

Il resto delle colonne è lo stesso di pg_stat_statements.

limitless_stat_statements_info

Questa vista contiene le statistiche relative alla vista. `limitless_stat_statements` Ogni riga contiene i dati per la vista pg_stat_statements_info di ciascun nodo. La `subcluster_id` colonna identifica ogni nodo.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_statements_info;

 subcluster_id | subcluster_type | dealloc |          stats_reset
---------------+-----------------+---------+-------------------------------
             1 | router          |       0 | 2023-06-30 21:22:09.524781+00
             2 | router          |       0 | 2023-06-30 21:21:40.834111+00
             3 | shard           |       0 | 2023-06-30 21:22:10.709942+00
             4 | shard           |       0 | 2023-06-30 21:22:10.740179+00
             5 | shard           |       0 | 2023-06-30 21:22:10.774282+00
             6 | shard           |       0 | 2023-06-30 21:22:10.808267+00
(6 rows)
```

Il parametro di output è il seguente:

- `subcluster_id`(testo) — L'ID del sottocluster a cui appartiene questo processo.

Il resto delle colonne è lo stesso di pg_stat_statements_info.

limitless_stat_subclusters

Questa vista contiene statistiche di rete tra router e altri nodi. Contiene una riga per coppia di router e altro nodo, ad esempio:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_subclusters;

 orig_subcluster | orig_instance_az | dest_subcluster | dest_instance_az |
 latency_us |       latest_collection       | failed_requests | received_bytes |
 sent_bytes | same_az_requests | cross_az_requests |     stat_reset_timestamp
-----------------+------------------+-----------------+------------------
+------------+-------------------------------+-----------------+---------------
+------------+------------------+-------------------+------------------------------
 3               | us-west-2b       | 2               | us-west-2a       |
 847 | 2024-10-07 17:25:39.518617+00 |               0 |       35668633 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.239675+00
 3               | us-west-2b       | 4               | us-west-2b       |
 419 | 2024-10-07 17:25:39.546376+00 |               0 |      101190464 |  248795719
 |           883478 |                 0 | 2024-10-05 12:39:55.231218+00
 3               | us-west-2b       | 5               | us-west-2c       |
    1396 | 2024-10-07 17:25:39.52122+00  |               0 |       72864849 |
 172086292 |                0 |           557726 | 2024-10-05 12:39:55.196412+00
 3               | us-west-2b       | 6               | us-west-2c       |
 729 | 2024-10-07 17:25:39.54828+00  |               0 |       35668584 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.247334+00
 3               | us-west-2b       | 7               | us-west-2a       |
    1702 | 2024-10-07 17:25:39.545307+00 |               0 |       71699576 |
 171634844 |                0 |           556278 | 2024-10-05 12:39:52.715168+00
 2               | us-west-2a       | 3               | us-west-2b       |
 868 | 2024-10-07 17:25:40.293927+00 |               0 |       35659611 |   92011872
 |                0 |           302817 | 2024-10-05 12:39:54.420758+00
 2               | us-west-2a       | 4               | us-west-2b       |
 786 | 2024-10-07 17:25:40.296863+00 |               0 |      102437253 |  251838024
 |                0 |           895060 | 2024-10-05 12:39:54.404081+00
 2               | us-west-2a       | 5               | us-west-2c       |
    1232 | 2024-10-07 17:25:40.292021+00 |               0 |       71990027 |
 168828110 |                0 |           545453 | 2024-10-05 12:39:36.769549+00
```

I parametri di output sono i seguenti:

- `orig_subcluster`(testo) — L'ID del router da cui provengono le comunicazioni
- `orig_subcluster_az`(text) — La zona di disponibilità (AZ) del router originatore

- `dest_subcluster`(text) — L'ID del nodo di destinazione
- `dest_subcluster_az`(text) — L'ultima AZ raccolta del nodo di destinazione
- `latency_us`(bigint) — L'ultima latenza di rete raccolta tra i nodi, in microsecondi. Il valore è `0` se il nodo non è raggiungibile.
- `latest_collection`(timestamp) — Il timestamp dell'ultima raccolta di AZ e la latenza per il nodo di destinazione
- `failed_requests`(bigint) — Il conteggio cumulativo delle richieste interne non riuscite
- `received_bytes`(bigint) — Il numero cumulativo stimato di byte ricevuti da questo nodo
- `sent_bytes`(bigint) — Il numero cumulativo stimato di byte inviati a questo nodo
- `same_az_requests`(bigint) — Il numero cumulativo di richieste DB interne a questo nodo quando si trova nella stessa AZ del router originatore
- `cross_az_requests`(bigint) — Il numero cumulativo di richieste DB interne a questo nodo quando si trova in una AZ diversa dal router originatore
- `stat_reset_timestamp`(timestamp) — Il timestamp in cui le statistiche cumulative per questa vista sono state reimpostate l'ultima volta

limitless_statio_all_indexes

Questa vista contiene statistiche input/output (I/O) per tutti gli indici del gruppo di shard DB. Per esempio:

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_statio_all_indexes WHERE
 relname like'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      |
 indexrelname             | idx_blks_read | idx_blks_hit
---------------+-----------------+------------+-------------------
+-------------------------------------+ --------------+--------------
             3 | shard           | public     | customers_ts00002 |
 customers_ts00002_customer_name_idx |             1 |            0
             3 | shard           | public     | customers_ts00001 |
 customers_ts00001_customer_name_idx |             1 |            0
             4 | shard           | public     | customers_ts00003 |
 customers_ts00003_customer_name_idx |             1 |            0
             4 | shard           | public     | customers_ts00004 |
 customers_ts00004_customer_name_idx |             1 |            0
             5 | shard           | public     | customers_ts00005 |
 customers_ts00005_customer_name_idx |             1 |            0
```

```
             5 | shard           | public     | customers_ts00006 |
 customers_ts00006_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00007 |
 customers_ts00007_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00008 |
 customers_ts00008_customer_name_idx |             1 |           0
(8 rows)
```

limitless_statio_all_tables

Questa vista contiene statistiche input/output (I/O) per tutte le tabelle del gruppo di shard DB. Per esempio:

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    schemaname,
    relname,
    heap_blks_read,
    heap_blks_hit
FROM
    rds_aurora.limitless_statio_all_tables
WHERE
    relname LIKE 'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      | heap_blks_read |
 heap_blks_hit
---------------+-----------------+------------+-------------------+----------------
+---------------
             3 | shard           | public     | customers_ts00002 |            305 |
      57780
             3 | shard           | public     | customers_ts00001 |            300 |
      56972
             4 | shard           | public     | customers_ts00004 |            302 |
      57291
             4 | shard           | public     | customers_ts00003 |            302 |
      57178
             5 | shard           | public     | customers_ts00006 |            300 |
      56932
             5 | shard           | public     | customers_ts00005 |            302 |
      57386
             6 | shard           | public     | customers_ts00008 |            300 |
      56881
```

```
             6 | shard           | public     | customers_ts00007 |            304 |
      57635
(8 rows)
```

limitless_tables

Questa vista contiene informazioni sulle tabelle in Aurora PostgreSQL Limitless Database.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | placeholder | active       | placeholder |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     | HASH
 (order_id)
         2 |     18579 | public      | customer    | active       | sharded     | HASH
 (cust_id)
(4 rows)
```

limitless_table_collocations

Questa vista contiene informazioni sulle tabelle suddivise collocate.

Nell'esempio seguente, le `customers` tabelle `orders` and sono collocate e le `users` `followers` tabelle and sono collocate. Le tabelle collocate hanno le stesse caratteristiche. `collocation_id`

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
              2 | public      | orders
              2 | public      | customers
              5 | public      | users
              5 | public      | followers
(4 rows)
```

limitless_table_collocation_distribution

Questa vista mostra la distribuzione delle chiavi per ogni collocazione.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
              2 |             6 | -9223372036854775808 | -4611686018427387904
              2 |             5 | -4611686018427387904 |                    0
              2 |             4 |                    0 |  4611686018427387904
              2 |             3 |  4611686018427387904 |  9223372036854775807
              5 |             6 | -9223372036854775808 | -4611686018427387904
              5 |             5 | -4611686018427387904 |                    0
              5 |             4 |                    0 |  4611686018427387904
              5 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Attendi gli eventi per Aurora SQL Postgre Limitless Database

Un evento di attesa in Aurora Postgre SQL indica una risorsa per la quale una sessione è in attesa, ad esempio) e si blocca. input/output (I/O Gli eventi di attesa sono utili per scoprire perché le sessioni sono in attesa di risorse e identificare i punti deboli. Per ulteriori informazioni, consulta Eventi di attesa Aurora PostgreSQL.

Aurora Postgre SQL Limitless Database ha i propri eventi di attesa correlati a router e shard. Molti di questi sono per router che attendono che gli shard completino le attività. Gli eventi di attesa negli shard contengono dettagli sulle attività che vengono eseguite.

Argomenti

- Interrogazione degli eventi di attesa
- Eventi di attesa illimitati del database

Interrogazione degli eventi di attesa

È possibile utilizzare la vista limitless_stat_activity per interrogare gli eventi di attesa, come mostrato nell'esempio seguente.

```
SELECT wait_event FROM rds_aurora.limitless_stat_activity WHERE
 wait_event_type='AuroraLimitless';

      wait_event
----------------------
 RemoteStatementSetup
 RemoteStatementSetup
(2 rows)
```

È inoltre possibile utilizzare la `aurora_stat_system_waits` funzione per elencare il numero di attese e il tempo totale impiegato per ogni evento di attesa, come illustrato nell'esempio seguente.

```
postgres_limitless=> SELECT type_name,event_name,waits,wait_time
    FROM aurora_stat_system_waits()
    NATURAL JOIN aurora_stat_wait_event()
    NATURAL JOIN aurora_stat_wait_type()
    WHERE type_name='AuroraLimitless'
    ORDER BY wait_time DESC;
```

```
    type_name    |       event_name          |  waits  |  wait_time
-----------------+---------------------------+---------+-------------
 AuroraLimitless | RemoteStatementSetup      |    7518 | 75236507897
 AuroraLimitless | RemoteStatementExecution  |      40 |      132986
 AuroraLimitless | Connect                   |       5 |        1453
(3 rows)
```

Eventi di attesa illimitati del database

I seguenti eventi di attesa si applicano al database Aurora PostgreSQL Limitless. È possibile monitorare questi eventi di attesa per identificare i colli di bottiglia nell'elaborazione del database Aurora PostgreSQL Limitless.

Argomenti

- TwophaseFilePoolWrite IO: evento di attesa
- IO: evento di TwophaseFilePoolRead attesa
- AuroraLimitless:Evento Connect Wait
- AuroraLimitless: AsyncConnect evento di attesa
- AuroraLimitless: evento RemoteStatementSetup di attesa
- AuroraLimitlessOttimizza le DDLExecution query per ridurre il numero di operazioni di scansione o aumentare l'idoneità per l'ottimizzazione a shard singolo. ----SEP----:Evento di attesa remota
- AuroraLimitless: evento RemoteStatementExecution di attesa
- AuroraLimitless: evento di FetchRemoteResults attesa
- AuroraLimitless: evento di AsyncGetInitialResponse attesa
- AuroraLimitless: evento di attesa AsyncGetNextResponse
- AuroraLimitless: evento di attesa AbortedCommandCleanup
- AuroraLimitless: evento DistributedCommitPrepare di attesa
- AuroraLimitless: evento di DistributedCommit attesa
- AuroraLimitless: evento di DistributedCommitPrepareThrottle attesa
- AuroraLimitless: evento PreparedTransactionResolution di attesa
- AuroraLimitless: evento di SendPreparedTransactionOutcome attesa
- AuroraLimitless: evento di CommitClockBarrier attesa
- AuroraLimitless: evento di SnapshotClockBarrier attesa

- AuroraLimitless: evento ReaderSnapshotClockBarrier di attesa
- AuroraLimitless: evento di attesa GatherDistributedDeadlockGraph
- AuroraLimitless: evento DistributedDeadlockDetection di attesa
- AuroraLimitless: evento DistributedDeadlockAbort di attesa
- AuroraLimitless: evento GatherRemoteStats di attesa
- AuroraLimitless: evento di attesa GlobalSequenceRefresh
- AuroraLimitless: evento GlobalVacuumTimeExchange di attesa
- AuroraLimitless: DistributedTransactionMonitorGather evento di attesa
- AuroraLimitlessActivity: evento di attesa AdminTaskSchedulerMain
- AuroraLimitlessActivity: evento di AdminTaskExecutorMain attesa
- AuroraLimitlessActivity: evento di AdminTaskMonitorMain attesa
- AuroraLimitlessActivity: evento di DatabaseCleanupMonitorMain attesa
- AuroraLimitlessActivity: evento di TopologyCleanupMonitorMain attesa
- AuroraLimitlessActivity: evento ToplogyChangeMonitorMain di attesa
- AuroraLimitlessActivity: evento di DistributedTransactionMonitorMain attesa
- AuroraLimitlessActivity: evento di GlobalVacuumMonitorMain attesa

TwophaseFilePoolWrite IO: evento di attesa

In attesa della scrittura di un file di stato a due fasi all'interno del pool di file di stato a due fasi. Questo è un evento specifico di Aurora.

Cause

I processi che eseguono un `PREPARED TRANSACTION` comando, inclusi i partecipanti a una transazione distribuita con Limitless Database, devono mantenere lo stato della transazione in un file a due fasi. Aurora utilizza un pool di file per migliorare le prestazioni di questa operazione.

Azione

Si tratta di un'operazione di I/O di scrittura sincrona e pertanto un'elevata latenza in questo evento ha cause simili `IO:XactSync` e può essere esaminata allo stesso modo. Se si utilizza Limitless Database, potrebbe essere necessario ridurre il numero di transazioni distribuite eseguite.

IO: evento di TwophaseFilePoolRead attesa

In attesa della lettura di un file di stato a due fasi all'interno del pool di file di stato a due fasi.

Cause

I processi che eseguono un `COMMIT PREPARED` comando su una transazione preparata in precedenza, inclusi i partecipanti a una transazione distribuita da Limitless Database, potrebbero dover leggere lo stato della transazione precedentemente persistente da un file a due fasi. Aurora utilizza un pool di file per migliorare le prestazioni di questa operazione.

Azione

Si tratta di un'operazione di I/O di lettura. Pertanto, un'elevata latenza in questo evento ha cause simili `IO:DataFileRead` e può essere studiata allo stesso modo. Se si utilizza Limitless Database, potrebbe essere necessario ridurre il numero di transazioni distribuite eseguite.

AuroraLimitless:Evento Connect Wait

Il processo è in attesa che venga stabilita una connessione a un altro nodo del cluster.

Cause

Vengono stabilite connessioni tra processi e nodi remoti per eseguire query, transazioni distribuite ed eseguire DDLs.

Azione

Riduci il numero di connessioni simultanee al cluster o ottimizza l'uso di query cross-shard.

AuroraLimitless: AsyncConnect evento di attesa

Questo evento è simile a`Connect`, ma rappresenta un processo in attesa che vengano stabilite connessioni parallele a un insieme di nodi.

Cause

La creazione di connessioni parallele viene eseguita più comunemente durante l'esecuzione di istruzioni DDL.

Azione

Riduci il numero di istruzioni DDL o combinane più istruzioni DDLs nella stessa sessione per migliorare il riutilizzo della connessione.

AuroraLimitless: evento RemoteStatementSetup di attesa

Il processo è in attesa della configurazione dell'esecuzione remota delle query, ad esempio l'apertura, la chiusura del cursore o la creazione di istruzioni preparate.

Cause

Questo evento di attesa aumenta con il numero di scansioni su tabelle frammentate in cui l'istruzione non può essere ottimizzata per un singolo shard.

Azione

Ottimizza le query per ridurre il numero di operazioni di scansione o aumentare l'idoneità all'ottimizzazione a shard singolo.

AuroraLimitlessOttimizza le DDLExecution query per ridurre il numero di operazioni di scansione o aumentare l'idoneità per l'ottimizzazione a shard singolo. ----SEP----:Evento di attesa remota

Il processo è in attesa del completamento di un comando DDL remoto.

Cause

Quando si esegue un comando DDL su un gruppo di shard DB, è necessario distribuirlo ad altri router e nodi shard prima di confermare l'operazione. Alcune operazioni DDL possono essere eseguite a lungo, poiché i dati devono essere adattati alle modifiche dello schema.

Azione

Identifica i comandi DDL a esecuzione prolungata in modo da poterli ottimizzare.

AuroraLimitless: evento RemoteStatementExecution di attesa

Un processo è in attesa del completamento di un comando remoto.

Cause

Un comando SQL è in esecuzione su un nodo remoto. Questo evento verrà visualizzato frequentemente per le comunicazioni interne, ad esempio per i `auto_analyze` controlli del battito cardiaco.

Azione

Identifica i comandi a esecuzione prolungata utilizzando la vista limitless_stat_statements. In molti casi si tratta di un evento previsto, specialmente per i lavoratori in background o i processi interni, e non è necessaria alcuna azione.

AuroraLimitless: evento di FetchRemoteResults attesa

Un processo è in attesa di recuperare le righe da un nodo remoto.

Cause

Questo evento di attesa può aumentare quando si recupera un numero elevato di righe da una tabella remota, ad esempio una tabella frammentata o di riferimento.

Azione

Identifica le `SELECT` query non ottimizzate utilizzando la vista. `limitless_stat_statements` Ottimizza le query per recuperare solo i dati necessari. Puoi anche ottimizzare il parametro. `rds_aurora.limitless_maximum_adaptive_fetch_size`

AuroraLimitless: evento di AsyncGetInitialResponse attesa

Il processo è in attesa di una risposta iniziale quando viene utilizzata la modalità pipeline nell'esecuzione delle query.

Cause

Ciò si verifica in genere durante l'esecuzione di router to shard per le query con posizionamento di dati a frammento singolo ed è una parte prevista della normale esecuzione.

Azione

Non è richiesta alcuna operazione aggiuntiva.

AuroraLimitless: evento di attesa AsyncGetNextResponse

Il processo è in attesa di ulteriori risposte quando viene utilizzata la modalità pipeline nell'esecuzione delle query.

Cause

Ciò si verifica in genere durante l'esecuzione di router to shard per le query con posizionamento di dati a frammento singolo ed è una parte prevista della normale esecuzione.

Azione

Non è richiesta alcuna operazione aggiuntiva.

AuroraLimitless: evento di attesa AbortedCommandCleanup

Il processo è in attesa del risultato di una query di pulizia remota. Le query di pulizia vengono inviate ai nodi shard per riportarli allo stato appropriato al termine di una transazione distribuita.

Cause

La pulizia delle transazioni viene eseguita quando una transazione viene interrotta o perché è stato rilevato un errore o perché un utente ha emesso un comando ABORT esplicito o ha annullato la query in esecuzione.

Azione

Indagate sulla causa dell'annullamento della transazione.

AuroraLimitless: evento DistributedCommitPrepare di attesa

Il processo sta eseguendo una transazione distribuita ed è in attesa che tutti i partecipanti confermino il comando prepare.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito. Una lunga attesa `DistributedCommitPrepare` potrebbe essere causata da lunghe attese nell'`IO:TwophaseFilePoolWrite`evento sui nodi partecipanti.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizza `IO:TwophaseFilePoolWrite` gli eventi in altri nodi del cluster.

AuroraLimitless: evento di DistributedCommit attesa

Il processo sta commettendo una transazione distribuita ed è in attesa che il partecipante principale confermi il comando commit.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito. Una lunga attesa `DistributedCommit` potrebbe essere causata da lunghe attese nell'`IO:XactSync`evento da parte del partecipante principale.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizza `IO:XactSync` gli eventi in altri nodi del cluster.

AuroraLimitless: evento di DistributedCommitPrepareThrottle attesa

Il processo sta tentando di preparare una transazione distribuita e viene rallentato a causa delle transazioni preparate esistenti.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito. I partecipanti a queste transazioni devono eseguire un'operazione di preparazione come parte del protocollo di commit. Aurora limita il numero di preparazioni simultanee e, se questo limite viene superato, il processo attenderà l'evento. `DistributedCommitPrepareThrottle`

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Esamina `IO:TwophaseFilePoolWrite` gli eventi, poiché il prolungamento del tempo trascorso in tali eventi potrebbe causare l'accumulo di transazioni preparate esistenti, con conseguente rallentamento dei nuovi tentativi di preparazione.

AuroraLimitless: evento PreparedTransactionResolution di attesa

Il processo ha rilevato una tupla modificata da una transazione distribuita che si trova nello stato preparato. Il processo deve determinare se la transazione distribuita diventerà visibile nella relativa istantanea.

Cause

Le transazioni che modificano più nodi devono eseguire un commit distribuito che include una fase di preparazione. Un numero elevato di transazioni distribuite o una maggiore latenza sui commit distribuiti possono causare l'evento di `PreparedTransactionResolution` attesa in altri processi.

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizzate gli eventi relativi al commit distribuito, in quanto l'aumento del tempo trascorso in tali eventi potrebbe aumentare la latenza nel percorso di commit delle transazioni distribuite. Potresti anche voler esaminare i carichi di rete e della CPU.

AuroraLimitless: evento di SendPreparedTransactionOutcome attesa

Il processo è in esecuzione su un nodo che coordina una transazione distribuita e un altro processo ha richiesto informazioni sullo stato della transazione, oppure il processo ha eseguito una transazione distribuita e sta inviando il risultato ai partecipanti.

Cause

I processi che riscontrano l'evento di `PreparedTransactionResolution` attesa interrogheranno il coordinatore della transazione. La risposta sul coordinatore della transazione verrà ricevuta. SendPreparedTransactionOutcome

Azione

Riduci il numero di transazioni che modificano i dati su più nodi. Analizza gli eventi `IO:TwophaseFilePoolWrite` e `IO:TwophaseFilePoolRead` gli eventi relativi al commit distribuito, poiché tali eventi potrebbero aumentare la latenza nel percorso di commit delle transazioni distribuite. Potresti anche voler esaminare i carichi di rete e della CPU.

AuroraLimitless: evento di CommitClockBarrier attesa

Il processo sta eseguendo il commit di una transazione e deve attendere per garantire che il tempo di commit assegnato sia passato per tutti i nodi del cluster.

Cause

La staturazione della CPU o della rete potrebbe causare un aumento della deriva del clock, con conseguente tempo impiegato in questo evento di attesa.

Azione

Analizza la saturazione della CPU o della rete nel cluster.

AuroraLimitless: evento di SnapshotClockBarrier attesa

Il processo ha ricevuto un'istantanea dell'ora da un altro nodo con un orologio del futuro ed è in attesa che il proprio orologio raggiunga quell'ora.

Cause

Ciò si verifica in genere dopo che il processo ha ricevuto i risultati da una funzione che è stata trasferita su uno shard e si è verificata una variazione dell'orologio tra i nodi. La staturazione della CPU o della rete potrebbe causare un aumento della deriva del clock, con conseguente tempo impiegato in questo evento di attesa.

Azione

Analizza la staturazione della CPU o della rete nel cluster.

AuroraLimitless: evento ReaderSnapshotClockBarrier di attesa

Questo evento si verifica sui nodi di lettura. Il processo è in attesa che il nodo di lettura riproduca il flusso di scrittura in modo che tutte le scritture avvenute prima dell'ora dell'istantanea del processo siano state applicate.

Cause

Un aumento del ritardo della replica di Aurora può causare un aumento del tempo di attesa in questo evento.

Azione

Indaga sul ritardo della replica di Aurora.

AuroraLimitless: evento di attesa GatherDistributedDeadlockGraph

Il processo sta comunicando con altri nodi per raccogliere grafici di blocco come parte del rilevamento distribuito dei deadlock.

Cause

Quando un processo è in attesa di un blocco, eseguirà un controllo di deadlock distribuito dopo aver atteso più di. `rds_aurora.limitless_distributed_deadlock_timeout`

Azione

Analizzate le cause della contrapposizione dei blocchi nella vostra applicazione e valutate la possibilità di ottimizzarla. `rds_aurora.limitless_distributed_deadlock_timeout`

AuroraLimitless: evento DistributedDeadlockDetection di attesa

Il processo sta comunicando con altri nodi per rilevare un deadlock distribuito.

Cause

Quando un processo è in attesa di un blocco, eseguirà un controllo di deadlock distribuito dopo aver atteso più di. `rds_aurora.limitless_distributed_deadlock_timeout`

Azione

Analizzate le cause della contrapposizione dei blocchi nella vostra applicazione e valutate la possibilità di ottimizzarla. `rds_aurora.limitless_distributed_deadlock_timeout`

AuroraLimitless: evento DistributedDeadlockAbort di attesa

Il processo sta comunicando con un altro nodo per interrompere una sessione scelta come vittima in un deadlock distribuito.

Cause

I modelli di applicazione danno luogo a situazioni di stallo distribuite.

Azione

Analizzate i modelli di applicazione che determinano situazioni di stallo distribuite.

AuroraLimitless: evento GatherRemoteStats di attesa

Il processo sta raccogliendo statistiche da altri nodi del cluster.

Cause

Il monitoraggio delle nostre attività, le query e le visualizzazioni, ad esempio`limitless_stat_activity`, recupereranno le statistiche da altri nodi.

Azione

Non è richiesta alcuna operazione aggiuntiva.

AuroraLimitless: evento di attesa GlobalSequenceRefresh

Il processo sta generando un nuovo valore di sequenza e deve richiedere un nuovo blocco dalla sequenza globale.

Cause

In questo caso, un'elevata velocità di generazione del valore di sequenza può causare uno stallo se `rds_aurora.limitless_sequence_chunk_size` è insufficiente.

Azione

Si tratta di un evento normale. Se in questo caso il tempo a disposizione è eccessivo, valuta la possibilità di effettuare la sintonizzazione. `rds_aurora.limitless_sequence_chunk_size` Consulta la documentazione sulle sequenze in Limitless Database.

AuroraLimitless: evento GlobalVacuumTimeExchange di attesa

Il processo consiste nello scambio di dati istantanei per supportare il vuoto.

Cause

I nodi di Limitless Database scambiano i dati temporali più vecchi relativi alle istantanee attive con altri nodi per calcolare l'ora limite corretta per l'esecuzione sotto vuoto.

Azione

Non è richiesta alcuna operazione aggiuntiva.

AuroraLimitless: DistributedTransactionMonitorGather evento di attesa

Il processo sta raccogliendo i metadati delle transazioni da altri nodi per supportare la pulizia distribuita delle transazioni.

Cause

I nodi di Limitless Database scambiano i metadati delle transazioni con altri nodi per determinare quando è possibile eliminare lo stato della transazione distribuita.

Azione

Non è richiesta alcuna operazione aggiuntiva.

AuroraLimitlessActivity: evento di attesa AdminTaskSchedulerMain

In attesa nel ciclo principale del processo di pianificazione delle attività.

AuroraLimitlessActivity: evento di AdminTaskExecutorMain attesa

In attesa nel ciclo principale del processo di esecuzione delle attività.

AuroraLimitlessActivity: evento di AdminTaskMonitorMain attesa

In attesa nel ciclo principale del processo di monitoraggio delle attività.

AuroraLimitlessActivity: evento di DatabaseCleanupMonitorMain attesa

In attesa nel ciclo principale del processo di monitoraggio della pulizia del database.

AuroraLimitlessActivity: evento di TopologyCleanupMonitorMain attesa

In attesa del ciclo principale del processo di monitoraggio della pulizia della topologia.

AuroraLimitlessActivity: evento ToplogyChangeMonitorMain di attesa

Attesa nel ciclo principale del processo di monitoraggio della modifica della topologia.

AuroraLimitlessActivity: evento di DistributedTransactionMonitorMain attesa

Attesa nel ciclo principale del processo di monitoraggio distribuito delle transazioni.

AuroraLimitlessActivity: evento di GlobalVacuumMonitorMain attesa

Attesa nel ciclo principale del processo globale di monitoraggio del vuoto.

Backup e ripristino del database Aurora Postgre Limitless SQL

È possibile eseguire il backup e il ripristino di un cluster DB che utilizza Aurora SQL Postgre Limitless Database.

Indice

- Backup di un cluster DB che utilizza SQL Aurora Postgre Limitless Database
  - Creazione di uno snapshot del cluster database
- Ripristino di un cluster DB che utilizza SQL Aurora Postgre Limitless Database
  - Ripristino di un cluster DB da un'istantanea DB
  - Ripristino di un cluster DB mediante ripristino point-in-time
- Le utilità di SQL backup e ripristino di Postgre non sono supportate

Backup di un cluster DB che utilizza SQL Aurora Postgre Limitless Database

Il backup di un cluster DB con Aurora SQL Postgre Limitless Database presenta somiglianze e differenze di funzionalità rispetto al backup di un cluster Aurora DB standard.

- Quando si scatta un'istantanea manuale di un cluster DB di un cluster Aurora DB che utilizza Limitless Database, l'istantanea include i dati del gruppo di shard DB.
- I backup continui includono i dati del gruppo di shard DB.
- Le istantanee giornaliere automatizzate includono i dati del gruppo di shard DB.
- È supportata la copia delle istantanee del cluster DB. Per ulteriori informazioni, consulta Copia di snapshot del cluster DB.
- È supportata la condivisione di istantanee del cluster DB. Per ulteriori informazioni, consulta Condivisione di uno snapshot cluster database.
- Non è possibile utilizzare l'`pg_dumpall`utilità `pg_dump` or per eseguire il backup dei database nel gruppo di shard DB.
- L'acquisizione di istantanee finali durante l'eliminazione dei cluster DB è supportata per Aurora Postgre Limitless Database. SQL
- Il mantenimento dei backup automatici durante l'eliminazione dei cluster DB non è supportato per Aurora Postgre Limitless Database. SQL

Creazione di uno snapshot del cluster database

È possibile creare uno snapshot del cluster Aurora Postgre SQL Limitless Database DB nello stesso modo di un cluster Aurora DB standard, come illustrato nell'esempio seguente: AWS CLI

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier my-db-cluster \
    --db-cluster-snapshot-identifier my-db-cluster-snapshot
```

Per ulteriori informazioni sul backup dei cluster DB, vedere. Panoramica di backup e ripristino di un cluster di database Aurora

Ripristino di un cluster DB che utilizza SQL Aurora Postgre Limitless Database

Il ripristino di un cluster DB con Aurora SQL Postgre Limitless Database presenta somiglianze e differenze di funzionalità rispetto al ripristino di un cluster Aurora DB standard.

- È possibile ripristinare un cluster Limitless Database DB solo da un cluster DB di origine che utilizza una versione del motore DB compatibile con Limitless Database, ad esempio. `16.4-limitless`
- Quando si ripristina un cluster DB da un'istantanea manuale di un cluster DB che utilizza Limitless Database, viene ripristinato l'intero storage del cluster DB. Ciò include l'archiviazione del gruppo di shard DB.

  È necessario creare un gruppo di shard DB per accedere allo storage del database Limitless.
- È possibile ripristinare un cluster DB utilizzando point-in-time recovery (PITR) in qualsiasi momento entro il periodo di conservazione. Il cluster DB ripristinato include l'archiviazione del gruppo di shard DB.

  È necessario creare un gruppo di shard DB per accedere allo storage del database Limitless.
- PITRnon è supportato per i cluster Aurora Postgre SQL Limitless Database DB eliminati.
- Quando si ripristina un cluster DB da uno snapshot giornaliero automatico, viene ripristinato anche lo storage per il gruppo di shard DB.
- Quando si ripristina un cluster Aurora Postgre SQL Limitless Database DB, è necessario abilitare Enhanced Monitoring and Performance Insights. Assicurati di includere l'ID KMS chiave Performance Insights.

Dopo aver ripristinato un cluster Aurora Postgre SQL Limitless Database DB, assicurati di verificarne la funzionalità eseguendo le query su di esso.

Ripristino di un cluster DB da un'istantanea DB

AWS CLI Gli esempi seguenti mostrano come ripristinare un cluster Aurora Postgre SQL Limitless Database DB da un'istantanea del cluster DB.

È necessario utilizzare la versione del motore DB. `16.4-limitless`

Per ripristinare un cluster Limitless Database DB da un'istantanea del cluster DB

1. Ripristina il cluster DB:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-db-cluster \
    --snapshot-identifier my-db-cluster-snapshot \
    --engine aurora-postgresql \
    --engine-version 16.4-limitless \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Crea il gruppo di shard DB:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Per ulteriori informazioni, consulta Aggiungere un gruppo di shard DB a un cluster Aurora SQL Postgre Limitless Database DB esistente.

Per ulteriori informazioni sul ripristino dei cluster Aurora DB dalle istantanee dei cluster DB, vedere. Ripristino da uno snapshot cluster database

Ripristino di un cluster DB mediante ripristino point-in-time

AWS CLI Gli esempi seguenti mostrano come ripristinare un cluster Aurora Postgre SQL Limitless Database DB utilizzando recovery (). point-in-time PITR

Per ripristinare un cluster Limitless Database DB utilizzando PITR

1. Ripristina il cluster DB:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-db-cluster \
    --db-cluster-identifier my-new-db-cluster \
    --use-latest-restorable-time \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Crea il gruppo di shard DB:

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Per ulteriori informazioni, consulta Aggiungere un gruppo di shard DB a un cluster Aurora SQL Postgre Limitless Database DB esistente.

Per ulteriori informazioni su PITR, consulta Ripristino di un cluster di database a un determinato momento.

Le utilità di SQL backup e ripristino di Postgre non sono supportate

Le seguenti SQL utilità Postgre non sono supportate né per il cluster DB primario né per il gruppo di shard DB:

- `pg_dump`
- `pg_dumpall`
- `pg_restore`

Sebbene sia possibile utilizzarle con file binari open source o metodi alternativi, così facendo si potrebbero ottenere risultati incoerenti.

Aggiornamento del database Amazon Aurora PostgreSQL Limitless

Quanto segue si applica all'aggiornamento del database Aurora PostgreSQL Limitless:

- Sono supportati aggiornamenti di versione minori.
- È supportata l'applicazione di patch al database Aurora PostgreSQL Limitless. Le patch vengono visualizzate come operazioni di manutenzione in sospeso da applicare durante la finestra di manutenzione.

Aggiornamento di cluster DB che utilizzano Amazon Aurora PostgreSQL Limitless Database

È possibile aggiornare un cluster DB modificandolo e scegliendo una nuova versione del motore DB. È possibile utilizzare AWS Management Console o il AWS CLI.

Console

Per aggiornare il cluster Limitless Database DB

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Passare alla pagina Databases (Database).
3. Seleziona il cluster Limitless Database DB che desideri aggiornare.
4. Scegli Modifica.

   Viene visualizzata la pagina Modifica cluster DB.
5. Per la versione del motore DB, scegli la nuova versione del motore DB, ad esempio Aurora PostgreSQL with Limitless Database (compatibile con PostgreSQL 16.6).
6. Scegli Continua.
7. Nella pagina di riepilogo, seleziona se applicare le modifiche immediatamente o durante la successiva finestra di manutenzione pianificata, quindi scegli Modifica cluster.

CLI

Utilizza il comando modify-db-cluster AWS CLI come visualizzato nell'esempio seguente.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-sv2-cluster \
    --engine-version 16.6-limitless \
    --apply-immediately
```

Riferimento al database Aurora SQL Postgre Limitless

Forniamo i seguenti argomenti di riferimento per Aurora SQL Postgre Limitless Database.

Argomenti

- Comandi Data Definition Language (DDL) SQL supportati e non supportati
- Limitazioni DDL e altre informazioni per Aurora PostgreSQL Limitless Database
- Comandi SQL per l'elaborazione delle query e del linguaggio di manipolazione dei dati (DML) supportati e non supportati
- DMLlimitazioni e altre informazioni per Aurora SQL Postgre Limitless Database
- Variabili nel database Aurora SQL Postgre Limitless
- Parametri del cluster DB nel database Aurora SQL Postgre Limitless

Comandi Data Definition Language (DDL) SQL supportati e non supportati

La tabella seguente elenca i DDL comandi supportati e non supportati da Aurora Postgre SQL Limitless Database, con riferimenti alle limitazioni o ulteriori informazioni.

Comando	Supportato?	Limitazioni o ulteriori informazioni
ALTER AGGREGATE	No	Non applicabile
ALTER COLLATION	Sì	Nessuno
ALTER CONVERSION	Sì	Nessuno
ALTER DATABASE	No	Non applicabile
ALTER DEFAULT PRIVILEGES	No	Non applicabile
ALTER DOMAIN	No	Non applicabile
ALTER EVENT TRIGGER	No	Non applicabile
ALTER EXTENSION	Sì	Estensioni

Comando	Supportato?	Limitazioni o ulteriori informazioni
ALTER FOREIGN DATA WRAPPER	No	Non applicabile
ALTER FOREIGN TABLE	No	Non applicabile
ALTER FUNCTION	Sì	Funzioni
ALTER GROUP	Sì	Nessuno
ALTER INDEX	Sì	Nessuno
ALTER LANGUAGE	No	Non applicabile
ALTER LARGE OBJECT	No	Non applicabile
ALTER MATERIALIZED VIEW	No	Non applicabile
ALTER OPERATOR	Sì	Nessuno
ALTER OPERATOR CLASS	Sì	Nessuno
ALTER OPERATOR FAMILY	Sì	Nessuno
ALTER POLICY	No	Non applicabile
ALTER PROCEDURE	Sì	Nessuno
ALTER PUBLICATION	No	Non applicabile
ALTER ROLE	Sì	Nessuno
ALTER ROUTINE	No	Non applicabile
ALTER RULE	No	Non applicabile
ALTER SCHEMA	Sì	Nessuno
ALTER SEQUENCE	Sì	Sequenze

Comando	Supportato?	Limitazioni o ulteriori informazioni
ALTER SERVER	No	Non applicabile
ALTER STATISTICS	No	Non applicabile
ALTER SUBSCRIPTION	No	Non applicabile
ALTER SYSTEM	No	Non applicabile
ALTER TABLE	Sì	ALTER TABLE
ALTER TABLESPACE	No	Non applicabile
ALTER TEXT SEARCH CONFIGURATION	No	Non applicabile
ALTER TEXT SEARCH DICTIONARY	No	Non applicabile
ALTER TEXT SEARCH PARSER	No	Non applicabile
ALTER TEXT SEARCH TEMPLATE	No	Non applicabile
ALTER TRIGGER	No	Non applicabile
ALTER TYPE	Sì	Nessuno
ALTER USER	Sì	Nessuno
ALTER USER MAPPING	No	Non applicabile
ALTER VIEW	Sì	Nessuno
COMMENT	No	Non applicabile
CREATE ACCESS METHOD	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
CREATE AGGREGATE	No	Non applicabile
CREATE CAST	Sì	Nessuno
CREATE COLLATION	Sì	Nessuno
CREATE CONVERSION	Sì	Nessuno
CREATE DATABASE	Sì	CREATE DATABASE
CREATE DOMAIN	No	Non applicabile
CREATE EVENT TRIGGER	No	Non applicabile
CREATE EXTENSION	Sì	Estensioni
CREATE FOREIGN DATA WRAPPER	No	Non applicabile
CREATE FOREIGN TABLE	No	Non applicabile
CREATE FUNCTION	Sì	Funzioni
CREATE GROUP	Sì	Nessuno
CREATE INDEX	Sì	CREATE INDEX
CREATE LANGUAGE	No	Non applicabile
CREATE MATERIALIZED VIEW	No	Non applicabile
CREATE OPERATOR	Sì	Nessuno
CREATE OPERATOR CLASS	Sì	Nessuno
CREATE OPERATOR FAMILY	Sì	Nessuno

Comando	Supportato?	Limitazioni o ulteriori informazioni
CREATE POLICY	Sì	Nessuno
CREATE PROCEDURE	Sì	Nessuno
CREATE PUBLICATION	No	Non applicabile
CREATE ROLE	Sì	Nessuno
CREATE RULE	No	Non applicabile
CREATE SCHEMA	Sì	CREATE SCHEMA
CREATE SEQUENCE	Sì	Sequenze
CREATE SERVER	No	Non applicabile
CREATE STATISTICS	No	Non applicabile
CREATE SUBSCRIPTION	No	Non applicabile
CREATE TABLE	Sì	CREATE TABLE
CREATETABLEAS	Sì	CREATE TABLE AS
CREATE TABLESPACE	No	Non applicabile
CREATE TEMPORARY TABLE	No	Non applicabile
CREATETEMPORARYTABLECOME	No	Non applicabile
CREATE TEXT SEARCH CONFIGURATION	No	Non applicabile
CREATE TEXT SEARCH DICTIONARY	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
CREATE TEXT SEARCH PARSER	No	Non applicabile
CREATE TEXT SEARCH TEMPLATE	No	Non applicabile
CREATE TRANSFORM	No	Non applicabile
CREATE TRIGGER	No	Non applicabile
CREATE TYPE	Sì	Nessuno
CREATE USER	Sì	Nessuno
CREATE USER MAPPING	No	Non applicabile
CREATE VIEW	Sì	Nessuno
DROP ACCESS METHOD	No	Non applicabile
DROP AGGREGATE	Sì	Nessuno
DROP CAST	Sì	Nessuno
DROP COLLATION	Sì	Nessuno
DROP CONVERSION	Sì	Nessuno
DROP DATABASE	Sì	DROP DATABASE
DROP DOMAIN	No	Non applicabile
DROP EVENT TRIGGER	No	Non applicabile
DROP EXTENSION	Sì	Estensioni
DROP FOREIGN DATA WRAPPER	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
DROP FOREIGN TABLE	No	Non applicabile
DROP FUNCTION	Sì	Funzioni
DROP GROUP	Sì	Nessuno
DROP INDEX	Sì	Nessuno
DROP LANGUAGE	No	Non applicabile
DROP MATERIALIZED VIEW	No	Non applicabile
DROP OPERATOR	Sì	Nessuno
DROP OPERATOR CLASS	Sì	Nessuno
DROP OPERATOR FAMILY	Sì	Nessuno
DROP OWNED	No	Non applicabile
DROP POLICY	No	Non applicabile
DROP PROCEDURE	Sì	Nessuno
DROP PUBLICATION	No	Non applicabile
DROP ROLE	Sì	Nessuno
DROP ROUTINE	No	Non applicabile
DROP RULE	No	Non applicabile
DROP SCHEMA	Sì	Nessuno
DROP SEQUENCE	Sì	Nessuno
DROP SERVER	No	Non applicabile
DROP STATISTICS	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
DROP SUBSCRIPTION	No	Nessuno
DROP TABLE	Sì	Nessuno
DROP TABLESPACE	No	Non applicabile
DROP TEXT SEARCH CONFIGURATION	No	Non applicabile
DROP TEXT SEARCH DICTIONARY	No	Non applicabile
DROP TEXT SEARCH PARSER	No	Non applicabile
DROP TEXT SEARCH TEMPLATE	No	Non applicabile
DROP TRANSFORM	No	Non applicabile
DROP TRIGGER	No	Non applicabile
DROP TYPE	Sì	Nessuno
DROP USER	Sì	Nessuno
DROP USER MAPPING	No	Non applicabile
DROP VIEW	Sì	Nessuno
GRANT	Sì	Nessuno
REASSIGN OWNED	No	Non applicabile
REVOKE	Sì	Nessuno
SECURITY LABEL	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
SELECT INTO	Sì	SELECT INTO
SET	Sì	Nessuno
SET CONSTRAINTS	No	Non applicabile
SET ROLE	Sì	Nessuno
SET SESSION AUTHORIZATION	Sì	Nessuno
SET TRANSACTION	Sì	Nessuno
TRUNCATE	Sì	Nessuno

Limitazioni DDL e altre informazioni per Aurora PostgreSQL Limitless Database

I seguenti argomenti descrivono le limitazioni o forniscono ulteriori informazioni per i comandi SQL DDL in Aurora PostgreSQL Limitless Database.

Argomenti

- ALTER TABLE
- CREATE DATABASE
- CREATE INDEX
- CREATE SCHEMA
- CREATE TABLE
- CREATE TABLE AS
- DROP DATABASE
- SELECT INTO
- Vincoli
- Valori predefiniti
- Estensioni
- Chiavi esterne
- Funzioni
- Sequenze

ALTER TABLE

Il `ALTER TABLE` comando è generalmente supportato in Aurora PostgreSQL Limitless Database. Per ulteriori informazioni, vedere ALTER TABLE nella documentazione di PostgreSQL.

Limitazioni

`ALTER TABLE`presenta le seguenti limitazioni per le opzioni supportate.

Rimozione di una colonna

- Nelle tabelle condivise, non è possibile rimuovere le colonne che fanno parte della chiave shard.
- Nelle tabelle di riferimento, non è possibile rimuovere le colonne chiave primarie.

Modifica del tipo di dati di una colonna

- L'`USING`espressione non è supportata.
- Nelle tabelle condivise, non è possibile modificare il tipo delle colonne che fanno parte della chiave shard.

Aggiungere o rimuovere un vincolo

Per informazioni dettagliate su ciò che non è supportato, consulta. Vincoli

Modifica del valore predefinito di una colonna

I valori predefiniti sono supportati. Per ulteriori informazioni, consulta Valori predefiniti.

Opzioni non supportate

Alcune opzioni non sono supportate perché dipendono da funzionalità non supportate, come i trigger.

Le seguenti opzioni a livello di tabella per `ALTER TABLE` non sono supportate:

- `ALL IN TABLESPACE`
- `ATTACH PARTITION`
- `DETACH PARTITION`
- `ONLY`contrassegnare
- `RENAME CONSTRAINT`

Le seguenti opzioni a livello di colonna per `ALTER TABLE` non sono supportate:

- AGGIUNGI GENERATO
- ELIMINA ESPRESSIONE [SE ESISTE]
- DROP IDENTITY [SE ESISTE]
- RESET
- RESTART
- SET
- IMPOSTA LA COMPRESSIONE
- SET STATISTICS

CREATE DATABASE

In Aurora PostgreSQL Limitless Database, sono supportati solo database illimitati.

Durante `CREATE DATABASE` l'esecuzione, i database che sono stati creati correttamente in uno o più nodi potrebbero non riuscire negli altri nodi, poiché la creazione del database è un'operazione non transazionale. In questo caso, gli oggetti del database che sono stati creati correttamente vengono rimossi automaticamente da tutti i nodi entro un periodo di tempo predeterminato per mantenere la coerenza nel gruppo di shard DB. Durante questo periodo, la ricreazione di un database con lo stesso nome potrebbe generare un errore che indica che il database esiste già.

Sono supportate le seguenti opzioni:

- Collazione:

```
CREATE DATABASE name WITH
    [LOCALE = locale]
    [LC_COLLATE = lc_collate]
    [LC_CTYPE = lc_ctype]
    [ICU_LOCALE = icu_locale]
    [ICU_RULES = icu_rules]
    [LOCALE_PROVIDER = locale_provider]
    [COLLATION_VERSION = collation_version];
```

- `CREATE DATABASE WITH OWNER`:

```
CREATE DATABASE name WITH OWNER = user_name;
```

Le seguenti opzioni non sono supportate:

- `CREATE DATABASE WITH TABLESPACE`:

```
CREATE DATABASE name WITH TABLESPACE = tablespace_name;
```

- `CREATE DATABASE WITH TEMPLATE`:

```
CREATE DATABASE name WITH TEMPLATE = template;
```

CREATE INDEX

`CREATE INDEX CONCURRENTLY`è supportato per le tabelle condivise:

```
CREATE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX`è supportato per tutti i tipi di tabella:

```
CREATE UNIQUE INDEX index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX CONCURRENTLY`non è supportato:

```
CREATE UNIQUE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

Per ulteriori informazioni, consulta UNIQUE. Per informazioni generali sulla creazione di indici, vedere CREATE INDEX nella documentazione di PostgreSQL.

Mostrare gli indici

> Non tutti gli indici sono visibili sui router quando si utilizzano comandi simili. `\d table_name` Utilizzate invece la `pg_catalog.pg_indexes` vista per ottenere gli indici, come illustrato nell'esempio seguente.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items (id int PRIMARY KEY, val int);
CREATE INDEX items_my_index on items (id, val);

postgres_limitless=> SELECT * FROM pg_catalog.pg_indexes WHERE tablename='items';
```

```
 schemaname | tablename |   indexname    | tablespace |
   indexdef
------------+-----------+----------------+------------
+-------------------------------------------------------------------------
 public     | items     | items_my_index |            | CREATE INDEX items_my_index
 ON ONLY public.items USING btree (id, val)
 public     | items     | items_pkey     |            | CREATE UNIQUE INDEX
 items_pkey ON ONLY public.items USING btree (id)
(2 rows)
```

CREATE SCHEMA

`CREATE SCHEMA`con un elemento dello schema non è supportato:

```
CREATE SCHEMA my_schema CREATE TABLE (column_name INT);
```

Ciò genera un errore simile al seguente:

```
ERROR: CREATE SCHEMA with schema elements is not supported
```

CREATE TABLE

Le relazioni nelle `CREATE TABLE` istruzioni non sono supportate, ad esempio:

```
CREATE TABLE orders (orderid int, customerId int, orderDate date) WITH
 (autovacuum_enabled = false);
```

`IDENTITY`le colonne non sono supportate, ad esempio:

```
CREATE TABLE orders (orderid INT GENERATED ALWAYS AS IDENTITY);
```

Aurora PostgreSQL Limitless Database supporta fino a 54 caratteri per i nomi di tabella condivisi.

CREATE TABLE AS

Per creare una tabella utilizzando`CREATE TABLE AS`, è necessario utilizzare la variabile. `rds_aurora.limitless_create_table_mode` Per le tabelle condivise, è necessario utilizzare anche la `rds_aurora.limitless_create_table_shard_key` variabile. Per ulteriori informazioni, consulta Creazione di tabelle illimitate utilizzando variabili.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

CREATE TABLE ctas_table AS SELECT 1 a;

-- "source" is the source table whose columns and data types are used to create the new
 "ctas_table2" table.
CREATE TABLE ctas_table2 AS SELECT a,b FROM source;
```

Non è possibile utilizzarla `CREATE TABLE AS` per creare tabelle di riferimento, poiché richiedono vincoli di chiave primaria. `CREATE TABLE AS`non propaga le chiavi primarie a nuove tabelle.

Per informazioni generali, vedere CREATE TABLE AS nella documentazione di PostgreSQL.

DROP DATABASE

Puoi eliminare i database che hai creato.

Il `DROP DATABASE` comando viene eseguito in modo asincrono in background. Mentre è in esecuzione, riceverai un errore se provi a creare un nuovo database con lo stesso nome.

SELECT INTO

`SELECT INTO`è funzionalmente simile a. CREATE TABLE AS È necessario utilizzare la `rds_aurora.limitless_create_table_mode` variabile. Per le tabelle condivise, è necessario utilizzare anche la `rds_aurora.limitless_create_table_shard_key` variabile. Per ulteriori informazioni, consulta Creazione di tabelle illimitate utilizzando variabili.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

-- "source" is the source table whose columns and data types are used to create the new
 "destination" table.
SELECT * INTO destination FROM source;
```

Attualmente, l'`SELECT INTO`operazione viene eseguita tramite il router, non direttamente tramite gli shard. Pertanto, le prestazioni possono essere lente.

Per informazioni generali, vedere SELECT INTO nella documentazione di PostgreSQL.

Vincoli

Le seguenti limitazioni si applicano ai vincoli in Aurora PostgreSQL Limitless Database.

CHECK

Sono supportati semplici vincoli che coinvolgono operatori di confronto con valori letterali. Le espressioni e i vincoli più complessi che richiedono la convalida delle funzioni non sono supportati, come illustrato negli esempi seguenti.

```
CREATE TABLE my_table (
    id  INT CHECK (id > 0)                                      -- supported
  , val INT CHECK (val > 0 AND val < 1000)                      -- supported
  , tag TEXT CHECK (length(tag) > 0)                            -- not supported:
 throws "Expression inside CHECK constraint is not supported"
  , op_date TIMESTAMP WITH TIME ZONE CHECK (op_date <= now())  -- not supported:
 throws "Expression inside CHECK constraint is not supported"
);
```

È possibile assegnare nomi espliciti ai vincoli, come illustrato nell'esempio seguente.

```
CREATE TABLE my_table (
    id  INT CONSTRAINT positive_id  CHECK (id > 0)
  , val INT CONSTRAINT val_in_range CHECK (val > 0 AND val < 1000)
);
```

È possibile utilizzare la sintassi dei vincoli a livello di tabella con il vincolo, come illustrato nell'esempio seguente`CHECK`.

```
CREATE TABLE my_table (
    id INT CONSTRAINT positive_id  CHECK (id > 0)
  , min_val INT CONSTRAINT min_val_in_range CHECK (min_val > 0 AND min_val < 1000)
  , max_val INT
  , CONSTRAINT max_val_in_range CHECK (max_val > 0 AND max_val < 1000 AND max_val >
 min_val)
);
```

EXCLUDE

I vincoli di esclusione non sono supportati in Aurora PostgreSQL Limitless Database.

FOREIGN KEY

Per ulteriori informazioni, consulta Chiavi esterne.

NON NULLO

`NOT NULL`i vincoli sono supportati senza restrizioni.

PRIMARY KEY

La chiave primaria implica vincoli univoci e pertanto le stesse restrizioni sui vincoli unici si applicano alla chiave primaria. Ciò significa:

- Se una tabella viene convertita in una tabella condivisa, la chiave shard deve essere un sottoinsieme della chiave primaria. Cioè, la chiave primaria contiene tutte le colonne della chiave shard.
- Se una tabella viene convertita in una tabella di riferimento, deve avere una chiave primaria.

Gli esempi seguenti illustrano l'uso delle chiavi primarie.

```
-- Create a standard table.
CREATE TABLE public.my_table (
    item_id INT
  , location_code INT
  , val INT
  , comment text
);

-- Change the table to a sharded table using the 'item_id' and 'location_code'
 columns as shard keys.
CALL rds_aurora.limitless_alter_table_type_sharded('public.my_table',
 ARRAY['item_id', 'location_code']);
```

Sto cercando di aggiungere una chiave primaria che non contenga una chiave shard:

```
-- Add column 'item_id' as the primary key.
-- Invalid because the primary key doesnt include all columns from the shard key:
-- 'location_code' is part of the shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR

-- add column "val" as primary key
-- Invalid because primary key does not include all columns from shard key:
--  item_id and location_code iare part of shard key but not part of the primary key
```

```
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR
```

Cercando di aggiungere una chiave primaria che contenga una chiave shard:

```
-- Add the 'item_id' and 'location_code' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code); -- OK

-- Add the 'item_id', 'location_code', and 'val' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code, val); -- OK
```

Trasforma una tabella standard in una tabella di riferimento.

```
-- Create a standard table.
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);

-- Convert the table to a reference table.
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcode');
```

Per ulteriori informazioni sulla creazione di tabelle frammentate e di riferimento, vedereCreazione di tabelle del database Aurora SQL Postgre Limitless.

UNIQUE

Nelle tabelle suddivise, la chiave univoca deve contenere la chiave shard, ovvero la chiave shard deve essere un sottoinsieme della chiave univoca. Questo viene controllato quando si cambia il tipo di tabella in sharded. Nelle tabelle di riferimento non ci sono restrizioni.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT UNIQUE
);
```

I UNIQUE vincoli a livello di tabella sono supportati, come illustrato nell'esempio seguente.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT
  , CONSTRAINT zipcode_and_email UNIQUE (zipcode, email)
```

```
);
```

L'esempio seguente mostra l'uso congiunto di una chiave primaria e di una chiave univoca. Entrambe le chiavi devono includere la chiave shard.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"p_id"}';
CREATE TABLE t1 (
p_id BIGINT NOT NULL,
c_id BIGINT NOT NULL,
PRIMARY KEY (p_id),
UNIQUE (p_id, c_id)
);
```

Per ulteriori informazioni, consulta Vincoli nella documentazione di PostgreSQL.

Valori predefiniti

Aurora PostgreSQL Limitless Database supporta espressioni con valori predefiniti.

L'esempio seguente mostra l'uso di valori predefiniti.

```
CREATE TABLE t (
    a INT DEFAULT 5,
    b TEXT DEFAULT 'NAN',
    c NUMERIC
);

CALL rds_aurora.limitless_alter_table_type_sharded('t', ARRAY['a']);
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c
---+-----+---
 5 | NAN |
(1 row)
```

Le espressioni sono supportate, come illustrato nell'esempio seguente.

```
CREATE TABLE t1 (a NUMERIC DEFAULT random());
```

L'esempio seguente aggiunge una nuova colonna che è `NOT NULL` e ha un valore predefinito.

```
ALTER TABLE t ADD COLUMN d BOOLEAN NOT NULL DEFAULT FALSE;
SELECT * FROM t;

 a |  b  | c | d
---+-----+---+---
 5 | NAN |   | f
(1 row)
```

L'esempio seguente modifica una colonna esistente con un valore predefinito.

```
ALTER TABLE t ALTER COLUMN c SET DEFAULT 0.0;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
(2 rows)
```

L'esempio seguente elimina un valore predefinito.

```
ALTER TABLE t ALTER COLUMN a DROP DEFAULT;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
   | NAN | 0.0 | f
(3 rows)
```

Per ulteriori informazioni, consulta Valori predefiniti nella documentazione di PostgreSQL.

Estensioni

Le seguenti estensioni PostgreSQL sono supportate in Aurora PostgreSQL Limitless Database:

- `aurora_limitless_fdw`— Questa estensione è preinstallata. Non puoi lasciarlo cadere.
- `aws_s3`— Questa estensione funziona in Aurora PostgreSQL Limitless Database in modo simile a Aurora PostgreSQL.

  Puoi importare dati da un bucket Amazon S3 a un cluster Aurora PostgreSQL Limitless Database DB o esportare dati da un cluster Aurora PostgreSQL Limitless Database DB a un bucket Amazon S3. Per ulteriori informazioni, consultare Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL e Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.
- `btree_gin`
- `citext`
- `ip4r`
- `pg_buffercache`— Questa estensione si comporta in modo diverso in Aurora PostgreSQL Limitless Database rispetto alla community PostgreSQL. Per ulteriori informazioni, consulta differenze pg_buffercache nel database Aurora PostgreSQL Limitless.
- `pg_stat_statements`
- `pg_trgm`
- `pgcrypto`
- `pgstattuple`— Questa estensione si comporta in modo diverso in Aurora PostgreSQL Limitless Database rispetto alla community PostgreSQL. Per ulteriori informazioni, consulta differenze pgstattuple nel database Aurora PostgreSQL Limitless.
- `pgvector`
- `plpgsql`— Questa estensione è preinstallata, ma è possibile eliminarla.
- `PostGIS`— Le transazioni lunghe e le funzioni di gestione delle tabelle non sono supportate. La modifica della tabella di riferimento spaziale non è supportata.
- `unaccent`
- `uuid`

La maggior parte delle estensioni PostgreSQL attualmente non è supportata in Aurora PostgreSQL Limitless Database. Tuttavia, puoi comunque utilizzare l'impostazione di configurazione

shared_preload_libraries (SPL) per caricare le estensioni nel cluster DB primario Aurora PostgreSQL. Vengono inoltre caricati nel database Aurora PostgreSQL Limitless, ma potrebbero non funzionare correttamente.

Ad esempio, è possibile caricare l'`pg_hint_plan`estensione, ma il suo caricamento non garantisce che vengano utilizzati i suggerimenti passati nei commenti delle query.

> **Note**
>
> Non puoi modificare gli oggetti associati all'estensione pg_stat_statements. Per informazioni sull'installazione, vedi limitless_stat_statements. `pg_stat_statements`

È possibile utilizzare le `pg_available_extension_versions` funzioni `pg_available_extensions` and per trovare le estensioni supportate in Aurora PostgreSQL Limitless Database.

Per le estensioni sono supportate le seguenti opzioni: DDLs

CREATE EXTENSION

È possibile creare estensioni, come in PostgreSQL.

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
    [ WITH ] [ SCHEMA schema_name ]
             [ VERSION version ]
             [ CASCADE ]
```

Per ulteriori informazioni, consulta CREATE EXTENSION nella documentazione di PostgreSQL.

ALTER EXTENSION

Sono supportati: DDLs

```
ALTER EXTENSION name UPDATE [ TO new_version ]

ALTER EXTENSION name SET SCHEMA new_schema
```

Per ulteriori informazioni, vedere ALTER EXTENSION nella documentazione di PostgreSQL.

DROP EXTENSION

Puoi eliminare le estensioni, come in PostgreSQL.

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Per ulteriori informazioni, consulta DROP EXTENSION nella documentazione di PostgreSQL.

Quanto segue DDLs non è supportato per le estensioni:

ALTER EXTENSION

Non puoi aggiungere o eliminare oggetti membri dalle estensioni.

```
ALTER EXTENSION name ADD member_object

ALTER EXTENSION name DROP member_object
```

differenze pg_buffercache nel database Aurora PostgreSQL Limitless

In Aurora PostgreSQL Limitless Database, quando installi l'estensione pg_buffercache e utilizzi la `pg_buffercache` vista, ricevi informazioni relative al buffer solo dal nodo a cui sei attualmente connesso: il router. Analogamente, utilizza la funzione o fornisce informazioni solo dal nodo connesso. `pg_buffercache_summary` `pg_buffercache_usage_counts`

È possibile disporre di numerosi nodi e potrebbe essere necessario accedere alle informazioni sul buffer da qualsiasi nodo per diagnosticare i problemi in modo efficace. Pertanto, Limitless Database offre le seguenti funzioni:

- `rds_aurora.limitless_pg_buffercache(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_summary(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_usage_counts(subcluster_id)`

Inserendo l'ID del sottocluster di qualsiasi nodo, che si tratti di un router o di uno shard, è possibile accedere facilmente alle informazioni sul buffer specifiche di quel nodo. Queste funzioni sono disponibili direttamente quando si installa l'`pg_buffercache`estensione nel database illimitato.

> **Note**
>
> Aurora PostgreSQL Limitless Database supporta queste funzioni per la versione 1.4 e successive dell'estensione. `pg_buffercache`

Le colonne mostrate nella `limitless_pg_buffercache` vista differiscono leggermente da quelle nella vista: `pg_buffercache`

- `bufferid`— Rimane invariato da`pg_buffercache`.
- `relname`— Invece di visualizzare il numero di nodo del file come in precedenza`pg_buffercache`, `limitless_pg_buffercache` presenta quello associato, `relname` se disponibile, nel database corrente o nei cataloghi del sistema condiviso, altrimenti. `NULL`
- `parent_relname`— Questa nuova colonna, non presente in`pg_buffercache`, mostra il genitore `relname` se il valore nella `relname` colonna rappresenta una tabella partizionata (nel caso di tabelle condivise). Altrimenti, viene visualizzata. `NULL`
- `spcname`— Invece di visualizzare l'identificatore dell'oggetto del tablespace (OID) come in precedenza`pg_buffercache`, `limitless_pg_buffercache` visualizza il nome del tablespace.
- `datname`— Invece di visualizzare l'OID del database come in precedenza`pg_buffercache`, `limitless_pg_buffercache` visualizza il nome del database.
- `relforknumber`— Rimane invariato da`pg_buffercache`.
- `relblocknumber`— Rimane invariato da. `pg_buffercache`
- `isdirty`— Rimane invariato da. `pg_buffercache`
- `usagecount`— Rimane invariato da. `pg_buffercache`
- `pinning_backends`— Rimane invariato da. `pg_buffercache`

Le colonne `limitless_pg_buffercache_summary` e le `limitless_pg_buffercache_usage_counts` viste sono uguali a quelle delle viste normali `pg_buffercache_summary` e delle `pg_buffercache_usage_counts` viste, rispettivamente.

Utilizzando queste funzioni, è possibile accedere a informazioni dettagliate sulla cache del buffer su tutti i nodi dell'ambiente Limitless Database, facilitando una diagnosi e una gestione più efficaci dei sistemi di database.

differenze pgstattuple nel database Aurora PostgreSQL Limitless

In Aurora PostgreSQL, l'estensione pgstattuple attualmente non supporta tabelle esterne, tabelle partizionate o indici partizionati. Tuttavia, in Aurora PostgreSQL Limitless Database, gli oggetti creati dall'utente sono spesso tra questi tipi non supportati. Sebbene esistano tabelle e indici normali (ad esempio, le tabelle di catalogo e i relativi indici), la maggior parte degli oggetti risiede su nodi esterni, il che li rende oggetti estranei al router.

Riconosciamo l'importanza di questa estensione per ottenere statistiche a livello di tuple, fondamentali per attività come la rimozione del bloat e la raccolta di informazioni diagnostiche. Pertanto, Aurora PostgreSQL Limitless Database fornisce supporto per l'estensione in database illimitati. `pgstattuple`

Il database Aurora PostgreSQL Limitless include le seguenti funzioni nello schema: `rds_aurora`

Funzioni statistiche a livello di tuple

`rds_aurora.limitless_pgstattuple(relation_name)`

- Scopo: Estrarre statistiche a livello di tuple per tabelle standard e relativi indici
- Input: `relation_name` (testo) — Il nome della relazione
- Output: colonne coerenti con quelle restituite dalla `pgstattuple` funzione in Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple(relation_name, subcluster_id)`

- Scopo: Estrarre statistiche a livello di tuple per tabelle di riferimento, tabelle condivise, tabelle di catalogo e relativi indici
- Input:
  - `relation_name`(testo) — Il nome della relazione
  - `subcluster_id`(testo) — L'ID del sottocluster del nodo in cui devono essere estratte le statistiche
- Output:
  - Per le tabelle di riferimento e di catalogo (compresi i relativi indici), le colonne sono coerenti con quelle di Aurora PostgreSQL.
  - Per le tabelle suddivise, le statistiche rappresentano solo la partizione della tabella condivisa che risiede nel sottocluster specificato.

Funzioni di statistica dell'indice

`rds_aurora.limitless_pgstatindex(relation_name)`

- Scopo: Estrarre le statistiche per gli indici B-tree su tabelle standard
- Input: `relation_name` (testo) — Il nome dell'indice B-tree
- Output: `root_block_no` vengono restituite tutte le colonne tranne. Le colonne restituite sono coerenti con la `pgstatindex` funzione di Aurora PostgreSQL.

`rds_aurora.limitless_pgstatindex(relation_name, subcluster_id)`

- Scopo: Estrarre le statistiche per gli indici B-tree su tabelle di riferimento, tabelle ripartite e tabelle di catalogo.
- Input:
  - `relation_name`(testo) — Il nome dell'indice B-tree
  - `subcluster_id`(testo) — L'ID del sottocluster del nodo in cui devono essere estratte le statistiche
- Output:
  - Per gli indici delle tabelle di riferimento e del catalogo, vengono restituite tutte le colonne (tranne`root_block_no`). Le colonne restituite sono coerenti con Aurora PostgreSQL.
  - Per le tabelle suddivise, le statistiche rappresentano solo la partizione dell'indice della tabella condivisa che risiede nel sottocluster specificato. La `tree_level` colonna mostra la media di tutte le sezioni di tabella del sottocluster richiesto.

`rds_aurora.limitless_pgstatginindex(relation_name)`

- Scopo: Estrarre le statistiche per gli indici invertiti generalizzati () su tabelle standard GINs
- Input: `relation_name` (testo) — Il nome del GIN
- Output: colonne coerenti con quelle restituite dalla `pgstatginindex` funzione in Aurora PostgreSQL

`rds_aurora.limitless_pgstatginindex(relation_name, subcluster_id)`

- Scopo: Estrarre le statistiche per gli indici GIN su tabelle di riferimento, tabelle condivise e tabelle di catalogo.
- Input:
  - `relation_name`(testo) — Il nome dell'indice
  - `subcluster_id`(testo) — L'ID del sottocluster del nodo in cui devono essere estratte le statistiche
- Output:
  - Per gli indici GIN delle tabelle di riferimento e di catalogo, le colonne sono coerenti con quelle di Aurora PostgreSQL.
  - Per le tabelle suddivise, le statistiche rappresentano solo la partizione dell'indice della tabella condivisa che risiede nel sottocluster specificato.

`rds_aurora.limitless_pgstathashindex(relation_name)`

- Scopo: Estrarre le statistiche per gli indici hash su tabelle standard

- Input: `relation_name` (text) — Il nome dell'indice hash
- Output: colonne coerenti con quelle restituite dalla `pgstathashindex` funzione in Aurora PostgreSQL

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Scopo: Estrarre le statistiche per gli indici hash su tabelle di riferimento, tabelle suddivise e tabelle di catalogo.
- Input:
  - `relation_name`(testo) — Il nome dell'indice
  - `subcluster_id`(testo) — L'ID del sottocluster del nodo in cui devono essere estratte le statistiche
- Output:
  - Per gli indici hash delle tabelle di riferimento e del catalogo, le colonne sono coerenti con Aurora PostgreSQL.
  - Per le tabelle suddivise, le statistiche rappresentano solo la partizione dell'indice della tabella condivisa che risiede nel sottocluster specificato.

Funzioni di conteggio delle pagine

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`)`

- Scopo: Estrarre il conteggio delle pagine per le tabelle standard e i relativi indici
- Input: `relation_name` (testo) — Il nome della relazione
- Output: numero di pagine della relazione specificata

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`, `*`subcluster_id`*`)`

- Scopo: estrae il conteggio delle pagine per le tabelle di riferimento, le tabelle suddivise e le tabelle del catalogo (compresi i relativi indici)
- Input:
  - `relation_name`(testo) — Il nome della relazione
  - `subcluster_id`(testo) — L'ID del sottocluster del nodo in cui deve essere estratto il conteggio delle pagine
- Output: per le tabelle suddivise, il conteggio delle pagine è la somma delle pagine di tutte le sezioni della tabella nel sottocluster specificato.

Funzioni statistiche approssimative a livello di tuple

`rds_aurora.limitless_pgstattuple_approx(`*`relation_name`*`)`

- Scopo: Estrarre statistiche approssimative a livello di tuple per le tabelle standard e i relativi indici
- Input: `relation_name` (testo) — Il nome della relazione
- Output: colonne coerenti con quelle restituite dalla funzione pgstattuple_approx in Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple_approx(`*`relation_name`*`, `*`subcluster_id`*`)`

- Scopo: Estrarre statistiche approssimative a livello di tuple per tabelle di riferimento, tabelle suddivise e tabelle di catalogo (compresi i relativi indici)
- Input:
  - `relation_name`(testo) — Il nome della relazione
  - `subcluster_id`(testo) — L'ID del sottocluster del nodo in cui devono essere estratte le statistiche
- Output:
  - Per le tabelle di riferimento e di catalogo (compresi i relativi indici), le colonne sono coerenti con quelle di Aurora PostgreSQL.
  - Per le tabelle suddivise, le statistiche rappresentano solo la partizione della tabella condivisa che risiede nel sottocluster specificato.

> **Note**
>
> Attualmente, Aurora PostgreSQL Limitless Database non supporta l'`pgstattuple`estensione su viste materializzate, tabelle TOAST o tabelle temporanee. In Aurora PostgreSQL Limitless Database, è necessario fornire l'input come testo, sebbene Aurora PostgreSQL supporti altri formati.

Chiavi esterne

I vincoli Foreign key () sono supportati con alcune limitazioni: `FOREIGN KEY`

- `CREATE TABLE`with `FOREIGN KEY` è supportato solo per le tabelle standard. Per creare una tabella frammentata o di riferimento con`FOREIGN KEY`, create innanzitutto la tabella senza un vincolo di chiave esterna. Quindi modificatela utilizzando la seguente dichiarazione:

  ```
  ALTER TABLE ADD CONSTRAINT;
  ```

- La conversione di una tabella standard in una tabella frammentata o di riferimento non è supportata quando la tabella ha un vincolo di chiave esterna. Eliminate il vincolo, quindi aggiungetelo dopo la conversione.
- Le seguenti limitazioni si applicano ai tipi di tabella per i vincoli di chiave esterna:
  - Una tabella standard può avere un vincolo di chiave esterna rispetto a un'altra tabella standard.
  - Una tabella condivisa può avere un vincolo di chiave esterna se le tabelle principale e secondaria sono collocate e la chiave esterna è un superset della chiave shard.
  - Una tabella condivisa può avere un vincolo di chiave esterna rispetto a una tabella di riferimento.
  - Una tabella di riferimento può avere un vincolo di chiave esterna rispetto a un'altra tabella di riferimento.

Argomenti

- Opzioni a chiave esterna
- Esempi

Opzioni a chiave esterna

Le chiavi esterne sono supportate in Aurora PostgreSQL Limitless Database per alcune opzioni DDL. La tabella seguente elenca le opzioni supportate e non supportate tra le tabelle del database Aurora PostgreSQL Limitless.

Opzione DDL	Riferimento al riferimento	Da frammentato a frammentato (collocato)	Suddiviso in base al riferimento	Da standard a standard
`DEFERRABLE`	Sì	Sì	Sì	Sì

Opzione DDL	Riferimento al riferimento	Da frammentato a frammentato (collocato)	Suddiviso in base al riferimento	Da standard a standard
`INITIALLY DEFERRED`	Sì	Sì	Sì	Sì
`INITIALLY IMMEDIATE`	Sì	Sì	Sì	Sì
`MATCH FULL`	Sì	Sì	Sì	Sì
`MATCH PARTIAL`	No	No	No	No
`MATCH SIMPLE`	Sì	Sì	Sì	Sì
`NOT DEFERRABLE`	Sì	Sì	Sì	Sì
`NOT VALID`	Sì	No	No	Sì
`ON DELETE CASCADE`	Sì	Sì	Sì	Sì
`ON DELETE NO ACTION`	Sì	Sì	Sì	Sì
`ON DELETE RESTRICT`	Sì	Sì	Sì	Sì
`ON DELETE SET DEFAULT`	No	No	No	No
`ON DELETE SET NULL`	Sì	No	No	Sì
`ON UPDATE CASCADE`	No	No	No	Sì

Opzione DDL	Riferimento al riferimento	Da frammentato a frammentato (collocato)	Suddiviso in base al riferimento	Da standard a standard
`ON UPDATE NO ACTION`	Sì	Sì	Sì	Sì
`ON UPDATE RESTRICT`	Sì	Sì	Sì	Sì
`ON UPDATE SET DEFAULT`	No	No	No	No
`ON UPDATE SET NULL`	Sì	No	No	Sì

Esempi

- Da standard a standard:

```
set rds_aurora.limitless_create_table_mode='standard';

CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products (product_no),
    quantity integer
);

SELECT constraint_name, table_name, constraint_type
FROM information_schema.table_constraints WHERE constraint_type='FOREIGN KEY';

 constraint_name         | table_name  | constraint_type
-------------------------+-------------+-----------------
 orders_product_no_fkey  | orders      | FOREIGN KEY
```

```
(1 row)
```

- Da suddiviso a frammentato (collocato):

```
set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"product_no"}';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
set rds_aurora.limitless_create_table_collocate_with='products';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Suddiviso in base al riferimento:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Riferimento al riferimento:

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

Funzioni

Le funzioni sono supportate in Aurora PostgreSQL Limitless Database.

Le seguenti DDLs funzioni sono supportate:

CREATE FUNCTION

È possibile creare funzioni, come in Aurora PostgreSQL, con l'eccezione di modificarne la volatilità sostituendole.

Per ulteriori informazioni, consultare CREATE FUNCTION nella documentazione di PostgreSQL.

ALTER FUNCTION

È possibile modificare le funzioni, come in Aurora PostgreSQL, ad eccezione della modifica della loro volatilità.

Per ulteriori informazioni, vedere ALTER FUNCTION nella documentazione di PostgreSQL.

DROP FUNCTION

È possibile eliminare le funzioni, come in Aurora PostgreSQL.

```
DROP FUNCTION [ IF EXISTS ] name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ]
 [, ...]
    [ CASCADE | RESTRICT ]
```

Per ulteriori informazioni, vedere DROP FUNCTION nella documentazione di PostgreSQL.

Argomenti

- Distribuzione delle funzioni
- Volatilità della funzione

Distribuzione delle funzioni

Quando tutte le istruzioni di una funzione sono indirizzate a un singolo shard, è utile spostare l'intera funzione sullo shard di destinazione. Quindi il risultato viene propagato al router, anziché annullare la funzione sul router stesso. La funzionalità pushdown di funzioni e stored procedure è utile per i clienti

che desiderano eseguire la propria funzione o la stored procedure più vicino alla fonte di dati, ovvero allo shard.

Per distribuire una funzione, è necessario innanzitutto creare la funzione, quindi chiamare la `rds_aurora.limitless_distribute_function` procedura per distribuirla. Questa funzione utilizza la seguente sintassi:

```
SELECT rds_aurora.limitless_distribute_function('function_prototype',
 ARRAY['shard_key'], 'collocating_table');
```

La funzione accetta i seguenti parametri:

- *`function_prototype`*— La funzione da distribuire. Menziona solo gli argomenti di input e non nessuno degli argomenti di output.

  Se uno qualsiasi degli argomenti è definito come 0UT parametro, non includete il loro tipo negli argomenti di`function_prototype`.
- `ARRAY['`*`shard_key`*`']`— L'elenco degli argomenti della funzione identificati come chiave shard per la funzione.
- *`collocating_table`*— La tabella condivisa che contiene l'intervallo di dati sullo shard di destinazione.

Per identificare lo shard su cui attivare questa funzione per farla funzionare, il sistema prende l'`ARRAY['`*`shard_key`*`']`argomento, ne esegue l'hash e trova lo shard *`collocating_table`* che ospita l'intervallo contenente questo valore hash.

Restrizioni

Quando distribuite una funzione o una procedura, questa tratta solo i dati delimitati dall'intervallo di chiavi dello shard in quel frammento. Nei casi in cui la funzione o la procedura tenta di accedere ai dati da un altro shard, i risultati restituiti dalla funzione o dalla procedura distribuita saranno diversi rispetto a quelli non distribuiti.

Ad esempio, create una funzione contenente interrogazioni che toccheranno più shard, ma poi chiamate la `rds_aurora.limitless_distribute_function` procedura per distribuirla. Quando si richiama questa funzione fornendo argomenti per una chiave shard, è probabile che i risultati della sua esecuzione siano limitati dai valori presenti in tale shard. Questi risultati sono diversi da quelli prodotti senza distribuire la funzione.

Esempi

Consideriamo la seguente funzione `func` in cui abbiamo la tabella condivisa `customers` con il tasto shard. `customer_id`

```
postgres_limitless=> CREATE OR REPLACE FUNCTION func(c_id integer, sc integer)
 RETURNS int
    language SQL
    volatile
    AS $$
    UPDATE customers SET score = sc WHERE customer_id = c_id RETURNING score;
    $$;
```

Ora distribuiamo questa funzione:

```
SELECT rds_aurora.limitless_distribute_function('func(integer, integer)',
 ARRAY['c_id'], 'customers');
```

Di seguito sono riportati alcuni esempi di piani di interrogazione.

```
EXPLAIN(costs false, verbose true) SELECT func(27+1,10);

                    QUERY PLAN
 ---------------------------------------------------
  Foreign Scan
    Output: (func((27 + 1), 10))
    Remote SQL:  SELECT func((27 + 1), 10) AS func
  Single Shard Optimized
 (4 rows)
```

```
EXPLAIN(costs false, verbose true)
 SELECT * FROM customers,func(customer_id, score) WHERE customer_id=10 AND score=27;

                          QUERY PLAN
 ---------------------------------------------------------------------
  Foreign Scan
    Output: customer_id, name, score, func
    Remote SQL:  SELECT customers.customer_id,
      customers.name,
      customers.score,
      func.func
```

```
     FROM public.customers,
      LATERAL func(customers.customer_id, customers.score) func(func)
    WHERE ((customers.customer_id = 10) AND (customers.score = 27))
  Single Shard Optimized
 (10 rows)
```

L'esempio seguente mostra una procedura con OUT parametri IN e parametri come argomenti.

```
CREATE OR REPLACE FUNCTION get_data(OUT id INTEGER, IN arg_id INT)
    AS $$
    BEGIN
        SELECT customer_id,
        INTO id
        FROM customer
        WHERE customer_id = arg_id;
    END;
    $$ LANGUAGE plpgsql;
```

L'esempio seguente distribuisce la procedura utilizzando solo IN parametri.

```
EXPLAIN(costs false, verbose true) SELECT * FROM get_data(1);

             QUERY PLAN
 -----------------------------------
  Foreign Scan
    Output: id
    Remote SQL:  SELECT customer_id
     FROM get_data(1) get_data(id)
  Single Shard Optimized
 (6 rows)
```

Volatilità della funzione

Puoi determinare se una funzione è immutabile, stabile o volatile controllando il `provolatile` valore nella vista pg_proc. Il `provolatile` valore indica se il risultato della funzione dipende solo dai suoi argomenti di input o è influenzato da fattori esterni.

Il valore è uno dei seguenti:

- `i`— Funzioni immutabili, che forniscono sempre lo stesso risultato per gli stessi input
- `s`— Funzioni stabili, i cui risultati (per input fissi) non cambiano durante una scansione

- v— Funzioni volatili, i cui risultati potrebbero cambiare in qualsiasi momento. Utilizzatele anche v per funzioni con effetti collaterali, in modo che le chiamate ad esse non possano essere ottimizzate.

Gli esempi seguenti mostrano le funzioni volatili.

```
SELECT proname, provolatile FROM pg_proc WHERE proname='pg_sleep';

 proname  | provolatile
----------+-------------
 pg_sleep | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='uuid_generate_v4';

     proname      | provolatile
------------------+-------------
 uuid_generate_v4 | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='nextval';

 proname | provolatile
---------+-------------
 nextval | v
(1 row)
```

La modifica della volatilità di una funzione esistente non è supportata in Aurora PostgreSQL Limitless Database. Questo vale sia per i comandi che per `ALTER FUNCTION` i `CREATE OR REPLACE FUNCTION` comandi, come illustrato negli esempi seguenti.

```
-- Create an immutable function
CREATE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$IMMUTABLE;

-- Altering the volatility throws an error
ALTER FUNCTION immutable_func1 STABLE;
```

```
-- Replacing the function with altered volatility throws an error
CREATE OR REPLACE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$VOLATILE;
```

Consigliamo vivamente di assegnare le volatilità corrette alle funzioni. Ad esempio, se la funzione utilizza `SELECT` più tabelle o fa riferimento a oggetti del database, non impostarla come. `IMMUTABLE` Se il contenuto della tabella cambia, l'immutabilità viene interrotta.

Aurora PostgreSQL consente funzioni `SELECT` interne immutabili, ma i risultati potrebbero essere errati. Aurora PostgreSQL Limitless Database può restituire sia errori che risultati errati. Per ulteriori informazioni sulla volatilità delle funzioni, consulta Categorie di volatilità delle funzioni nella documentazione di PostgreSQL.

Sequenze

Le sequenze denominate sono oggetti di database che generano numeri univoci in ordine crescente o decrescente. `CREATE SEQUENCE`crea un nuovo generatore di numeri di sequenza. È garantito che i valori di sequenza siano unici.

Quando si crea una sequenza denominata in Aurora PostgreSQL Limitless Database, viene creato un oggetto sequenza distribuito. Quindi Aurora PostgreSQL Limitless Database distribuisce blocchi di valori di sequenza non sovrapposti su tutti i Distributed Transaction Router (router). I blocchi sono rappresentati come oggetti di sequenza locale sui router; pertanto, le operazioni di sequenza come e vengono eseguite localmente. `nextval` `currval` I router funzionano in modo indipendente e richiedono nuovi blocchi dalla sequenza distribuita quando necessario.

Per ulteriori informazioni sulle sequenze, vedere CREATE SEQUENCE nella documentazione di PostgreSQL.

Argomenti

- Richiedere un nuovo blocco
- Limitazioni
- Opzioni non supportate
- Esempi
- viste in sequenza
- Risoluzione dei problemi relativi alla sequenza

Richiedere un nuovo blocco

È possibile configurare le dimensioni dei blocchi allocati sui router utilizzando il parametro. `rds_aurora.limitless_sequence_chunk_size` Il valore predefinito è `250000`. Inizialmente ogni router possiede due blocchi: attivi e riservati. I blocchi attivi vengono utilizzati per configurare gli oggetti della sequenza locale (impostazione `minvalue` e`maxvalue`) e i blocchi riservati vengono archiviati in una tabella interna del catalogo. Quando un blocco attivo raggiunge il valore minimo o massimo, viene sostituito dal blocco riservato. Per fare ciò, `ALTER SEQUENCE` viene utilizzato internamente, il che significa che `AccessExclusiveLock` viene acquisito.

I background worker vengono eseguiti ogni 10 secondi sui nodi del router per scansionare le sequenze alla ricerca di blocchi riservati utilizzati. Se viene trovato un blocco usato, il lavoratore

richiede un nuovo blocco dalla sequenza distribuita. Assicurati di impostare la dimensione del blocco sufficientemente grande da consentire ai lavoratori in background di avere abbastanza tempo per richiederne di nuovi. Le richieste remote non avvengono mai nel contesto delle sessioni utente, il che significa che non è possibile richiedere direttamente una nuova sequenza.

Limitazioni

Le seguenti limitazioni si applicano alle sequenze in Aurora PostgreSQL Limitless Database:

- Il `pg_sequence` catalogo, la `pg_sequences` funzione e l'`SELECT * FROM` *`sequence_name`*istruzione mostrano tutti solo lo stato della sequenza locale, non lo stato distribuito.
- È garantito che i valori di sequenza siano unici e monotoni all'interno di una sessione. Tuttavia, possono non funzionare correttamente se le `nextval` istruzioni vengono eseguite in altre sessioni, se tali sessioni sono connesse ad altri router.
- Assicurati che la dimensione della sequenza (numero di valori disponibili) sia sufficientemente grande da poter essere distribuita su tutti i router. Utilizzate il `rds_aurora.limitless_sequence_chunk_size` parametro per configurare. `chunk_size` (Ogni router ha due blocchi).
- L'`CACHE`opzione è supportata, ma la cache deve essere più piccola di. `chunk_size`

Opzioni non supportate

Le seguenti opzioni non sono supportate per le sequenze in Aurora PostgreSQL Limitless Database.

Funzioni di manipolazione di sequenze

La funzione non è supportata. `setval` Per ulteriori informazioni, consulta Funzioni di manipolazione delle sequenze nella documentazione di PostgreSQL.

CREATE SEQUENCE

Le seguenti opzioni non sono supportate.

```
CREATE [{ TEMPORARY | TEMP} | UNLOGGED] SEQUENCE
    [[ NO ] CYCLE]
```

Per ulteriori informazioni, vedere CREATE SEQUENCE nella documentazione di PostgreSQL.

ALTERARE LA SEQUENZA

Le seguenti opzioni non sono supportate.

```
ALTER SEQUENCE
    [[ NO ] CYCLE]
```

Per ulteriori informazioni, vedere ALTER SEQUENCE nella documentazione di PostgreSQL.

ALTER TABLE

Il `ALTER TABLE` comando non è supportato per le sequenze.

Esempi

CREA/RILASCIA UNA SEQUENZA

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
       1
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16960 |       20 |        1 |            1 |  10000 |      1 |        1 | f
(1 row)

% connect to another router
postgres_limitless=> SELECT nextval('s');

 nextval
---------
   10001
(1 row)
```

```
postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16959 |       20 |    10001 |            1 |  20000 |  10001 |        1 | f
(1 row)

postgres_limitless=> DROP SEQUENCE s;
DROP SEQUENCE
```

ALTERARE LA SEQUENZA

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> ALTER SEQUENCE s RESTART 500;
ALTER SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
     500
(1 row)

postgres_limitless=> SELECT currval('s');

 currval
---------
     500
(1 row)
```

Funzioni di manipolazione di sequenze

```
postgres=# CREATE TABLE t(a bigint primary key, b bigint);
CREATE TABLE

postgres=# CREATE SEQUENCE s minvalue 0 START 0;
CREATE SEQUENCE
```

```
postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));
INSERT 0 1

postgres=# SELECT * FROM t;

 a | b
---+---
 0 | 0
 1 | 1
(2 rows)

postgres=# ALTER SEQUENCE s RESTART 10000;
ALTER SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# SELECT * FROM t;

   a   |   b
-------+-------
     0 |     0
     1 |     1
 10000 | 10000
(3 rows)
```

viste in sequenza

Aurora PostgreSQL Limitless Database offre le seguenti viste per le sequenze.

rds_aurora.limitless_distributed_sequence

Questa vista mostra lo stato e la configurazione della sequenza distribuiti. Le `cache` colonne `minvaluemaxvalue`,`start`,`inc`, e hanno lo stesso significato della vista pg_sequences e mostrano le opzioni con cui è stata creata la sequenza. La `lastval` colonna mostra l'ultimo valore allocato o riservato in un oggetto di sequenza distribuito. Ciò non significa che il valore sia già stato utilizzato, poiché i router mantengono localmente i blocchi di sequenza.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_distributed_sequence WHERE
 sequence_name='test_serial_b_seq';

 schema_name |   sequence_name   | lastval | minvalue |  maxvalue  | start | inc |
 cache
-------------+-------------------+---------+----------+------------+-------+-----
+-------
 public      | test_serial_b_seq | 1250000 |        1 | 2147483647 |     1 |   1 |
   1
(1 row)
```

rds_aurora.limitless_sequence_metadata

Questa vista mostra i metadati di sequenza distribuiti e aggrega i metadati di sequenza dai nodi del cluster. Utilizza le seguenti colonne:

- `subcluster_id`— L'ID del nodo del cluster che possiede un blocco.
- Blocco attivo: un blocco di una sequenza che viene utilizzato (`active_minvalue`,). `active_maxvalue`
- Chunk riservato: il blocco locale che verrà utilizzato successivamente (,). `reserved_minvalue` `reserved_maxvalue`
- `local_last_value`— L'ultimo valore osservato da una sequenza locale.
- `chunk_size`— La dimensione di un blocco, come configurato al momento della creazione.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_sequence_metadata WHERE
 sequence_name='test_serial_b_seq' order by subcluster_id;
```

```
 subcluster_id |   sequence_name   | schema_name | active_minvalue | active_maxvalue
 | reserved_minvalue | reserved_maxvalue | chunk_size | chunk_state |
 local_last_value
---------------+-------------------+-------------+-----------------
+-----------------+-------------------+-------------------+------------
+-------------+------------------
 1             | test_serial_b_seq | public      |          500001 |          750000
 |           1000001 |           1250000 |     250000 |           1 |
 550010
 2             | test_serial_b_seq | public      |          250001 |          500000
 |            750001 |           1000000 |     250000 |           1 |

(2 rows)
```

Risoluzione dei problemi relativi alla sequenza

I seguenti problemi possono verificarsi con le sequenze.

La dimensione del blocco non è sufficientemente grande

Se la dimensione del blocco non è sufficientemente grande e il tasso di transazione è elevato, i lavoratori in background potrebbero non avere abbastanza tempo per richiedere nuovi blocchi prima che i blocchi attivi siano esauriti. Ciò può portare a situazioni di contesa e attesa come,, e. `LIMITLESS:AuroraLimitlessSequenceReplace` `LWLock:LockManager` `Lockrelation` `LWlock:bufferscontent`

Incrementa il valore del parametro `rds_aurora.limitless_sequence_chunk_size`.

La cache delle sequenze è troppo alta

In PostgreSQL, la memorizzazione nella cache delle sequenze avviene a livello di sessione. Ogni sessione assegna valori di sequenza successivi durante un accesso all'oggetto sequenza e aumenta di conseguenza quelli dell'oggetto sequenza. `last_value` Quindi, gli usi `nextval` successivi di quella sessione restituiscono semplicemente i valori preallocati senza toccare l'oggetto della sequenza.

Tutti i numeri assegnati ma non utilizzati all'interno di una sessione vengono persi al termine di tale sessione, con conseguenti «buchi» nella sequenza. Ciò può consumare rapidamente sequence_chunk e portare a eventi di contesa e attesa come,, e. `LIMITLESS:AuroraLimitlessSequenceReplace` `LWLock:LockManager` `Lockrelation` `LWlock:bufferscontent`

Riduci l'impostazione della cache della sequenza.

La figura seguente mostra gli eventi di attesa causati da problemi di sequenza.

Comandi SQL per l'elaborazione delle query e del linguaggio di manipolazione dei dati (DML) supportati e non supportati

La tabella seguente elenca i comandi DML supportati e non supportati da Aurora PostgreSQL Limitless Database, con riferimenti a limitazioni o ulteriori informazioni.

Comando	Supportato?	Limitazioni o ulteriori informazioni
ABORT	Sì	Nessuno
ANALYZE	Sì	ANALYZE
BEGIN	Sì	Nessuno
CALL	Sì	Nessuno
CHECKPOINT	Sì	Nessuno
CLOSE	Sì	Nessuno
CLUSTER	Sì	CLUSTER
COMMIT	Sì	Nessuno
IMPEGNARSI PREPARATI	No	Non applicabile
COPY	Sì	Nessuno
DEALLOCATE	Sì	Nessuno
DECLARE	Sì	Nessuno
DELETE	Sì	Nessuno
SCARTARE	Sì	Nessuno
DO	Sì	Nessuno
END	Sì	Nessuno

Comando	Supportato?	Limitazioni o ulteriori informazioni
EXECUTE	Sì	Nessuno
EXPLAIN	Sì	EXPLAIN
FETCH	Sì	Nessuno
IMPORTARE UNO SCHEMA ESTERNO	No	Non applicabile
INSERT	Sì	INSERT
LISTEN	No	Non applicabile
LOCK	Sì	Nessuno
MERGE	No	Non applicabile
MOVE	Sì	Nessuno
NOTIFY	No	Non applicabile
OPEN	Sì	Nessuno
PREPARE	Sì	Nessuno
PREPARA LA TRANSAZIONE	No	Non applicabile
REFRESH MATERIALIZED VIEW	No	Non applicabile
REINDEX	No	Non applicabile
PUNTO DI SALVATAGGIO DI RILASCIO	Sì	Nessuno
ROLLBACK	Sì	Nessuno
ROLLBACK PREPARATO	No	Non applicabile

Comando	Supportato?	Limitazioni o ulteriori informazioni
ROLLBACK AL SAVEPOINT	Sì	Nessuno
SAVEPOINT	Sì	Nessuno
SELECT	Sì	Nessuno
SELECT INTO	Sì	Nessuno
MOSTRA	Sì	Nessuno
START TRANSACTION	Sì	Nessuno
UNLISTEN	No	Nessuno
UPDATE	Sì	UPDATE
AGGIORNA... DOVE È ATTUALE	No	Non applicabile
VACUUM	Sì	VACUUM
VALUES	Sì	Nessuno

DMLlimitazioni e altre informazioni per Aurora SQL Postgre Limitless Database

I seguenti argomenti descrivono le limitazioni o forniscono ulteriori informazioni DML e SQL comandi di elaborazione delle query in Aurora SQL Postgre Limitless Database.

Argomenti

- ANALYZE
- CLUSTER
- EXPLAIN
- INSERT
- UPDATE
- VACUUM

ANALYZE

Il `ANALYZE` comando raccoglie statistiche sul contenuto delle tabelle nel database. Successivamente, il pianificatore di query utilizza queste statistiche per determinare i piani di esecuzione più efficienti per le query. Per ulteriori informazioni, consulta la documentazione ANALYZEdi SQL Postgre.

In Aurora Postgre SQL Limitless Database, il `ANALYZE` comando raccoglie le statistiche delle tabelle su tutti i router e gli shard durante l'esecuzione.

Per impedire il calcolo delle statistiche su ogni router durante l'`ANALYZE`esecuzione, le statistiche delle tabelle vengono calcolate su uno dei router e quindi copiate sui router peer.

CLUSTER

Il `CLUSTER` comando riordina fisicamente una tabella in base a un indice. L'indice deve essere già stato definito nella tabella. In Aurora Postgre SQL Limitless Database, il clustering è locale rispetto alla parte dell'indice presente su ogni shard.

Per ulteriori informazioni, consulta la documentazione di Postgre. CLUSTERSQL

EXPLAIN

Utilizzate il seguente parametro per configurare l'output del `EXPLAIN` comando:

- `rds_aurora.limitless_explain_options`— Cosa includere nell'`EXPLAIN`output. Il valore predefinito è`single_shard_optimization`: viene mostrato se i piani sono ottimizzati per shard singolo, ma i piani shard non sono inclusi.

In questo esempio, l'`EXPLAIN`output non mostra i piani ricavati dagli shard.

```
postgres_limitless=> EXPLAIN SELECT * FROM employees where id =25;

                      QUERY PLAN
------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
 Single Shard Optimized
(2 rows)
```

Ora impostiamo l'opzione `rds_aurora.limitless_explain_options` per includere `shard_plans` e`single_shard_optimization`. Possiamo visualizzare i piani di esecuzione delle istruzioni sia sui router che sugli shard. Inoltre, disabilitiamo il `enable_seqscan` parametro per far sì che la scansione dell'indice venga utilizzata sullo shard layer.

```
postgres_limitless=> SET rds_aurora.limitless_explain_options = shard_plans,
 single_shard_optimization;
SET

postgres_limitless=> SET enable_seqscan = OFF;
SET

postgres_limitless=> EXPLAIN SELECT * FROM employees WHERE id = 25;

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Remote Plans from Shard postgres_s4:
         Index Scan using employees_ts00287_id_idx on employees_ts00287
 employees_fs00003  (cost=0.14..8.16 rows=1 width=15)
           Index Cond: (id = 25)
 Single Shard Optimized
(5 rows)
```

Per ulteriori informazioni sul `EXPLAIN` comando, consulta EXPLAINla documentazione di SQL Postgre.

INSERT

La maggior parte dei `INSERT` comandi è supportata in Aurora SQL Postgre Limitless Database.

Postgre SQL non ha un `UPSERT` comando esplicito, ma supporta le istruzioni. `INSERT ... ON CONFLICT`

`INSERT ... ON CONFLICT`non è supportato se l'azione di conflitto ha una sottoquery o una funzione mutabile:

```
-- RANDOM is a mutable function.
INSERT INTO sharded_table VALUES (1, 100) ON CONFLICT (id) DO UPDATE SET other_id =
 RANDOM();

ERROR: Aurora Limitless Tables doesn't support pushdown-unsafe functions with DO UPDATE
 clauses.
```

Per ulteriori informazioni sul `INSERT` comando, consulta INSERTla documentazione di SQL Postgre.

UPDATE

L'aggiornamento della chiave shard non è supportato. Ad esempio, hai una tabella condivisa chiamata`customers`, con una chiave shard. `customer_id` Le seguenti DML istruzioni causano errori:

```
postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_id =1;
ERROR:  Shard key column update is not supported

postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_name='abc';
ERROR:  Shard key column update is not supported
```

Per aggiornare una chiave shard, devi prima accedere alla riga con `DELETE` la shard key, poi a `INSERT` una nuova riga con il valore della shard key aggiornato.

Per ulteriori informazioni sul `UPDATE` comando, consulta Aggiornamento dei dati nella documentazione di Postgre. SQL

VACUUM

È possibile eseguire l'aspirazione sia su tabelle frammentate che su tabelle di riferimento. Le seguenti `VACUUM` funzioni sono completamente supportate in Aurora SQL Postgre Limitless Database:

- VACUUM
- ANALYZE
- DISABLE_PAGE_SKIPPING
- FREEZE
- FULL
- INDEX_CLEANUP
- PARALLEL
- PROCESS_TOAST
- TRUNCATE
- VERBOSE

`VACUUM`su Aurora Postgre SQL Limitless Database presenta le seguenti limitazioni:

- L'estensione pg_visibility_map non è supportata.
- Il controllo degli indici non utilizzati con la vista pg_stat_all_indexes non è supportato.
- Le viste consolidate per pg_stat_user_indexes, pg_class e pg_stats non sono implementate.

Per ulteriori informazioni sul comando, consulta la `VACUUM` documentazione di VACUUMPostgre. SQL Per ulteriori informazioni su come funziona l'aspirazione in Aurora SQL Postgre Limitless Database, vedere. Recupero dello spazio di archiviazione mediante aspirazione

Variabili nel database Aurora SQL Postgre Limitless

È possibile utilizzare le seguenti variabili per configurare Aurora SQL Postgre Limitless Database.

rds_aurora.limitless_active_shard_key

Imposta una singola chiave shard durante l'interrogazione del database, facendo sì che tutte le query e vengano aggiunte con la chiave shard come predicato costante. `SELECT` DML Per ulteriori informazioni, consulta Impostazione di una chiave shard attiva.

rds_aurora.limitless_create_table_collocate_with

Imposta questa variabile su un nome di tabella specifico per collocare le tabelle appena create con quella tabella. Per ulteriori informazioni, consulta Creazione di tabelle illimitate utilizzando variabili.

rds_aurora.limitless_create_table_mode

Imposta la modalità di creazione della tabella. Per ulteriori informazioni, consulta Creazione di tabelle illimitate utilizzando variabili.

rds_aurora.limitless_create_table_shard_key

Imposta questa variabile su una matrice di nomi di colonna da usare come chiavi shard. Per ulteriori informazioni, consulta Creazione di tabelle illimitate utilizzando variabili.

rds_aurora.limitless_explain_options

Cosa includere nell'output. `EXPLAIN` Per ulteriori informazioni, consulta EXPLAIN.

Parametri del cluster DB nel database Aurora SQL Postgre Limitless

È possibile utilizzare i seguenti parametri del cluster DB per configurare Aurora SQL Postgre Limitless Database.

rds_aurora.limitless_adaptive_fetch_size

Migliora la prefetching in batch. Se impostato su`true`, questo parametro consente una dimensione di recupero autoregolante (adattativa) per il prefetching. Se impostato su, la dimensione di recupero è `false` costante.

rds_aurora.limitless_auto_scale_options

Imposta le opzioni disponibili per aggiungere router o dividere gli shard in un gruppo di shard DB. Il valore può essere `add_router` o entrambi. `split_shard`

Per ulteriori informazioni, consulta Aggiungere un router a un gruppo di shard DB e Dividere uno shard in un gruppo di shard DB.

rds_aurora.limitless_distributed_deadlock_timeout

Il tempo di attesa di un blocco prima di verificare se esiste una condizione di deadlock distribuito, in millisecondi. L'impostazione predefinita è `1000` (1 secondo).

Per ulteriori informazioni, consulta Deadlock distribuiti nel database Aurora Postgre Limitless SQL.

rds_aurora.limitless_enable_auto_scale

Consente l'aggiunta di router e la suddivisione degli shard in un gruppo di shard DB.

Per ulteriori informazioni, consulta Aggiungere un router a un gruppo di shard DB e Dividere uno shard in un gruppo di shard DB.

rds_aurora.limitless_finalize_split_shard_mode

Determina come vengono finalizzate le suddivisioni degli shard avviate dal sistema. Per ulteriori informazioni, consulta Dividere uno shard in un gruppo di shard DB.

rds_aurora.limitless_maximum_adaptive_fetch_size

Imposta il limite superiore per la dimensione adattiva del recupero. L'intervallo è`1`:. `INT_MAX` Il valore predefinito è `1000`.

Utilizzo del database globale Amazon Aurora

Con la funzionalità Amazon Aurora Global Database, puoi configurare più cluster Aurora DB che si estendono su più cluster. Regioni AWS Aurora sincronizza automaticamente tutte le modifiche apportate nel cluster DB primario con uno o più cluster secondari. Un database globale Aurora ha un cluster DB primario in una regione e fino a 10 cluster DB secondari in diverse regioni. Questa configurazione multiregionale offre un ripristino rapido da rare interruzioni che potrebbero interessare un'intera area. Regione AWS La disponibilità di una copia completa di tutti i dati in più aree geografiche consente inoltre operazioni di lettura a bassa latenza per le applicazioni che si connettono da località molto diverse in tutto il mondo.

Argomenti

- Panoramica del database globale di Amazon Aurora
- Vantaggi del database globale di Amazon Aurora
- Disponibilità di regioni e versioni
- Limitazioni del database globale di Amazon Aurora
- Inizia a usare Amazon Aurora Global Database
- Gestione di un database globale Amazon Aurora
- Connessione al database globale di Amazon Aurora
- Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora
- Utilizzo dello switchover o del failover in Amazon Aurora Global Database
- Monitoraggio di un database globale Amazon Aurora
- Utilizzo di Amazon Aurora global database con altri servizi AWS
- Aggiornamento di un database globale Amazon Aurora

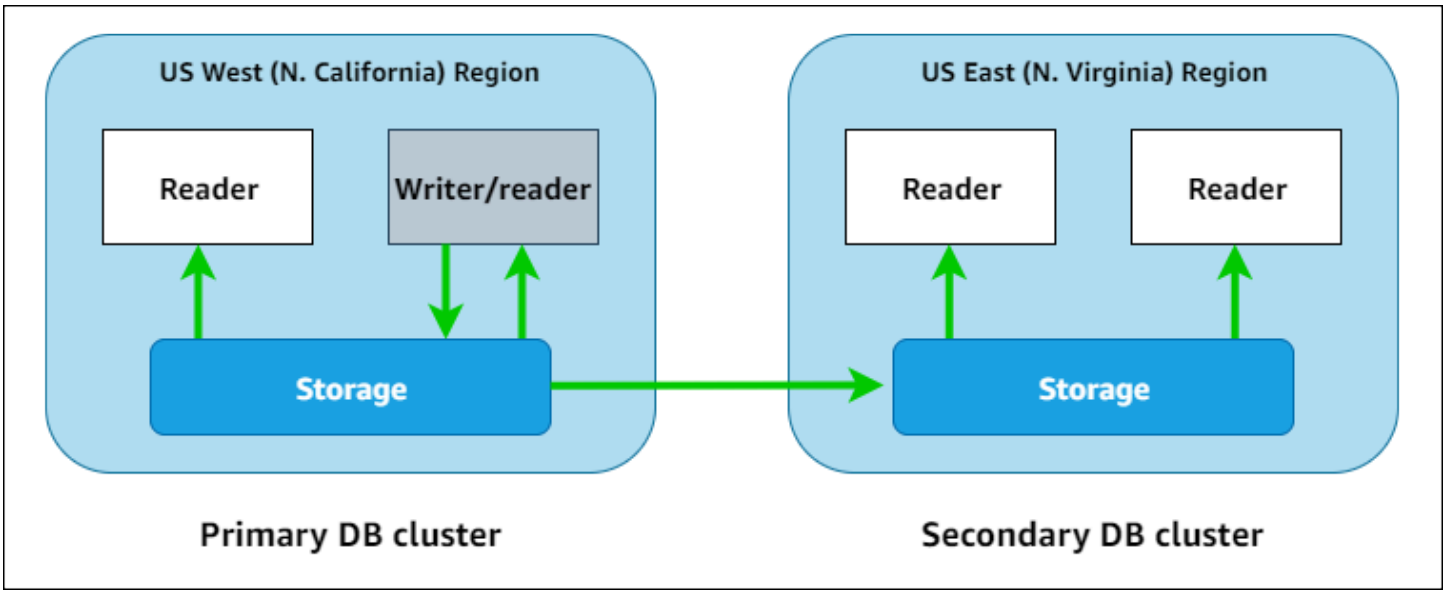
Panoramica del database globale di Amazon Aurora

Utilizzando la funzionalità Amazon Aurora Global Database, puoi eseguire le tue applicazioni distribuite a livello globale utilizzando un singolo database Aurora che si estende su più database. Regioni AWS

Un database globale Aurora è costituito da un database primario Regione AWS in cui vengono scritti i dati e fino a 10 secondari di sola lettura. Regioni AWS Eseguite operazioni di scrittura sul cluster DB

primario nel cluster primario. Regione AWS Il modo più comodo per farlo è connettersi all'endpoint Aurora Global Database writer, che punta sempre al cluster DB primario, anche dopo uno switchover o un failover verso un altro. Regione AWS Dopo ogni operazione di scrittura, Aurora replica i dati sul secondario Regioni AWS utilizzando un'infrastruttura dedicata, con una latenza in genere inferiore a un secondo.

Nel diagramma seguente, è possibile trovare un esempio di database globale Aurora che si estende su due. Regioni AWS



Puoi scalare ogni cluster secondario in modo indipendente, aggiungendo una o più istanze di lettore Aurora per servire carichi di lavoro di sola lettura. Puoi utilizzare le istanze For the Reader Aurora Serverless v2 per una scalabilità ancora più granulare e flessibile.

Solo il cluster primario eseguire operazioni di scrittura. I client che eseguono operazioni di scrittura si connettono all'endpoint di scrittura Aurora Global Database, che punta sempre all'istanza Writer DB del cluster primario. Come illustrato nel diagramma, Aurora utilizza il volume di archiviazione del cluster e non il motore di database per una replica rapida e a basso sovraccarico. Per ulteriori informazioni, consulta Panoramica dell'archiviazione di Amazon Aurora.

Aurora Global Database è progettato per applicazioni con un'impronta mondiale. I cluster DB secondari multipli di sola lettura Regioni AWS aiutano a ottimizzare le operazioni di lettura più vicine agli utenti delle applicazioni. Utilizzando la funzionalità di inoltro della scrittura, è inoltre possibile configurare il database globale in modo che i cluster secondari inviino richieste di scrittura al primario. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Aurora Global Database supporta due diverse operazioni per modificare la regione del cluster DB primario, a seconda dello scenario: lo switchover di Aurora Global Database e il failover di Aurora Global Database.

- Per le procedure operative pianificate come la rotazione regionale, utilizzate il meccanismo di switchover (precedentemente chiamato «failover pianificato gestito»). Con questa funzionalità, puoi trasferire il cluster primario di un database globale Aurora sano in una delle sue regioni secondarie senza perdita di dati. Per ulteriori informazioni, consulta Esecuzione di switchover per database globali Amazon Aurora.
- Per ripristinare l'Aurora Global Database dopo un'interruzione nella regione principale, utilizzate il meccanismo di failover. Con questa funzionalità, si esegue un failover dal cluster DB primario verso un'altra regione (failover tra regioni). Per ulteriori informazioni, consulta Esecuzione di failover gestiti per database globali Aurora.

Vantaggi del database globale di Amazon Aurora

Utilizzando Aurora Global Database, è possibile ottenere i seguenti vantaggi:

- Letture globali con latenza locale: se hai uffici in tutto il mondo, puoi utilizzare Aurora Global Database per mantenere aggiornate le tue principali fonti di informazioni in modo primario. Regione AWS Gli uffici nelle altre regioni possono accedere alle informazioni nella propria regione, con una latenza locale.
- Cluster Aurora DB secondari scalabili: puoi scalare i tuoi cluster secondari aggiungendo più istanze di sola lettura a un cluster secondario. Regione AWS Il cluster secondario è di sola lettura, quindi può supportare fino a 16 istanze DB di sola lettura anziché il normale limite di 15 per un singolo cluster Aurora.
- Replica rapida dai cluster Aurora DB primari a quelli secondari: la replica eseguita da Aurora Global Database ha un impatto minimo sulle prestazioni del cluster DB primario. Le risorse delle istanze database sono totalmente dedicate a servire carichi di lavoro di lettura e scrittura delle applicazioni.
- Ripristino da interruzioni a livello regionale: i cluster secondari consentono di rendere disponibile un database globale Aurora in un nuovo database primario Regione AWS più rapidamente (RTO inferiore) e con una minore perdita di dati (RPO inferiore) rispetto alle soluzioni di replica tradizionali.

Disponibilità di regioni e versioni

Il supporto e la disponibilità di questa funzionalità variano a seconda delle versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni con Aurora Global Database, consulta. Regioni e motori DB supportati per i database globali Aurora

Limitazioni del database globale di Amazon Aurora

Le seguenti limitazioni si applicano attualmente ad Aurora Global Database:

- Aurora Global Database è disponibile in alcune Regioni AWS e per versioni specifiche di Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta Regioni e motori DB supportati per i database globali Aurora.
- Aurora Global Database ha requisiti di configurazione specifici per le classi di istanze Aurora DB supportate, il numero massimo di Regioni AWS e così via. Per ulteriori informazioni, consulta Requisiti di configurazione di un database globale Amazon Aurora.
- Per la compatibilità con Aurora MySQL con MySQL 5.7, gli switchover di Aurora Global Database richiedono la versione 2.09.1 o una versione secondaria superiore.
- È possibile eseguire switchover o failover gestiti tra regioni con Aurora Global Database solo se i cluster DB primari e secondari hanno le stesse versioni del motore principale e secondario. A seconda del motore e delle versioni del motore, potrebbe essere necessario che i livelli di patch siano identici oppure i livelli di patch possono essere diversi. Per un elenco dei motori e delle versioni del motore che consentono queste operazioni tra cluster primari e secondari con diversi livelli di patch, vedereCompatibilità del livello di patch per switchover e failover gestiti tra regioni. Se le versioni del motore richiedono livelli di patch identici, è possibile eseguire il failover manualmente seguendo la procedura riportata di seguito. Esecuzione di failover gestiti per database globali Aurora
- Aurora Global Database attualmente non supporta le seguenti funzionalità di Aurora:
  - Aurora Serverless v1
  - Backtrack in Aurora
- Per le limitazioni all'utilizzo della funzionalità RDS Proxy con Aurora Global Database, vedere. Limitazioni di Server proxy per RDS con i database globali
- L'aggiornamento automatico della versione secondaria non si applica ai cluster Aurora MySQL e Aurora PostgreSQL che fanno parte di un database globale. Si noti che questa impostazione

può essere specificata per un'istanza database che fa parte di un cluster di database globale, ma l'impostazione non ha effetto.

- Aurora Global Database attualmente non supporta Aurora Auto Scaling per cluster DB secondari.
- Per utilizzare Database Activity Streams (DAS) su Aurora Global Database che esegue Aurora MySQL 5.7, la versione del motore deve essere la versione 2.08 o successiva. Per informazioni su DAS, vedere. Monitoraggio di Amazon Aurora tramite i flussi di attività del database
- Le seguenti limitazioni si applicano attualmente all'aggiornamento di Aurora Global Database:
  - Non è possibile applicare un gruppo di parametri personalizzato al cluster di database globale mentre si esegue un aggiornamento della versione principale del database globale Aurora. È possibile creare i gruppi di parametri personalizzati in ciascuna Regione del cluster globale e applicarli manualmente ai cluster regionali dopo l'aggiornamento.
  - Con un database globale Aurora basato su Aurora MySQL, non puoi eseguire un aggiornamento locale da Aurora MySQL versione 2 alla versione 3 se il parametro `lower_case_table_names` è attivato. Per ulteriori informazioni sui metodi disponibili all'uso, consulta Aggiornamenti di una versione principale.
  - Con Aurora Global Database, non è possibile eseguire un aggiornamento della versione principale del motore Aurora PostgreSQL DB se la funzionalità Recovery Point Objective (RPO) è attivata. Per ulteriori informazioni sulla caratteristica RPO, consulta Gestione RPOs di database globali basati su Aurora PostgreSQL.
  - Con un Aurora Global Database, non è possibile eseguire un aggiornamento di versione minore da Aurora MySQL versione 3.01 o 3.02 a 3.03 o superiore utilizzando il processo standard. Per informazioni dettagliate sul processo da usare, consulta Aggiornamento di Aurora MySQL modificando la versione del motore.

  Per informazioni sull'aggiornamento di Aurora Global Database, vedere. Aggiornamento di un database globale Amazon Aurora
- Non puoi arrestare o avviare singolarmente i cluster Aurora DB nel tuo database globale. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.
- Le istanze DB del lettore Aurora collegate al cluster Aurora DB secondario possono essere riavviate in determinate circostanze. Se l'istanza DB Regione AWS di scrittura principale viene riavviata o ripristinata, vengono riavviate anche le istanze DB Reader nelle regioni secondarie. Il cluster secondario non sarà quindi disponibile finché tutte le istanze DB di lettura non saranno nuovamente sincronizzate con l'istanza writer del cluster DB primario. Il comportamento del cluster primario durante il riavvio o durante un failover è lo stesso di un singolo cluster DB non globale. Per ulteriori informazioni, consulta Replica con Amazon Aurora.

Prima di apportare modifiche al cluster di database primario, assicurarsi di comprendere l'impatto sul database globale . Per ulteriori informazioni, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.

- Aurora Global Database attualmente non supporta lo `inaccessible-encryption-credentials-recoverable` stato in cui Amazon Aurora perde l'accesso alla chiave per AWS KMS il cluster DB. In questi casi, il cluster database crittografato entra nello stato terminale `inaccessible-encryption-credentials`. Per ulteriori informazioni su questi stati, consulta Visualizzazione dello stato del cluster del DB.
- Secrets Manager non supporta Aurora Global Database. Quando aggiungi una regione a un database globale, devi prima disattivare l'integrazione di Secrets Manager per l'istanza DB.
- I cluster DB basati su Aurora PostgreSQL che utilizzano Aurora Global Database presentano le seguenti limitazioni:
  - La gestione della cache del cluster non è supportata per i cluster DB secondari Aurora PostgreSQL che fanno parte dei database globali di Aurora.
  - Se il cluster DB primario del tuo database globale è basato su una replica di un'istanza Amazon RDS PostgreSQL, non puoi creare un cluster secondario. Non tentare di creare un secondario da quel cluster utilizzando l'operazione AWS Management Console, la o l'API AWS CLI. `CreateDBCluster` I tentativi di eseguire questa operazione scadono e il cluster secondario non viene creato.

Ti consigliamo di creare cluster DB secondari per i tuoi database globali utilizzando la stessa versione del motore Aurora DB del primario. Per ulteriori informazioni, consulta Creazione di un database globale Amazon Aurora.

Inizia a usare Amazon Aurora Global Database

Per iniziare a usare Aurora Global Database, devi prima decidere quale motore Aurora DB vuoi usare e in quale. Regioni AWS Solo versioni specifiche dei SQL motori di database Aurora My e SQL Aurora Postgre supportano Aurora Global Database. Regioni AWS Per l'elenco completo, consulta Regioni e motori DB supportati per i database globali Aurora.

È possibile creare un database globale Aurora in uno dei seguenti modi:

- Crea un nuovo Aurora Global Database con nuovi cluster Aurora DB e istanze Aurora DB: puoi farlo seguendo la procedura riportata di seguito. Creazione di un database globale Amazon Aurora

Dopo aver creato il cluster Aurora DB primario, aggiungi il cluster secondario Regione AWS seguendo i passaggi indicati in. Aggiungere un file Regione AWS a un database globale di Amazon Aurora

- Utilizza un cluster Aurora DB esistente che supporti la funzionalità Aurora Global Database e aggiungine uno Regione AWS : puoi farlo solo se il cluster Aurora DB esistente utilizza una versione del motore DB compatibile a livello globale.

  Verifica se puoi scegliere Aggiungi regione per l'azione AWS Management Console quando è selezionato il cluster Aurora DB. Se è possibile, è possibile utilizzare tale cluster database Aurora per il cluster Aurora globale. Per ulteriori informazioni, consulta Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Prima di creare un database globale Aurora, ti consigliamo di comprendere tutti i requisiti di configurazione.

Argomenti

- Requisiti di configurazione di un database globale Amazon Aurora
- Creazione di un database globale Amazon Aurora
- Aggiungere un file Regione AWS a un database globale di Amazon Aurora
- Creazione di un cluster database Aurora headless in una regione secondaria
- Creazione di un database globale Amazon Aurora da uno snapshot di Aurora o Amazon RDS

Requisiti di configurazione di un database globale Amazon Aurora

Prima di iniziare a lavorare con il database globale, pianifica i nomi del cluster e delle istanze, AWS le regioni, il numero di istanze e le classi di istanze che intendi utilizzare. Assicurati che le tue scelte siano conformi ai seguenti requisiti di configurazione.

Un database globale Aurora si estende su almeno due. Regioni AWS Il primario Regione AWS supporta un cluster Aurora DB con un'istanza Writer Aurora DB. Un secondario Regione AWS esegue un cluster Aurora DB di sola lettura composto interamente da repliche Aurora. Regione AWS È richiesto almeno un database secondario, ma un database globale Aurora può avere fino a 10 secondari. Regioni AWS Nella tabella sono elencati il numero massimo di cluster di database Aurora, di istanze database Aurora e repliche Aurora consentiti in un database globale Aurora.

Descrizione	Primario Regione AWS	Secondario Regioni AWS
Cluster di database Aurora	1	10 (massimo)
Istanze di scrittura	1	0
Istanze di sola lettura (repliche Aurora), per cluster di database Aurora	15 (massimo)	16 (totali)
Istanze di sola lettura (massimo consentito, dato il numero effettivo di Regioni secondarie)	15 - s	s = numero totale di unità secondarie Regioni AWS

I cluster di database Aurora che compongono un database globale Aurora hanno i seguenti requisiti specifici:

- Requisiti delle classi di istanza database – Un database globale Aurora richiede classi di istanza database ottimizzate per le applicazioni che fanno un uso intensivo della memoria. Per informazioni sulle classi di istanze DB ottimizzate per la memoria, vedereTipi di classi di istanza database. Si consiglia di utilizzare una classe di istanza db.r5 o superiore.
- Regione AWS requisiti: un database globale Aurora necessita di un cluster Aurora DB primario in uno Regione AWS e almeno un cluster Aurora DB secondario in un'altra regione. È possibile creare fino a 10 cluster Aurora DB secondari (di sola lettura), ognuno dei quali deve trovarsi in una regione diversa. In altre parole, nessun cluster di database Aurora in un database globale Aurora può trovarsi nella stessa Regione AWS.

  Per informazioni su quali combinazioni di motore Aurora DB, versione del motore e quali versioni del motore Regione AWS è possibile utilizzare con Aurora Global Database, vedere. Regioni e motori DB supportati per i database globali Aurora
- Requisiti di denominazione: i nomi che scegli per ciascuno dei tuoi cluster Aurora DB devono essere unici, in tutti i casi. Regioni AWS Non è possibile utilizzare lo stesso nome per cluster di database Aurora diversi anche se si trovano in regioni diverse.
- Requisiti di capacità per Aurora Serverless v2: per un database globale conAurora Serverless v2, la capacità minima consigliata per il cluster DB nel cluster primario Regione AWS è 8. ACUs

Prima di poter seguire le procedure descritte in questa sezione, è necessario disporre di un Account AWS. Completare le attività di configurazione per lavorare con Amazon Aurora. Per ulteriori informazioni, consulta Configurazione dell'ambiente per Amazon Aurora. È inoltre necessario completare altre fasi preliminari per la creazione di un cluster di database Aurora. Per ulteriori informazioni, vedi Creazione di un cluster database Amazon Aurora.

Quando sei pronto per configurare il tuo database globale, consulta la procedura Creazione di un database globale Amazon Aurora per creare tutte le risorse necessarie. Puoi anche seguire la procedura descritta Aggiungere un file Regione AWS a un database globale di Amazon Aurora per creare un database globale utilizzando un cluster Aurora esistente come cluster primario.

Creazione di un database globale Amazon Aurora

Per creare un database globale Aurora e le relative risorse associate utilizzando il AWS Management Console, il o AWS CLI RDSAPI, utilizzare i passaggi seguenti.

> **Note**
>
> Se disponi di un cluster Aurora DB esistente che esegue un motore di database Aurora compatibile a livello globale, puoi utilizzare una forma più breve di questa procedura. In tal caso, puoi aggiungerne un altro Regione AWS al cluster DB esistente per creare il tuo database globale Aurora. A questo proposito, consulta Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Console

I passaggi per la creazione di un database globale Aurora iniziano con l'accesso a un database Regione AWS che supporta la funzionalità di database globale Aurora. Per un elenco completo, consulta Regioni e motori DB supportati per i database globali Aurora.

Uno dei seguenti passaggi consiste nella scelta di un cloud privato virtuale (VPC) basato su Amazon VPC per il cluster Aurora DB. Per utilizzare il tuoVPC, ti consigliamo di crearlo in anticipo in modo che sia disponibile a tua scelta. Allo stesso tempo, crea tutte le sottoreti correlate e, in base alle esigenze, un gruppo di sottoreti e un gruppo di sicurezza. Per scoprire come, consulta Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 .

Per informazioni generali sulla creazione di un cluster di database Aurora, consulta Creazione di un cluster database Amazon Aurora.

Per creare un database globale Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Create database (Crea database). Nella pagina Crea database completa le seguenti operazioni:
   - Scegli Standard create (Creazione standard) per il metodo di creazione del database. (Non selezionare Easy Create (Creazione rapida)).
   - `Engine type`Nella sezione Opzioni motore, scegli il tipo di motore applicabile, Aurora (My SQL Compatible) o Aurora (Postgre Compatible). SQL
3. Continua a creare il database globale Aurora utilizzando i passaggi descritti nelle procedure riportate di seguito.

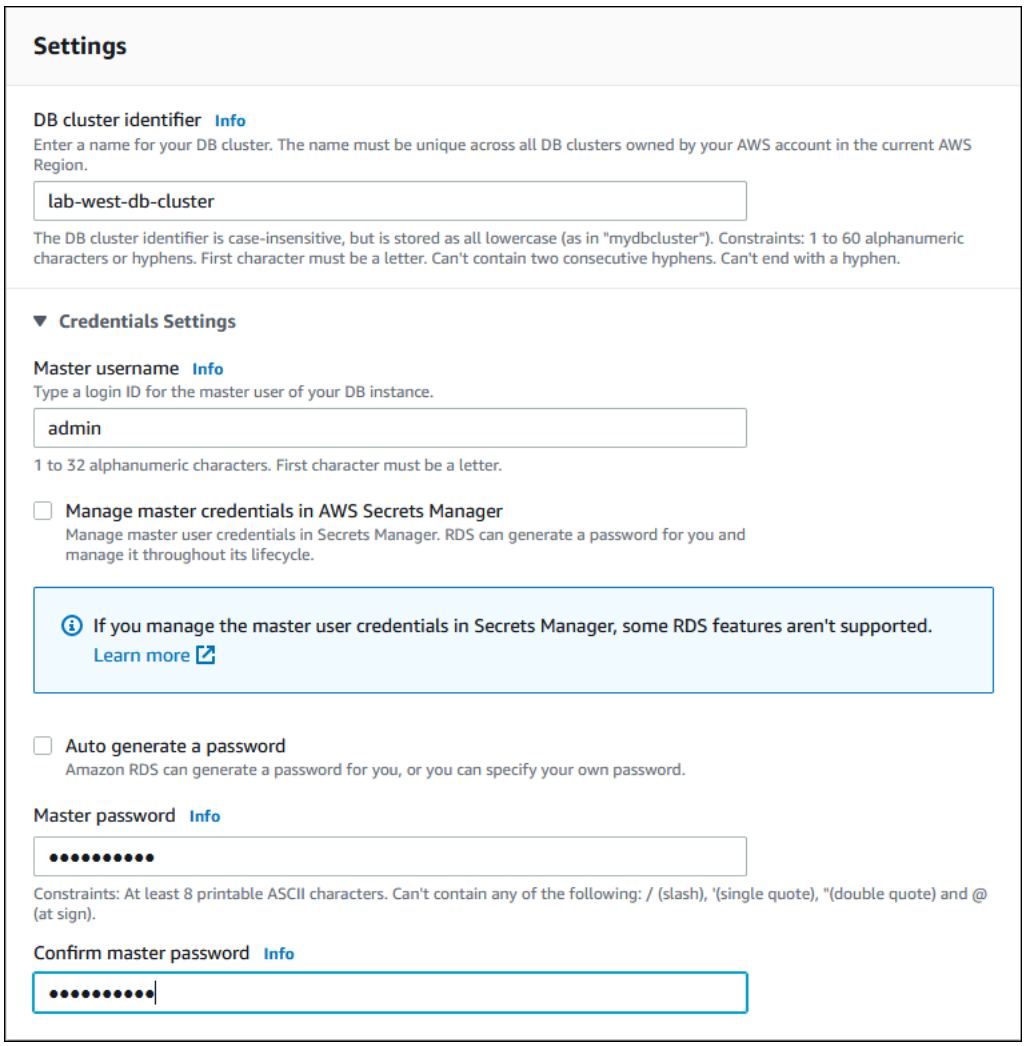
Creazione di un database globale con Aurora My SQL

I passaggi seguenti si applicano a tutte le versioni di Aurora My. SQL

Per creare un database globale Aurora utilizzando Aurora My SQL

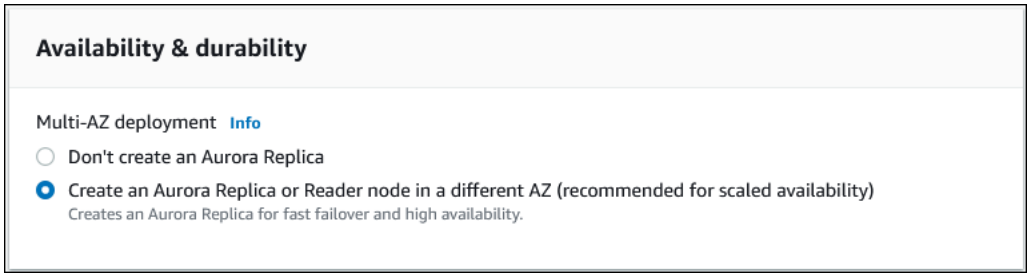
Completare la pagina Create database (Crea database).

1. Per Opzioni motore, scegli quanto segue:
   - Per la versione Engine, scegli la versione di Aurora My SQL che desideri utilizzare per il tuo database globale Aurora.
2. Per Templates (Modelli), scegliere Production (Produzione). Oppure puoi scegliere Dev/Test if appropriate for your use case. Don't use Dev/Test negli ambienti di produzione.
3. In Settings (Impostazioni), eseguire la seguente operazione:
   a. Immettere un nome specifico per l'identificatore del cluster di database. Al termine della creazione del database globale Aurora, questo nome identificherà il cluster di database primario.
   b. Specifica la password per l'account utente `admin` per l'istanza database o lascia che Aurora ne generi una. Se scegli di generare automaticamente una password, apparirà l'opzione per copiare la password.



4. Per DB instance class (Classe dell'istanza database), scegli `db.r5.large` o un'altra classe dell'istanza database ottimizzata per la memoria. Si consiglia di utilizzare una classe di istanza db.r5 o superiore.

5. Per Disponibilità e durata, ti consigliamo di lasciare che Aurora crei autonomamente una replica Aurora in una zona di disponibilità differente. Se non crei subito una replica Aurora, sarà necessario farlo in un secondo momento.



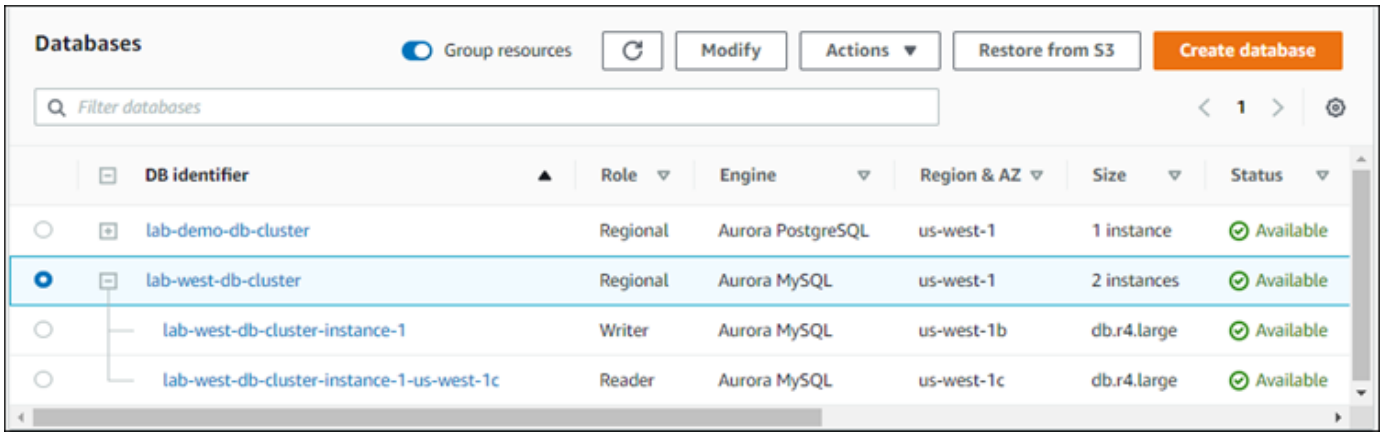
6. Per la connettività, scegli il cloud privato virtuale (VPC) basato su Amazon VPC che definisce l'ambiente di rete virtuale per questa istanza DB. È possibile scegliere i valori predefiniti per semplificare questa attività.
7. Completa le impostazioni di autenticazione del database . Per semplificare il processo, puoi scegliere l'autenticazione tramite password ora e configurare AWS Identity and Access Management (IAM) in un secondo momento.
8. In Additional configuration (Configurazione aggiuntiva), eseguire le operazioni seguenti:

   a. Immettere un nome per Initial database name (Nome del database iniziale) per creare l'istanza database Aurora primaria per il cluster. Questo è il nodo di scrittura per il cluster di database Aurora primario.

      Lasciare i valori predefiniti selezionati per il gruppo di parametri del cluster di database e il gruppo di parametri di database, a meno che non si disponga di gruppi di parametri personalizzati che si desidera utilizzare.

   b. Se selezionata, deseleziona l'opzione Abilita backtrack. I database globali Aurora non supportano il backtracking. Puoi accettare tutte le altre impostazioni predefinite per Configurazione aggiuntiva.
9. Scegliere Create database (Crea database).

Il completamento del processo di creazione dell'istanza database Aurora, della replica Aurora e del cluster di database Aurora può richiedere alcuni minuti ad Aurora. È possibile sapere quando il cluster database Aurora è pronto per l'uso come cluster database primario in un database globale Aurora in base al relativo stato. Quando è così, il suo stato e quello del writer e del nodo di replica sarà Disponibile, come illustrato di seguito.



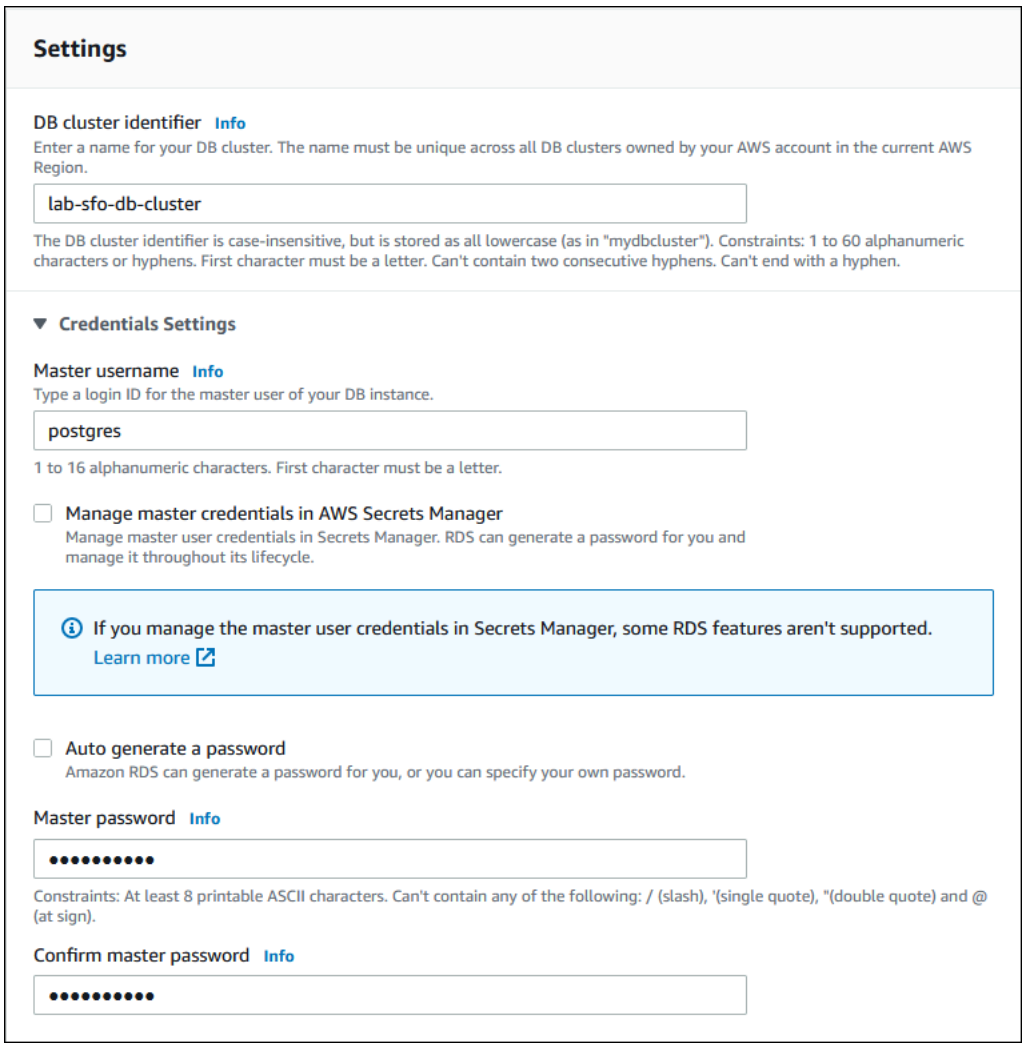
Quando il cluster di database primario è disponibile, crea il database globale Aurora aggiungendo un cluster secondario. A tale scopo, segui le fasi in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Creazione di un database globale utilizzando Aurora Postgre SQL

Per creare un database globale Aurora utilizzando Aurora Postgre SQL

Completare la pagina Create database (Crea database).

1. Per Opzioni motore, scegli quanto segue:
   - Per la versione Engine, scegli la versione di Aurora Postgre SQL che desideri utilizzare per il tuo database globale Aurora.
2. Per Templates (Modelli), scegliere Production (Produzione). Oppure puoi scegliere Dev/Test if appropriate. Don't use Dev/Test negli ambienti di produzione.
3. In Settings (Impostazioni), eseguire la seguente operazione:
   a. Immettere un nome specifico per l'identificatore del cluster di database. Al termine della creazione del database globale Aurora, questo nome identificherà il cluster di database primario.
   b. Specifica la password per l'account amministratore predefinito per il cluster database o lascia che Aurora ne generi una. Se si sceglie Genera automaticamente una password, apparirà l'opzione per copiare la password.



4. Per DB instance class (Classe dell'istanza database), scegli `db.r5.large` o un'altra classe dell'istanza database ottimizzata per la memoria. Si consiglia di utilizzare una classe di istanza db.r5 o superiore.
5. Per Availability & durability (Disponibilità e durata), consigliamo di lasciare che Aurora crei autonomamente una replica Aurora in un'altra zona di disponibilità. Se non crei subito una replica Aurora, sarà necessario farlo in un secondo momento.
6. Per la connettività, scegli il cloud privato virtuale (VPC) basato su Amazon VPC che definisce l'ambiente di rete virtuale per questa istanza DB. È possibile scegliere i valori predefiniti per semplificare questa attività.
7. (Facoltativo) Completa le impostazioni di Database authentication (Autenticazione database). L'autenticazione password è sempre abilitata. Per semplificare il processo, puoi saltare questa sezione e configurare IAM la password e l'autenticazione Kerberos in un secondo momento.
8. In Additional configuration (Configurazione aggiuntiva), eseguire le operazioni seguenti:

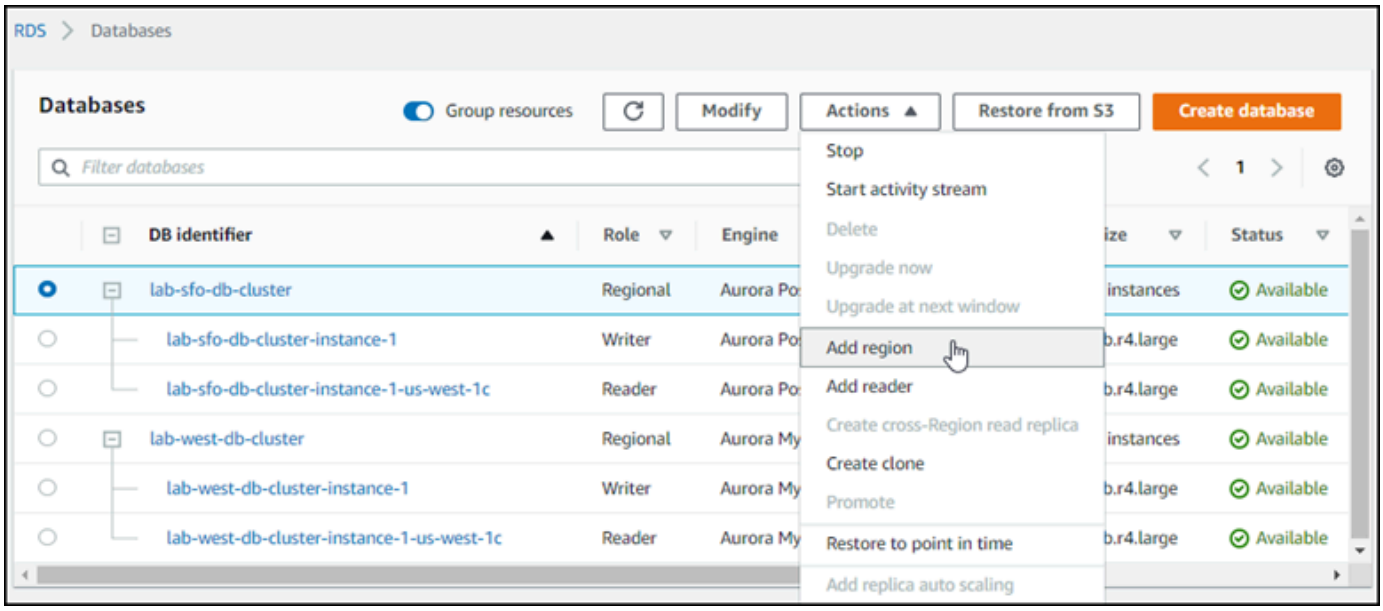
a. Immettere un nome per Initial database name (Nome del database iniziale) per creare l'istanza database Aurora primaria per il cluster. Questo è il nodo di scrittura per il cluster di database Aurora primario.

   Lasciare i valori predefiniti selezionati per il gruppo di parametri del cluster di database e il gruppo di parametri di database, a meno che non si disponga di gruppi di parametri personalizzati che si desidera utilizzare.

b. Accetta tutte le altre impostazioni predefinite per Configurazione aggiuntiva, ad esempio Crittografia, Esportazioni log e così via.

9. Scegliere Create database (Crea database).

   Il completamento del processo di creazione dell'istanza database Aurora, della replica Aurora e del cluster di database Aurora può richiedere alcuni minuti ad Aurora. Quando il cluster è pronto per l'uso, il cluster di database Aurora e i relativi nodi di scrittura e replica presentano tutti lo stato Disponibile. Questo diventa il cluster di database primario del database globale Aurora, dopo averne aggiunto uno secondario.



Dopo che il cluster di database primario è disponibile, creare uno o più cluster secondari seguendo le fasi in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

AWS CLI

I AWS CLI comandi nelle procedure seguenti eseguono le seguenti attività:

1. Crea un database globale Aurora, assegnandogli un nome e specificando il tipo di motore del database Aurora che si desidera utilizzare.
2. Creare un cluster di database Aurora per il database globale Aurora.
3. Creare l'istanza database Aurora per il cluster. Questo è il cluster database Aurora primario per il database globale.
4. Creare una seconda istanza database per il cluster di database Aurora. Questo è un reader per completare il cluster di database Aurora.
5. Creare un secondo cluster di database Aurora in un'altra regione e aggiungerlo al database globale Aurora, seguendo la procedura descritta in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Segui la procedura per il motore di database Aurora.

Creazione di un database globale con Aurora My SQL

Per creare un database globale Aurora utilizzando Aurora My SQL

1. Usa il `create-global-cluster` CLI comando, passando il nome e la Regione AWS versione del motore di database Aurora.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-mysql \
       --engine-version version # optional
   ```

   In Windows:

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-mysql ^
       --engine-version version # optional
   ```

   Questo crea un database globale Aurora "vuoto", con solo un nome (identificatore) e un motore di database Aurora. Prima che il database globale Aurora sia disponibile, potrebbero essere necessari alcuni minuti. Prima di passare al passaggio successivo, usa il `describe-global-clusters` CLI comando per vedere se è disponibile.

```
aws rds describe-global-clusters --region primary_region --global-cluster-
identifier global_database_id
```

Quando il database globale Aurora è disponibile, è possibile creare il relativo cluster di database Aurora primario.

2. Per creare un cluster Aurora DB primario, usa il `create-db-cluster` CLI comando. Includi il nome del database globale Aurora utilizzando il parametro `--global-cluster-identifier`.

   In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster \
  --region primary_region \
  --db-cluster-identifier primary_db_cluster_id \
  --master-username userid \
  --master-user-password password \
  --engine aurora-mysql \
  --engine-version version \
  --global-cluster-identifier global_database_id
```

   In Windows:

```
aws rds create-db-cluster ^
  --region primary_region ^
  --db-cluster-identifier primary_db_cluster_id ^
  --master-username userid ^
  --master-user-password password ^
  --engine aurora-mysql ^
  --engine-version version ^
  --global-cluster-identifier global_database_id
```

   Usa il `describe-db-clusters` AWS CLI comando per confermare che il cluster Aurora DB è pronto. Per individuare un cluster di database Aurora specifico, utilizzare il parametro `--db-cluster-identifier`. Oppure è possibile lasciare vuoto il nome del cluster di database Aurora nel comando per ottenere dettagli su tutti i cluster di database Aurora nella regione specificata.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

Quando per il cluster viene visualizzata la risposta `"Status": "available"`, significa che è pronto per l'uso.

3. Creare l'istanza database per il cluster di database Aurora primario. A questo scopo, utilizza il comando CLI di `create-db-instance`. Assegnare al comando il nome del cluster di database Aurora e specificare i dettagli di configurazione per l'istanza. Nel comando non è necessario passare i parametri `--master-username` e `--master-user-password`, perché li ottiene dal cluster di database Aurora.

   Per `--db-instance-class`, è possibile utilizzare solo quelli dalle classi ottimizzate per la memoria, ad esempio `db.r5.large`. Si consiglia di utilizzare una classe di istanza db.r5 o superiore. Per informazioni su queste classi, consultaTipi di classi di istanza database.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-instance \
     --db-cluster-identifier primary_db_cluster_id \
     --db-instance-class instance_class \
     --db-instance-identifier db_instance_id \
     --engine aurora-mysql \
     --engine-version version \
     --region primary_region
   ```

   In Windows:

   ```
   aws rds create-db-instance ^
     --db-cluster-identifier primary_db_cluster_id ^
     --db-instance-class instance_class ^
     --db-instance-identifier db_instance_id ^
     --engine aurora-mysql ^
     --engine-version version ^
     --region primary_region
   ```

   L'operazione `create-db-instance` potrebbe richiedere alcuni minuti per il completamento. Prima di continuare, controllare lo stato per verificare se l'istanza di database Aurora è disponibile.

   ```
   aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
   ```

Quando il comando restituisce lo stato di`available`, è possibile creare un'altra istanza Aurora DB per il cluster DB primario. Si tratta dell'istanza del reader (la replica Aurora) per il cluster di database Aurora.

4. Per creare un'altra istanza Aurora DB per il cluster, usa il `create-db-instance` CLI comando.

   In Linux, macOS, oppure Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier replica_db_instance_id \
  --engine aurora-mysql
```

   In Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier replica_db_instance_id ^
  --engine aurora-mysql
```

Quando l'istanza database è disponibile, la replica inizia dal nodo del writer al nodo di replica. Prima di continuare, verifica che l'istanza DB sia disponibile con il `describe-db-instances` CLI comando.

A questo punto, si dispone di un database globale Aurora con il relativo cluster di database Aurora primario contenente un'istanza database writer e una replica Aurora. A questo punto è possibile aggiungere un cluster di database Aurora di sola lettura in una regione diversa per completare il database globale Aurora. A tale scopo, segui la procedura in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Creazione di un database globale utilizzando Aurora Postgre SQL

Quando si creano oggetti Aurora per un database globale Aurora utilizzando i comandi seguenti, possono essere necessari alcuni minuti prima che ciascuno diventi disponibile. Dopo aver completato un determinato comando, si consiglia di controllare lo stato dell'oggetto Aurora specifico per assicurarsi che lo stato sia disponibile.

A questo scopo, utilizza il comando CLI di `describe-global-clusters`.

```
aws rds describe-global-clusters --region primary_region
    --global-cluster-identifier global_database_id
```

Per creare un database globale Aurora utilizzando Aurora Postgre SQL

1. Utilizza il comando `create-global-cluster` CLI.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-postgresql \
       --engine-version version # optional
   ```

   In Windows:

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-postgresql ^
       --engine-version version # optional
   ```

   Quando il database globale Aurora è disponibile, è possibile creare il relativo cluster di database Aurora primario.

2. Per creare un cluster Aurora DB primario, usa il `create-db-cluster` CLI comando. Includi il nome del database globale Aurora utilizzando il parametro `--global-cluster-identifier`.

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-postgresql \
     --engine-version version \
     --global-cluster-identifier global_database_id
   ```

   In Windows:

```
aws rds create-db-cluster ^
  --region primary_region ^
  --db-cluster-identifier primary_db_cluster_id ^
  --master-username userid ^
  --master-user-password password ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --global-cluster-identifier global_database_id
```

Verificare che il cluster di database Aurora sia pronto. Quando dal seguente comando viene visualizzata la risposta `"Status": "available"` per il cluster di database Aurora, è possibile continuare.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

3. Creare l'istanza database per il cluster di database Aurora primario. A questo scopo, utilizza il comando CLI di `create-db-instance`.

   Invia il nome del cluster di database Aurora con il parametro `--db-cluster-identifier`.

   Nel comando non è necessario passare i parametri `--master-username` e `--master-user-password`, perché li ottiene dal cluster di database Aurora.

   Per `--db-instance-class`, è possibile utilizzare solo quelli dalle classi ottimizzate per la memoria, ad esempio `db.r5.large`. Si consiglia di utilizzare una classe di istanza db.r5 o superiore. Per informazioni su queste classi, vedereTipi di classi di istanza database.

   In Linux, macOS, oppure Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-postgresql \
  --engine-version version \
  --region primary_region
```

   In Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --region primary_region
```

4. Prima di continuare, controllare lo stato dell'istanza database Aurora.

   ```
   aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
   ```

   Se la risposta mostra che lo stato dell'istanza Aurora DB è`available`, puoi creare un'altra istanza Aurora DB per il tuo cluster DB primario.

5. Per creare una replica Aurora per il cluster Aurora DB, utilizzare il comando. `create-db-instance` CLI

   In Linux, macOS, oppure Unix:

   ```
   aws rds create-db-instance \
     --db-cluster-identifier primary_db_cluster_id \
     --db-instance-class instance_class \
     --db-instance-identifier replica_db_instance_id \
     --engine aurora-postgresql
   ```

   In Windows:

   ```
   aws rds create-db-instance ^
     --db-cluster-identifier primary_db_cluster_id ^
     --db-instance-class instance_class ^
     --db-instance-identifier replica_db_instance_id ^
     --engine aurora-postgresql
   ```

Quando l'istanza database è disponibile, la replica inizia dal nodo del writer al nodo di replica. Prima di continuare, verifica che l'istanza DB sia disponibile con il `describe-db-instances` CLI comando.

Il database globale Aurora esiste, ma ha solo la relativa regione principale con un cluster di database Aurora costituito da un'istanza database di scrittura e una replica Aurora. A questo punto è possibile aggiungere un cluster di database Aurora di sola lettura in una regione diversa per completare il database globale Aurora. A tale scopo, segui la procedura in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

RDS API

Per creare un database globale Aurora con RDSAPI, esegui l'operazione. CreateGlobalCluster

Aggiungere un file Regione AWS a un database globale di Amazon Aurora

È possibile utilizzare la procedura seguente per aggiungere un cluster secondario aggiuntivo a un database globale esistente. È inoltre possibile creare un database globale da un cluster Aurora DB autonomo utilizzando questa procedura per aggiungere la prima AWS regione secondaria.

Un database globale Aurora necessita di almeno un cluster Aurora DB secondario in un cluster Aurora DB diverso Regione AWS dal cluster Aurora DB primario. Puoi collegare fino a 10 cluster DB secondari al tuo database globale Aurora. Ripetere la procedura seguente per ogni nuovo cluster DB secondario. Per ogni cluster di database secondario aggiunto al database globale Aurora, ridurre di un'unità il numero di repliche Aurora consentite nel cluster di database primario.

Ad esempio, se il database globale Aurora ha 10 regioni secondarie, il cluster DB primario può avere solo 5 (anziché 15) repliche Aurora. Per ulteriori informazioni, consulta Requisiti di configurazione di un database globale Amazon Aurora.

Il numero di repliche Aurora (istanze di lettura) nel cluster di database primario determina il numero di cluster database secondari che è possibile aggiungere. Il numero di istanze di lettura nel cluster database primario più il numero di cluster secondari potrebbe essere pari a 15. Ad esempio, se esistono 14 istanze di lettura nel cluster di database primario e un cluster secondario, non è possibile aggiungere un altro cluster secondario al database globale.

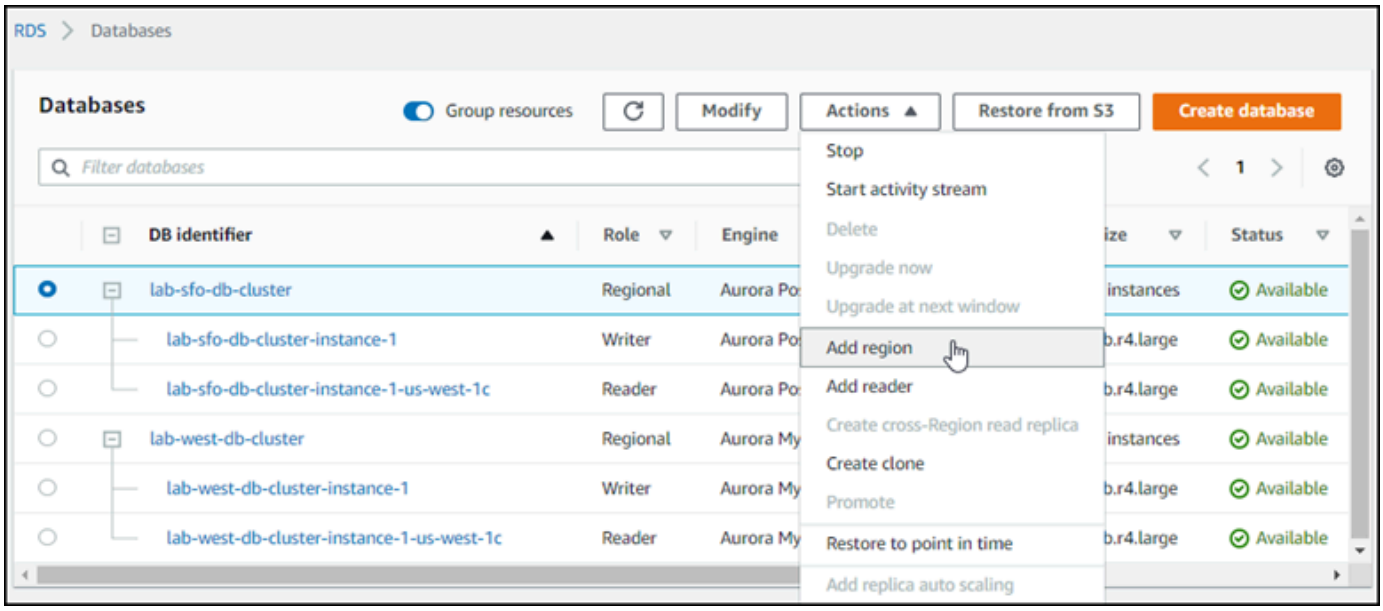
> **Note**
>
> Per Aurora MySQL versione 3, quando crei un cluster secondario, assicurati che il valore di `lower_case_table_names` corrisponda al valore nel cluster primario. Questa impostazione è un parametro del database che influisce sul modo in cui il server gestisce la distinzione tra maiuscole e minuscole. Per ulteriori informazioni sui parametri di database, vedi .
> Quando crei un cluster secondario, ti consigliamo di utilizzare la stessa versione del motore database per il primario e il secondario. Se necessario, aggiorna il primario in modo che

abbia la stessa versione del secondario. Per ulteriori informazioni, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni.

Console

Per aggiungere un file Regione AWS a un database globale di Aurora

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione di AWS Management Console, scegli Database.
3. Scegliere il database globale Aurora che richiede un cluster di database Aurora secondario. Assicurarsi che il cluster di database Aurora primario sia `Available`.
4. Per Azioni, scegli Aggiungi AWS regione.

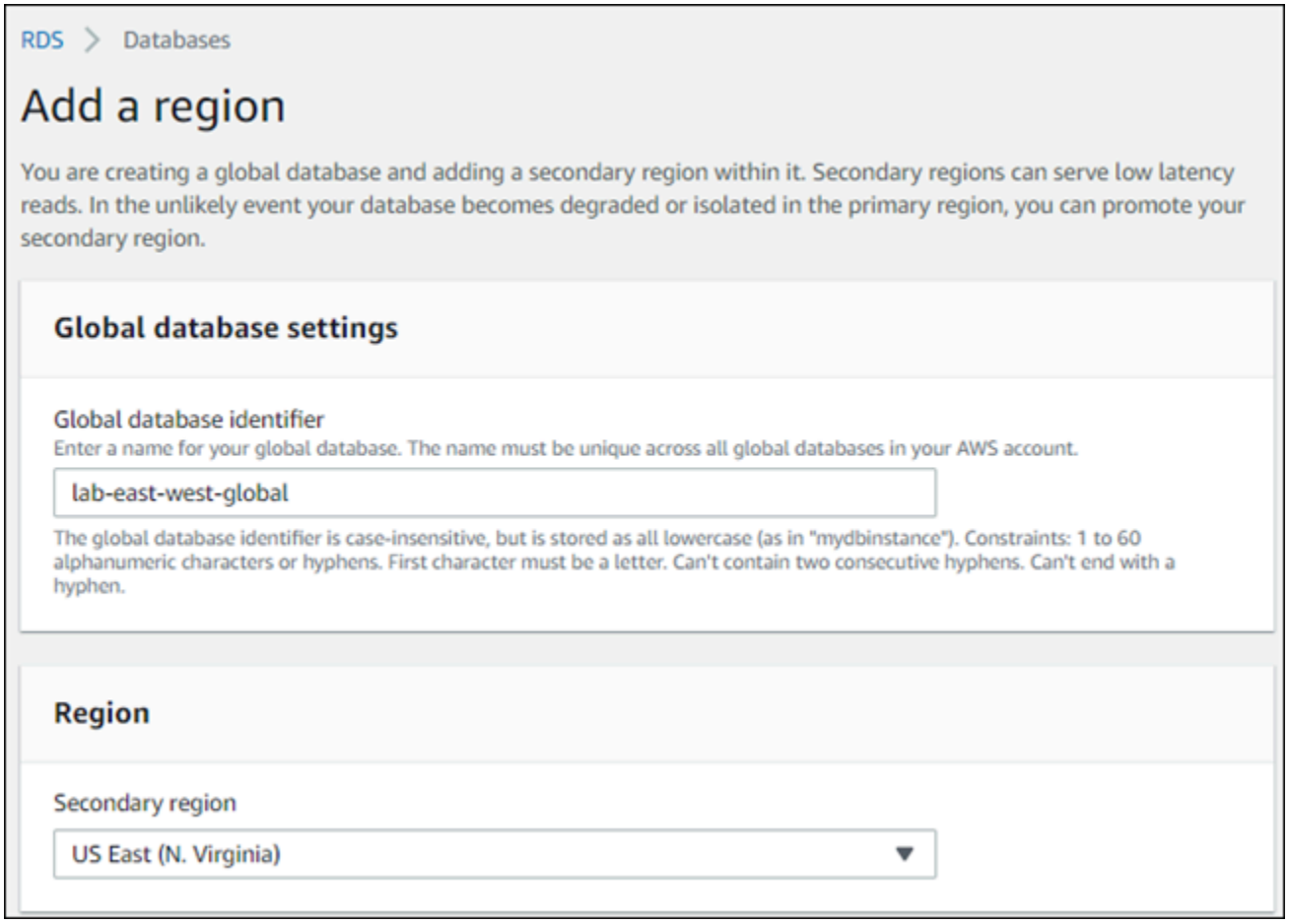


5. Nella pagina Aggiungi una regione, scegli la secondaria. Regione AWS

   Non puoi sceglierne uno Regione AWS che abbia già un cluster Aurora DB secondario per lo stesso database globale Aurora. Inoltre, non può essere la stessa regione del cluster database Aurora primario.

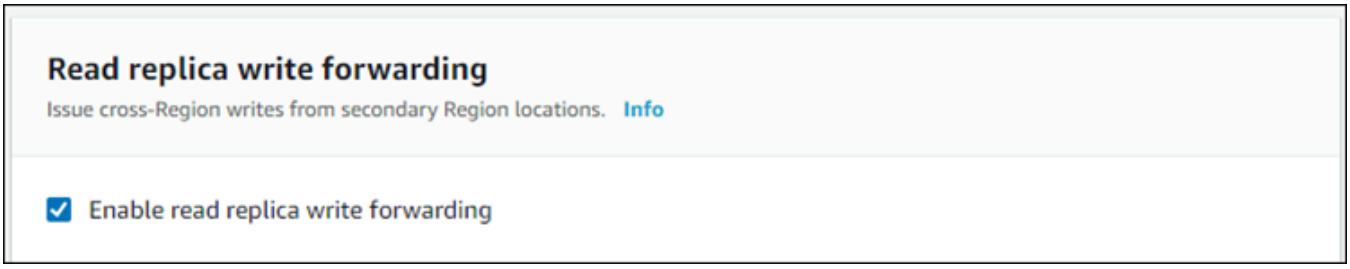
> **Note**
>
> I database globali PostgreSQL di Babelfish for Aurora funzionano nelle aree secondarie solo se i parametri che controllano le preferenze di Babelfish sono attivati in tali aree. Per

ulteriori informazioni, consulta Impostazioni del gruppo di parametri del cluster database per Babelfish



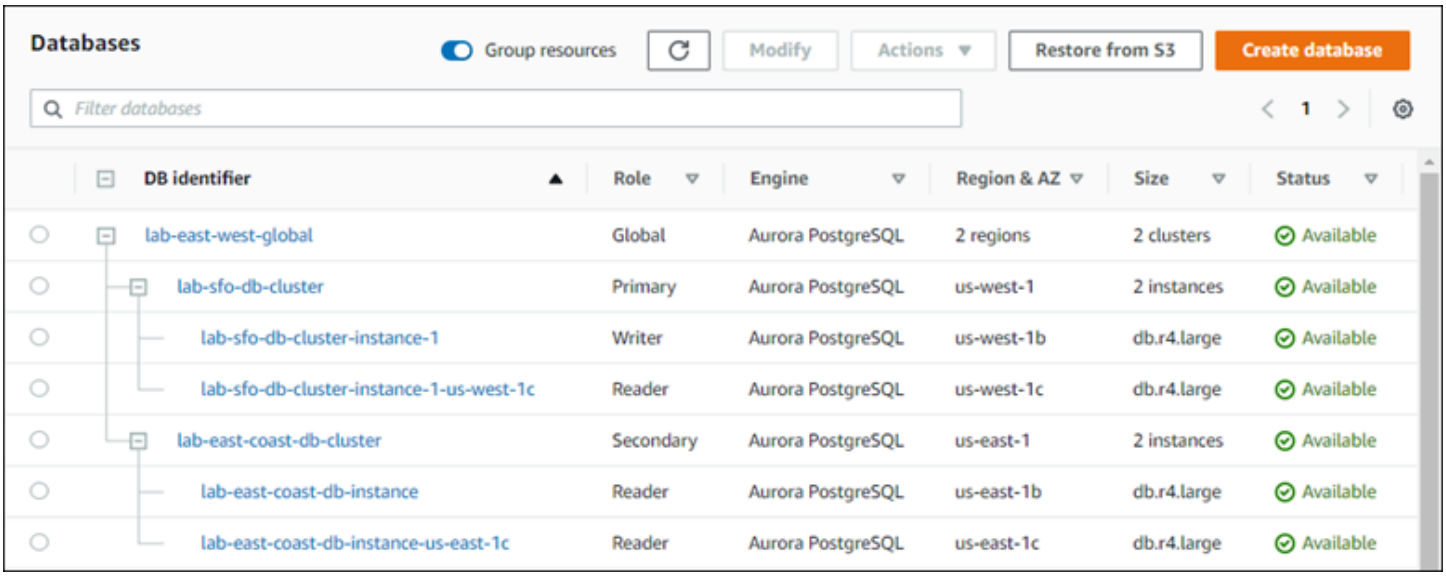
6. Completa i campi rimanenti per il cluster Aurora secondario nella nuova regione AWS . Queste sono le stesse opzioni di configurazione di qualsiasi istanza del cluster database Aurora, ad eccezione della seguente opzione solo per i database globali Aurora basati su Aurora MySQL–:

   - Abilita l'inoltro in scrittura di replica in lettura – Questa impostazione facoltativa consente ai cluster database secondari del database Aurora globale di inoltrare le operazioni di scrittura al cluster primario. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.



7. AWS Scegli Aggiungi regione.

Dopo aver aggiunto la regione al database globale di Aurora, puoi vederla nell'elenco dei database AWS Management Console come mostrato nello screenshot.



AWS CLI

Per aggiungere un database secondario Regione AWS a un database globale Aurora

Per aggiungere un cluster secondario al database globale utilizzando la CLI, è necessario disporre già dell'oggetto contenitore cluster globale. Se non hai già eseguito il `create-global-cluster` comando, consulta la procedura CLI in. Creazione di un database globale Amazon Aurora

1. Utilizzare il comando CLI `create-db-cluster` con il nome (`--global-cluster-identifier`) del database globale Aurora. Per gli altri parametri, effettuare le seguenti operazioni:
2. Perché`--region`, scegli una regione Regione AWS diversa da quella della tua regione principale Aurora.

3. Scegli i valori specifici per i parametri `--engine` e `--engine-version`. Questi valori sono gli stessi di quelli del cluster database Aurora primario nel database globale Aurora.
4. Per un cluster crittografato, specifica il tuo primario Regione AWS come quello `--source-region` per la crittografia.

Nell'esempio seguente viene creato un nuovo cluster di database Aurora e viene collegato a un database globale Aurora come cluster di database Aurora secondario di sola lettura. Nell'ultimo passaggio, viene aggiunta un'istanza database Aurora al nuovo cluster di database Aurora.

Per LinuxmacOS, oUnix:

```
aws rds --region secondary_region \
  create-db-cluster \
    --db-cluster-identifier secondary_cluster_id \
    --global-cluster-identifier global_database_id \
    --engine aurora-mysql | aurora-postgresql \
    --engine-version version

aws rds --region secondary_region \
  create-db-instance \
    --db-instance-class instance_class \
    --db-cluster-identifier secondary_cluster_id \
    --db-instance-identifier db_instance_id \
    --engine aurora-mysql | aurora-postgresql
```

Per Windows:

```
aws rds --region secondary_region ^
  create-db-cluster ^
    --db-cluster-identifier secondary_cluster_id ^
    --global-cluster-identifier global_database_id_id ^
    --engine aurora-mysql | aurora-postgresql ^
    --engine-version version

aws rds --region secondary_region ^
  create-db-instance ^
    --db-instance-class instance_class ^
    --db-cluster-identifier secondary_cluster_id ^
    --db-instance-identifier db_instance_id ^
    --engine aurora-mysql | aurora-postgresql
```

API RDS

Per aggiungerne uno nuovo Regione AWS a un database globale Aurora con l'API RDS, esegui l'operazione Create. DBCluster Specificare l'identificatore del database globale esistente utilizzando il parametro `GlobalClusterIdentifier`.

Creazione di un cluster database Aurora headless in una regione secondaria

Sebbene un database globale Aurora richieda almeno un cluster Aurora DB secondario in un cluster Regione AWS diverso da quello primario, è possibile utilizzare una configurazione headless per il cluster secondario. Un cluster database Aurora secondario headless è un cluster senza un'istanza database. Questo tipo di configurazione può ridurre le spese per un database globale Aurora. In un cluster database Aurora, il calcolo e l'archiviazione vengono disaccoppiati. Senza l'istanza database, non viene addebitato alcun costo per il calcolo, solo per lo storage. Se è configurato correttamente, il volume di archiviazione di un secondario headless viene mantenuto sincronizzato con il cluster database Aurora primario.

Puoi aggiungere il cluster secondario come si fa normalmente durante la creazione di un database globale Aurora. Se stai creando tutti i cluster nel database globale, segui la procedura riportata in. Creazione di un database globale Amazon Aurora Se disponi già di un cluster DB da utilizzare come cluster primario, segui la procedura riportata inAggiungere un file Regione AWS a un database globale di Amazon Aurora.

Dopo che il cluster Aurora DB primario inizia la replica sul secondario, si elimina l'istanza DB di sola lettura Aurora dal cluster Aurora DB secondario. Questo cluster secondario è ora considerato "headless" perché non dispone più dell'istanza DB. Anche senza alcuna istanza DB nel cluster secondario, Aurora mantiene il volume di archiviazione sincronizzato con il cluster Aurora DB primario.

> **Warning**
>
> Con Aurora PostgreSQL, per creare un cluster headless in un secondario Regione AWS, usa AWS CLI o RDS API per aggiungere il secondario. Regione AWS Salta il passaggio per creare l'istanza DB di lettura per il cluster secondario. Attualmente, la creazione di un cluster headless non è supportata nella console. RDS Per le API procedure CLI e le procedure da utilizzare, vedereAggiungere un file Regione AWS a un database globale di Amazon Aurora. Se il database globale utilizza una versione SQL del motore Aurora Postgre precedente alla 13.4, 12.8 o 11.13, la creazione di un'istanza DB di lettura in una regione secondaria e

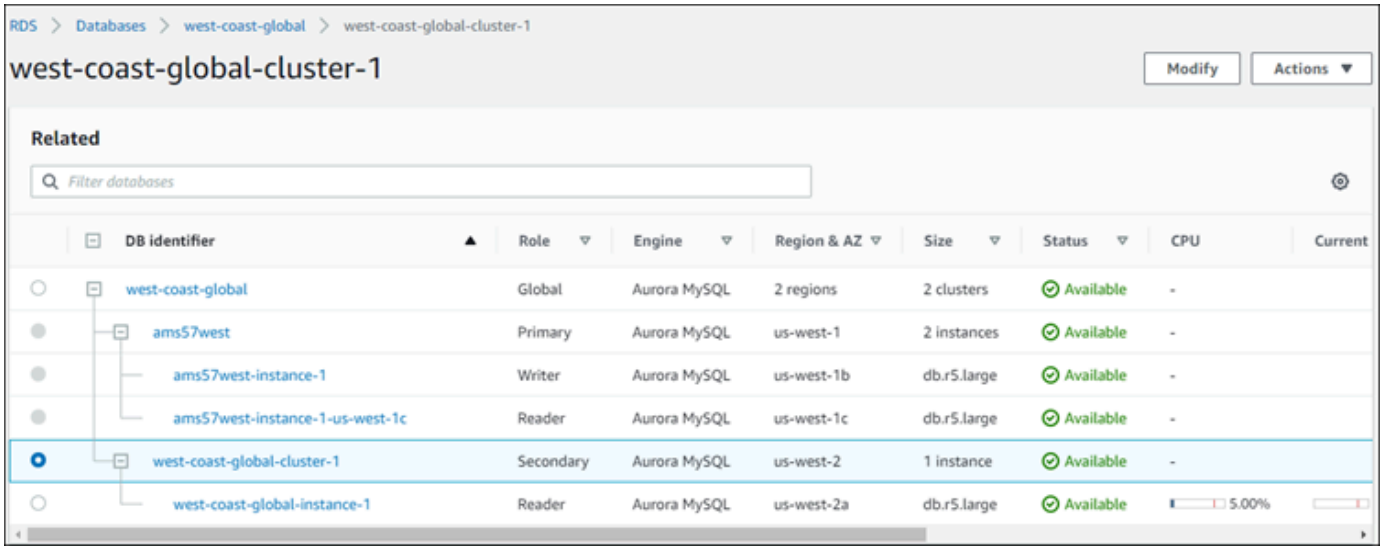
la successiva eliminazione potrebbe causare un problema di vuoto di Aurora SQL Postgre sull'istanza DB writer della regione principale. Se si verifica questo problema, riavviare l'istanza DB di scrittura della regione principale dopo aver eliminato l'istanza DB di lettura della regione secondaria.

Per aggiungere un cluster database Aurora secondario headless al database globale Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione di AWS Management Console, scegli Database.
3. Scegliere il database globale Aurora che richiede un cluster di database Aurora secondario. Assicurarsi che il cluster di database Aurora primario sia `Available`.
4. Per Azioni, scegli Aggiungi AWS regione.
5. Nella pagina Aggiungi una regione, scegli la secondaria Regione AWS.

   Non puoi sceglierne uno Regione AWS che abbia già un cluster Aurora DB secondario per lo stesso database globale Aurora. Inoltre, non può essere la stessa regione del cluster database Aurora primario.
6. Completa i campi rimanenti per il cluster Aurora secondario nel nuovo. Regione AWS Queste sono le stesse opzioni di configurazione di qualsiasi istanza del cluster database Aurora.

   Per un database globale Aurora SQL basato su Aurora My, ignora l'opzione Enable read Replica Write Forwarding. Questa opzione non ha alcuna funzione dopo aver eliminato l'istanza del lettore.
7. AWS Scegli Aggiungi regione. Dopo aver aggiunto la regione al database globale di Aurora, puoi vederla nell'elenco dei database AWS Management Console come mostrato nello screenshot.

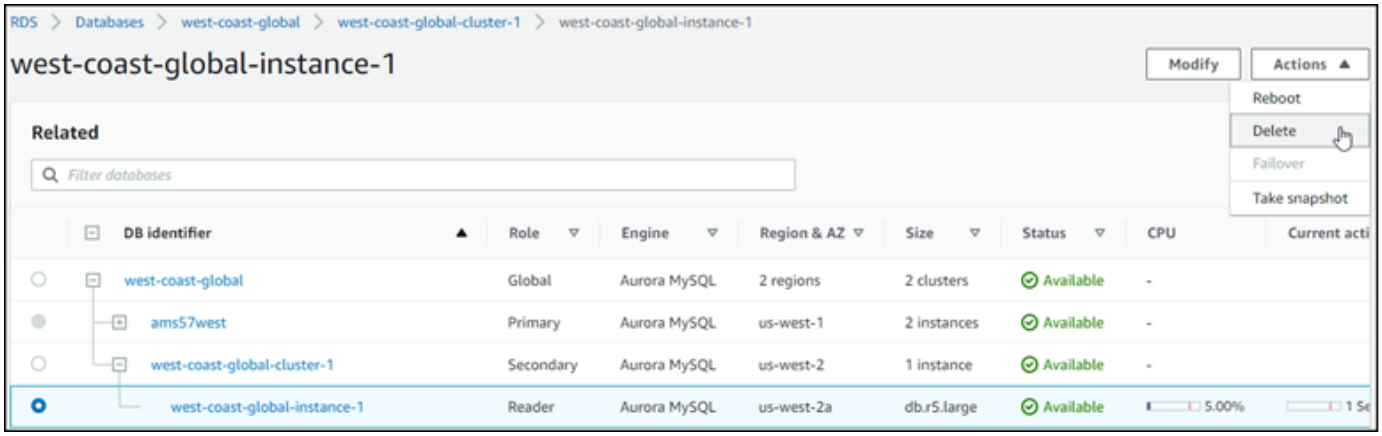


8. Controlla lo stato del cluster Aurora DB secondario e della relativa istanza di lettura prima di continuare, utilizzando AWS Management Console o il. AWS CLI Per esempio:

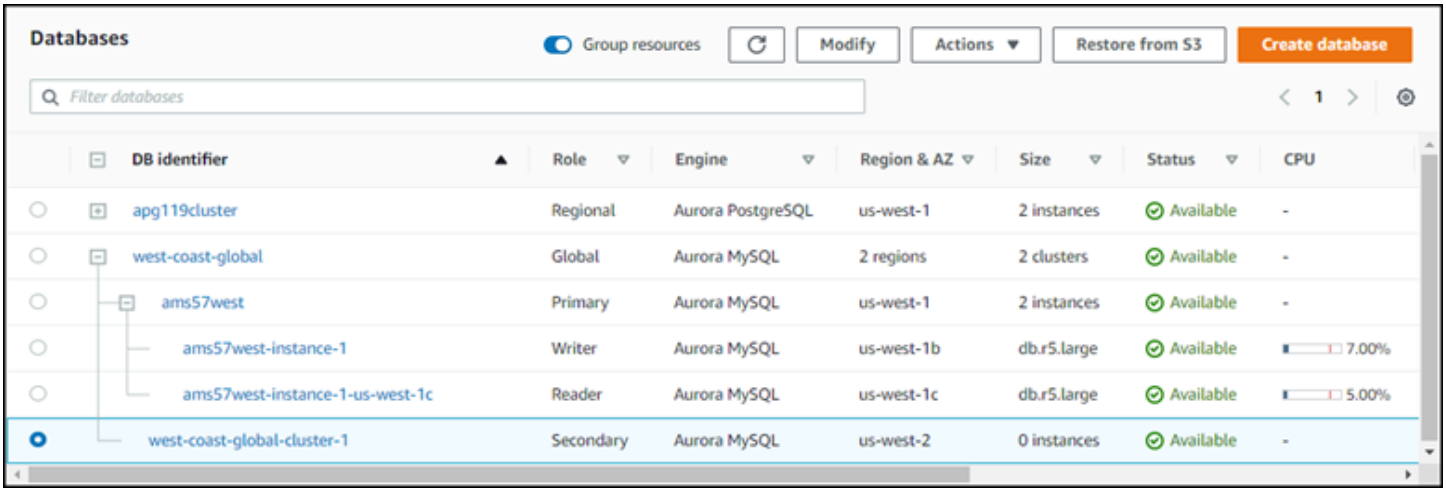
```
$ aws rds describe-db-clusters --db-cluster-identifier secondary-cluster-id --query
 '*[].[Status]' --output text
```

   Il passaggio dello stato di un cluster database Aurora secondario appena aggiunto da `creating` a `available` può richiedere alcuni minuti. Quando il cluster database Aurora è disponibile, è possibile eliminare l'istanza di lettura.

9. Seleziona l'istanza di lettura nel cluster database Aurora secondario, quindi scegli Elimina.



Dopo aver eliminato l'istanza di lettura, il cluster secondario rimane parte del database globale Aurora. Non ha alcuna istanza associata, come illustrato di seguito.



Puoi utilizzare questo cluster di database Aurora secondario headless per ripristinare manualmente il database globale Amazon Aurora da un'interruzione non pianificata nella Regione AWS principale se si verifica un'interruzione del servizio.

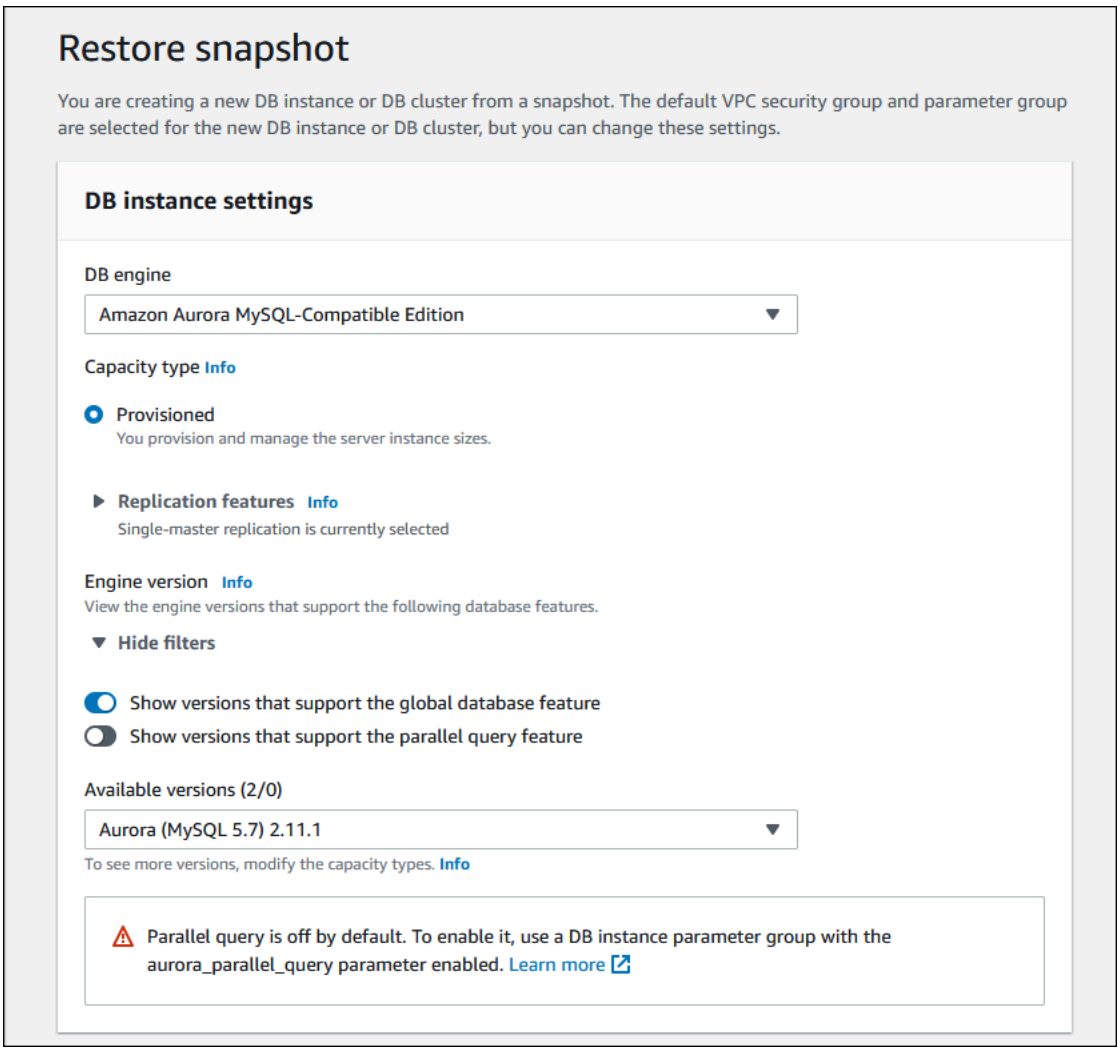
Creazione di un database globale Amazon Aurora da uno snapshot di Aurora o Amazon RDS

Puoi ripristinare uno snapshot di un cluster Aurora DB o da un'istanza RDS Amazon DB da utilizzare come punto di partenza per il tuo database globale Aurora. È possibile ripristinare uno snapshot e al contempo creare un nuovo cluster di database Aurora con provisioning. Quindi ne aggiungi un altro Regione AWS al cluster DB ripristinato, trasformandolo così in un database globale Aurora. Qualsiasi cluster di database Aurora creato utilizzando uno snapshot in questo modo diventa il cluster primario del database globale Aurora.

Lo snapshot utilizzato può essere preso da un cluster di database Aurora `provisioned` o `serverless`.

Durante il processo di ripristino, scegliere lo stesso tipo di motore di database dello snapshot. Ad esempio, si supponga di voler ripristinare un'istantanea creata da un Aurora Serverless Cluster DB che esegue Aurora Postgre. SQL In questo caso, si crea un cluster Aurora Postgre SQL DB utilizzando lo stesso motore e la stessa versione di Aurora DB.

Il cluster DB ripristinato assume il ruolo di cluster primario per il database globale Aurora quando ne aggiungi Regione AWS uno. Tutti i dati contenuti in questo cluster primario vengono replicati in tutti i cluster secondari aggiunti al database globale Aurora.



Gestione di un database globale Amazon Aurora

Puoi eseguire la maggior parte delle operazioni di gestione sui singoli cluster che costituiscono un database globale Aurora. Quando selezioni Raggruppa risorse correlate nella pagina Database della console, i cluster primario e secondario vengono visualizzati come raggruppati nel database globale associato. Per scoprire Regioni AWS dove sono in esecuzione i cluster DB di un database globale, il relativo motore e versione di Aurora DB e il relativo identificatore, utilizza la scheda Configurazione.

Il processo di failover del database tra regioni è disponibile solo per i database globali Aurora e non per un singolo cluster database Aurora. Per ulteriori informazioni, consulta Utilizzo dello switchover o del failover in Amazon Aurora Global Database.

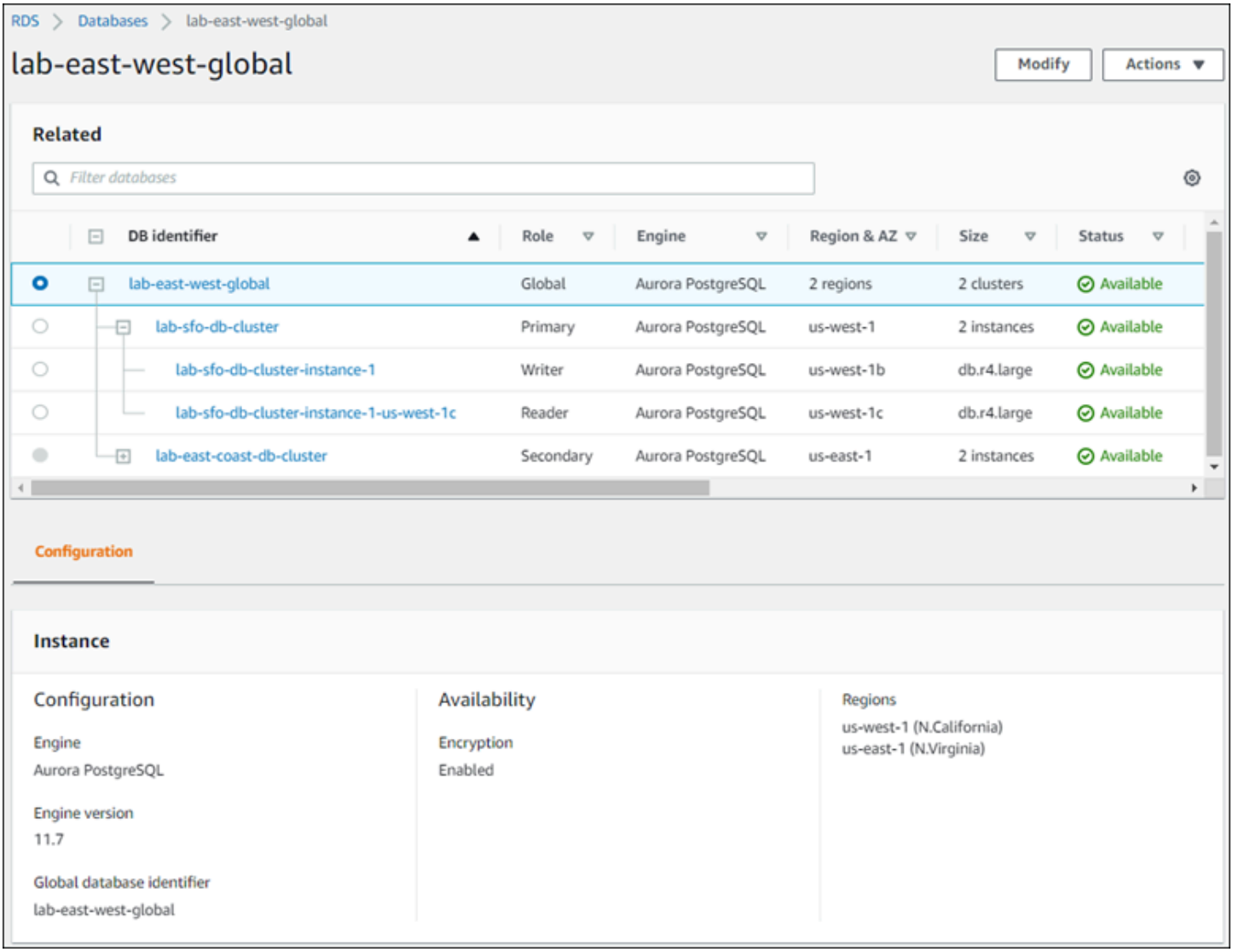
Per ripristinare un database globale Aurora da un'interruzione non pianificata nella relativa regione principale, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.

Argomenti

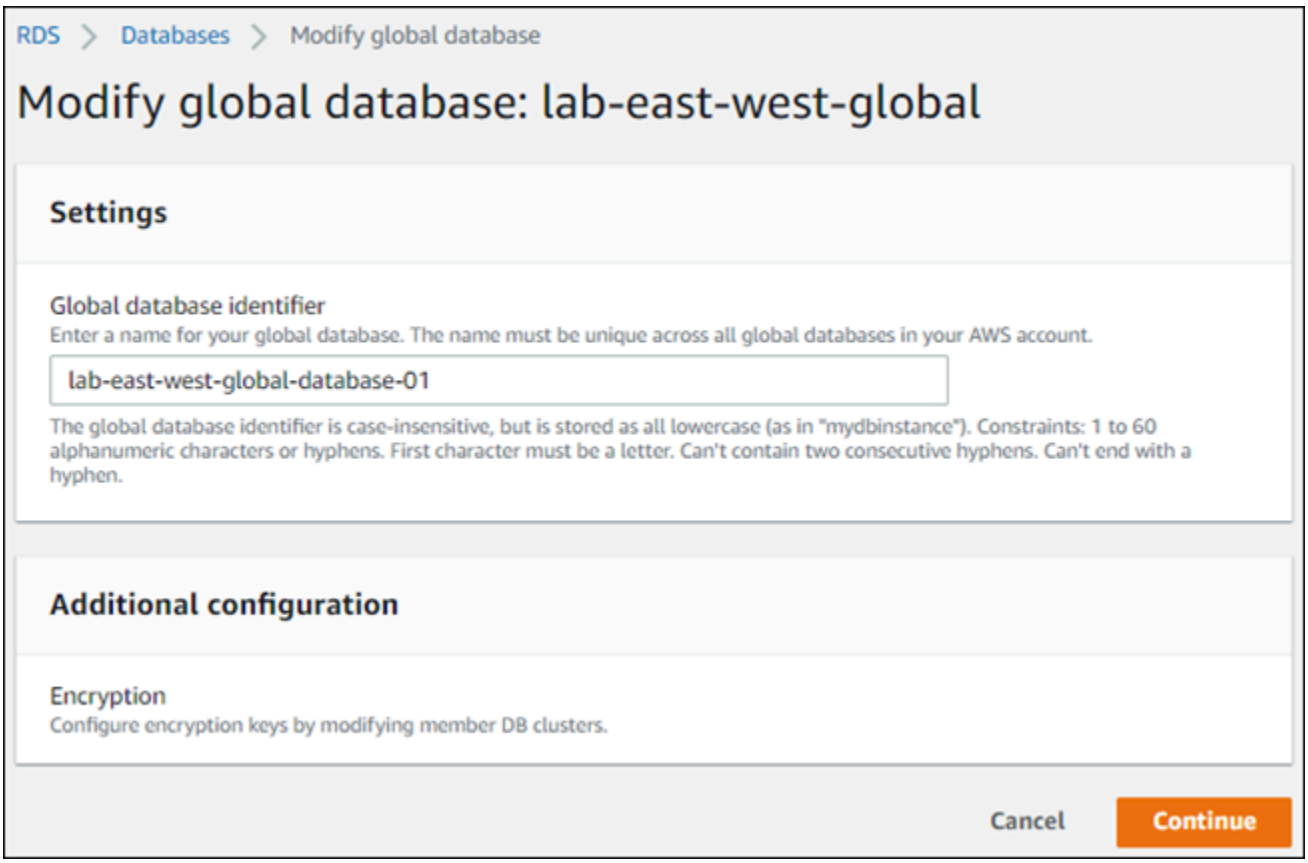
- Modifica di un database globale Amazon Aurora
- Modifica dei parametri per un database globale Aurora
- Rimozione di un cluster da un database globale Amazon Aurora
- Eliminazione di un database globale Amazon Aurora
- Etichettatura delle risorse del database globale di Amazon Aurora

Modifica di un database globale Amazon Aurora

La pagina Database nella pagina AWS Management Console elenca tutti i database globali Aurora, mostrando il cluster primario e i cluster secondari per ciascuno di essi. Il database globale Aurora dispone di impostazioni di configurazione proprie. In particolare, è stato Regioni AWS associato ai suoi cluster primari e secondari, come mostrato nella schermata seguente.



Quando si apportano modifiche al database globale Aurora, si ha la possibilità di annullare le modifiche, come mostrato nella schermata seguente.



Quando si sceglie Continue (Continua), si confermano le modifiche.

Modifica dei parametri per un database globale Aurora

È possibile configurare i gruppi di parametri del cluster di database Aurora in modo indipendente per ogni cluster Aurora all'interno del database globale Aurora. La maggior parte dei parametri funzionano come per altri tipi di cluster Aurora: Si consiglia di mantenere le impostazioni coerenti tra tutti i cluster di un database globale. In questo modo è possibile evitare modifiche impreviste del comportamento se si promuove un cluster secondario come primario.

Ad esempio, utilizzare le stesse impostazioni per fusi orari e set di caratteri per evitare comportamenti incoerenti se un cluster diverso diventa un cluster primario.

Le impostazioni di configurazione `aurora_enable_repl_bin_log_filtering` e `aurora_enable_replica_log_compression` non hanno effetto.

Rimozione di un cluster da un database globale Amazon Aurora

È possibile rimuovere i cluster di database Aurora dal database globale Aurora per diversi motivi. Ad esempio, è possibile rimuovere un cluster di database Aurora da un database globale Aurora

se il cluster primario viene danneggiato o isolato. Diventa quindi un cluster database Aurora con provisioning autonomo che potrebbe essere utilizzato per creare un nuovo database globale Aurora. Per ulteriori informazioni, vedi Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.

È anche possibile rimuovere cluster di database Aurora se si desidera eliminare un database globale Aurora che non è più necessario. Non è possibile eliminare il database globale Aurora fino a quando non si rimuove (scollega) tutti i cluster database Aurora associati, lasciando il primario per ultimo. Per ulteriori informazioni, consulta Eliminazione di un database globale Amazon Aurora.

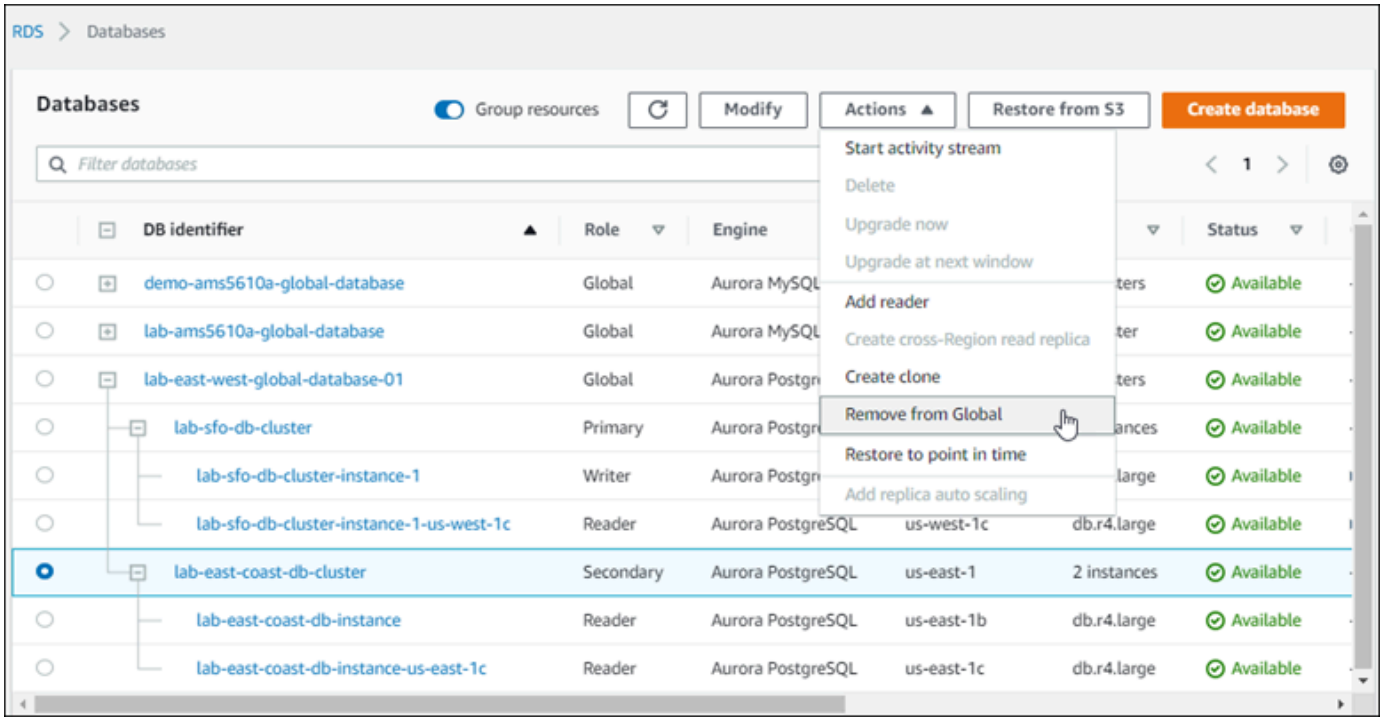
Quando un cluster database Aurora viene scollegato dal database globale Aurora, non è più sincronizzato con il primario. Diventa un cluster database Aurora con provisioning autonomo con funzionalità di lettura/scrittura complete.

È possibile rimuovere i cluster Aurora DB dal database globale Aurora utilizzando il AWS Management Console, il, o il. AWS CLI RDS API

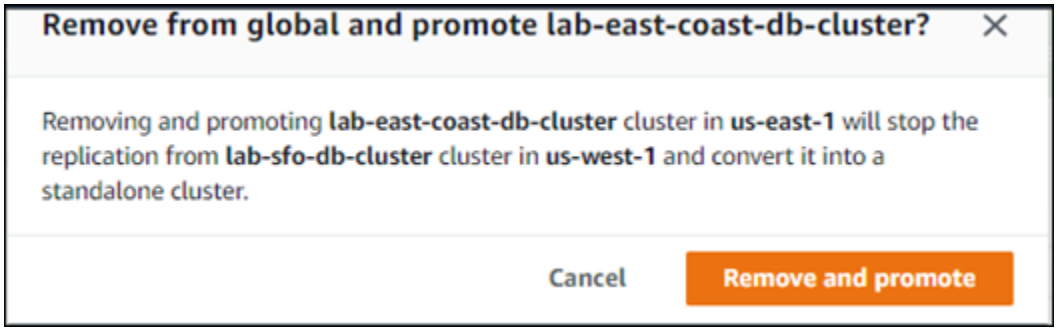
Console

Per rimuovere un cluster Aurora da un Aurora Global Database

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere il cluster nella pagina Databases (Database) .
3. Per Actions (Operazioni), scegliere Remove from Global (Rimuovi da Global).

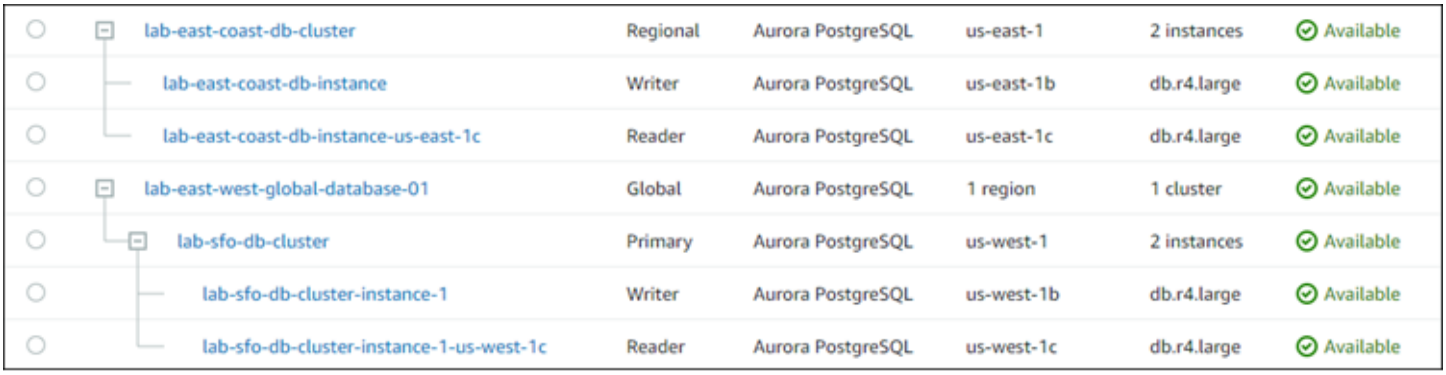


Viene visualizzato un messaggio di conferma che si desidera scollegare il cluster secondario dal database globale Aurora.



4. Scegliere Remove and promote (Rimuovi e promuovi) per rimuovere il cluster dal database globale.

Il cluster di database Aurora non è più utilizzato come secondario nel database globale Aurora e non viene più sincronizzato con il cluster di database primario. Si tratta di un cluster di database Aurora autonomo con funzionalità di lettura/scrittura complete.



Dopo aver rimosso o eliminato tutti i cluster secondari, puoi rimuovere il cluster primario nello stesso modo. Non è possibile scollegare (rimuovere) il cluster di database Aurora primario da un database globale Aurora fino a quando non si rimuovono tutti i cluster secondari.

Il database globale Aurora potrebbe rimanere nell'elenco Database, con zero Regioni e. AZs Laddove si desidera smettere di utilizzare questo database globale Aurora, è possibile eliminarlo. Per ulteriori informazioni, consulta Eliminazione di un database globale Amazon Aurora.

AWS CLI

Per rimuovere un cluster Aurora da un database globale Aurora, esegui il remove-from-global-clusterCLIcomando con i seguenti parametri:

- `--global-cluster-identifier` – Il nome (identificatore) del database globale Aurora.
- `--db-cluster-identifier` – Il nome di ogni cluster di database Aurora da rimuovere dal database globale Aurora. Prima di rimuovere il cluster primario, rimuovere tutti i cluster di database Aurora secondari.

Negli esempi di seguito, da un database globale Aurora viene prima rimosso un cluster secondario e poi il cluster primario.

In Linux, macOS, oppure Unix:

```
aws rds --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_database_id

aws rds --region primary_region \
  remove-from-global-cluster \
```

```
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_database_id
```

Ripeti il `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* comando per ogni elemento secondario Regione AWS nel database globale di Aurora.

In Windows:

```
aws rds --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_database_id

aws rds --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_database_id
```

Ripeti il `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* comando per ogni elemento secondario Regione AWS nel database globale di Aurora.

RDS API

Per rimuovere un cluster Aurora da un database globale Aurora con RDSAPI, esegui l'azione. RemoveFromGlobalCluster

Eliminazione di un database globale Amazon Aurora

Poiché un database globale Aurora contiene in genere dati critici per l'azienda, non è possibile eliminare il database globale e i cluster associati in un singolo passaggio. Per eliminare un database globale Aurora, completa le seguenti operazioni:

- Rimuovere tutti i cluster di database secondari dal database globale Aurora. Ogni cluster diventa un cluster di database Aurora autonomo. Per scoprire come, consulta Rimozione di un cluster da un database globale Amazon Aurora.
- Da ogni cluster di database Aurora autonomo, eliminare tutte le repliche Aurora.
- Rimuovere il cluster di database primario dal database globale Aurora. Questo diventa un cluster di database Aurora autonomo.

- Dal cluster di database primario Aurora, per prima cosa elimina tutte le repliche Aurora, quindi elimina l'istanza del database di scrittura.

L'eliminazione dell'istanza di scrittura dal nuovo cluster di database Aurora autonomo generalmente rimuove anche il cluster di database Aurora e il database globale Aurora.

Per ulteriori informazioni generali, consulta Eliminazione di un'istanza database da un cluster database Aurora.

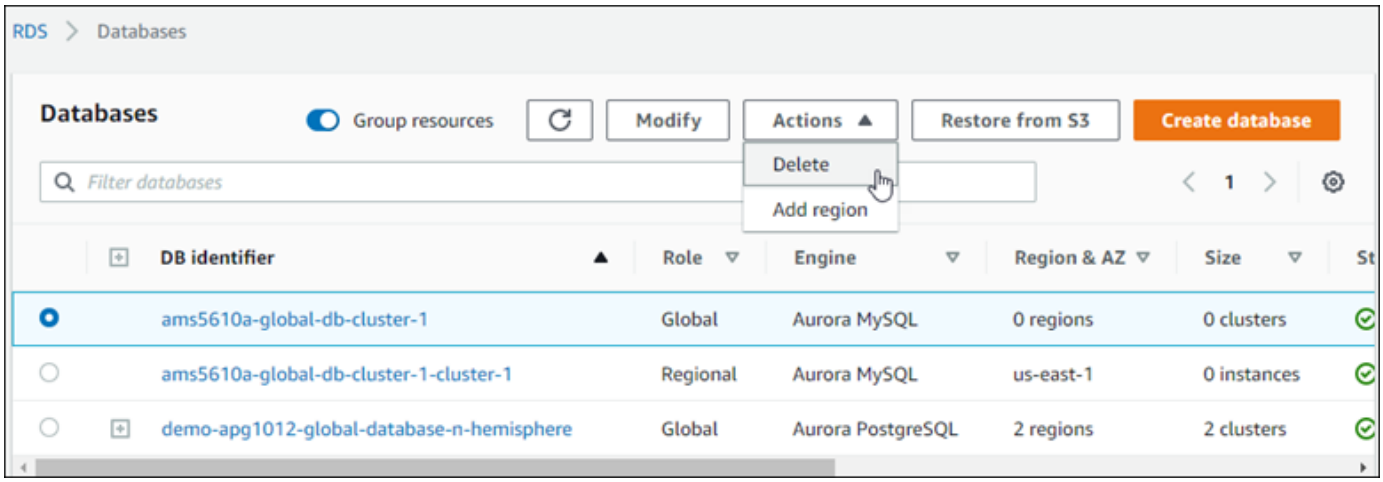
Per eliminare un database globale Aurora, è possibile utilizzare il AWS Management Console AWS CLI, o il. RDS API

Console

Per eliminare un database globale Aurora

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Database e trovare il database globale Aurora che si desidera eliminare nell'elenco.
3. Confermare che tutti gli altri cluster vengono rimossi dal database globale Aurora. Il database globale Aurora dovrebbe mostrare 0 regioni AZs e una dimensione di 0 cluster.

   Se il database globale Aurora contiene cluster di database Aurora, non è possibile eliminarlo. Se necessario, scollegare i cluster di database Aurora primario e secondario dal database globale Aurora. Per ulteriori informazioni, consulta Rimozione di un cluster da un database globale Amazon Aurora.
4. Scegli il database globale Aurora nell'elenco, quindi scegli Elimina nel menu Operazioni.



AWS CLI

Per eliminare un database globale Aurora, esegui il delete-global-clusterCLIcomando con il nome Regione AWS e l'identificatore globale del database Aurora, come illustrato nell'esempio seguente.

In Linux, macOS, oppure Unix:

```
aws rds --region primary_region delete-global-cluster \
   --global-cluster-identifier global_database_id
```

In Windows:

```
aws rds --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_database_id
```

RDS API

Per eliminare un cluster che fa parte di un database globale Aurora, esegui l'DeleteGlobalClusterAPIoperazione.

Etichettatura delle risorse del database globale di Amazon Aurora

Con la funzionalità Aurora Global Database, puoi applicare RDS tag a risorse a diversi livelli all'interno di un database globale. Se non conosci il modo in cui i tag vengono utilizzati con AWS le risorse Aurora, consulta Etichettatura di Amazon Aurora e risorse Amazon RDS prima di applicare i tag all'interno del tuo database globale.

> **Note**
>
> Poiché AWS elabora i dati dei tag come parte dei suoi meccanismi di rendicontazione dei costi, non includete dati sensibili o informazioni di identificazione personale (PII) nei nomi o nei valori dei tag.

È possibile applicare tag ai seguenti tipi di risorse all'interno di un database globale:

- L'oggetto contenitore per l'intero database globale. Questa risorsa è nota come cluster globale.

  Dopo aver creato il cluster globale eseguendo un'operazione Aggiungi AWS regione nella console, puoi aggiungere tag utilizzando la pagina dei dettagli per il cluster globale. Nella scheda Tag

all'interno della pagina dei dettagli del cluster globale, puoi aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag.

Con AWS CLI and RDSAPI, puoi aggiungere tag al cluster globale nello stesso momento in cui lo crei. È inoltre possibile aggiungere, rimuovere o modificare i tag per un cluster globale esistente.

- Il cluster primario. Qui si utilizzano le stesse procedure per lavorare con i tag utilizzate per i cluster Aurora autonomi. È possibile configurare i tag prima di trasformare il cluster Aurora originale in un database globale. Puoi anche aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag nella scheda Tag all'interno della pagina dei dettagli del cluster DB.
- Qualsiasi cluster secondario. Qui si utilizzano le stesse procedure per lavorare con i tag utilizzate per i cluster Aurora autonomi. Puoi configurare i tag contemporaneamente alla creazione di un cluster Aurora secondario utilizzando l'azione Aggiungi AWS regione nella console. Puoi anche aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag nella scheda Tag all'interno della pagina dei dettagli del cluster DB.
- Singole istanze DB all'interno dei cluster primari o secondari. Qui si utilizzano le stesse procedure per lavorare con i tag come per le istanze Aurora o RDS DB. È possibile configurare i tag contemporaneamente all'aggiunta di una nuova istanza DB al cluster Aurora utilizzando l'azione Aggiungi lettore nella console. Puoi anche aggiungere, rimuovere o modificare i tag e i relativi valori associati scegliendo Gestisci tag nella scheda Tag all'interno della pagina dei dettagli dell'istanza DB.

Ecco alcuni esempi dei tipi di tag che potresti assegnare all'interno di un database globale:

- È possibile aggiungere tag al cluster globale per registrare informazioni generali sull'applicazione, ad esempio identificatori anonimi che rappresentano proprietari e contatti all'interno dell'organizzazione. È possibile utilizzare i tag per rappresentare le proprietà dell'applicazione che utilizza il database globale.
- È possibile aggiungere tag al cluster primario e ai cluster secondari per tenere traccia dei costi dell'applicazione a livello di AWS regione. Per informazioni dettagliate su questa procedura, vedereCome funziona la AWS fatturazione con i tag in Amazon RDS.
- È possibile aggiungere tag a istanze DB specifiche con i cluster Aurora per indicarne lo scopo speciale. Ad esempio, all'interno del cluster primario, potresti avere un'istanza di lettura con una priorità di failover bassa che viene utilizzata esclusivamente per la generazione di report. Un tag può distinguere questa istanza DB per scopi speciali da altre istanze dedicate all'elevata disponibilità all'interno del cluster primario.

- È possibile utilizzare i tag a tutti i livelli delle risorse globali del database per controllare l'accesso tramite policy. IAM Per ulteriori informazioni, vedere Controlling access to AWS resources nella AWS Identity and Access Management User Guide.

  > **Tip**
  >
  > In AWS Management Console, si aggiungono tag al contenitore del cluster globale in un passaggio separato dopo averlo creato. Se si desidera evitare qualsiasi intervallo di tempo in cui il cluster globale esiste senza tag di controllo degli accessi, è possibile applicare i tag durante l'`CreateGlobalCluster`operazione creando quella risorsa tramite AWS CLI RDSAPI, o un AWS CloudFormation modello.

- È possibile utilizzare i tag a livello di cluster o per il cluster globale per registrare informazioni sul controllo della qualità e sui test dell'applicazione. Ad esempio, è possibile specificare un tag su un cluster DB per registrare l'ultima volta che è stato eseguito un passaggio a quel cluster. È possibile specificare un tag sul cluster globale per registrare l'ora dell'ultima esercitazione di disaster recovery per l'intera applicazione.

AWS CLI esempi di etichettatura per database globali

AWS CLI Gli esempi seguenti mostrano come specificare ed esaminare i tag per tutti i tipi di risorse Aurora nel database globale.

È possibile specificare i tag per il contenitore del cluster globale con il `create-global-cluster` comando. L'esempio seguente crea un cluster globale e gli assegna due tag. I tag hanno chiavi `tag1` e `tag2`.

```
$  aws rds create-global-cluster --global-cluster-identifier my_global_cluster_id \
  --engine aurora-mysql --tags Key=tag1,Value=val1 Key=tag2,Value=val2
```

È possibile elencare i tag nel contenitore del cluster globale con il `describe-global-clusters` comando. Quando lavori con i tag, in genere esegui prima questo comando per recuperare l'Amazon Resource Name (ARN) del cluster globale. Lo usi ARN come parametro nei comandi successivi per lavorare con i tag. Il comando seguente visualizza le informazioni sui tag nell'`TagList`attributo. Viene inoltre mostrato ilARN, che viene utilizzato come parametro negli esempi successivi.

```
$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
```

```
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "GlobalClusterArn": "my_global_cluster_arn",
            ...
            "TagList": [
                {
                    "Value": "val1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                }
            ]
        }
    ]
}
```

È possibile aggiungere nuovi tag con il `add-tags-to-resource` comando. Con questo comando, specifichi l'Amazon Resource Name (ARN) del cluster globale anziché il suo identificatore. L'aggiunta di un tag con lo stesso nome di un tag esistente sovrascrive il valore di quel tag. Se includi spazi o caratteri speciali nei valori dei tag, indica i valori appropriati per il tuo sistema operativo o la shell dei comandi. L'esempio seguente modifica i tag del cluster globale rispetto all'esempio precedente. Originariamente, il cluster aveva tag con chiavi `tag1` e `tag2`. Al termine del comando, il cluster globale ha un nuovo tag con chiave `tag3` e il tag con chiave `tag1` ha un valore diverso.

```
$  aws rds add-tags-to-resource --resource-name my_global_cluster_arn \
  --tags Key=tag1,Value="new value for tag1" Key=tag3,Value="entirely new tag"

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
```

```
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                },
                {
                    "Value": "entirely new tag",
                    "Key": "tag3"
                }
            ]
        }
    ]
}
```

È possibile eliminare un tag dal cluster globale con il `remove-tags-from-resource` comando. Con questo comando, si specifica solo un set di chiavi di tag, senza alcun valore di tag. L'esempio seguente modifica i tag del cluster globale dell'esempio precedente. Originariamente, il cluster aveva tag con chiavi `tag1tag2`, `etag3`. Al termine del comando, `tag1` rimane solo il tag con la chiave.

```
$  aws rds remove-tags-from-resource --resource-name my_global_cluster_arn --tag-keys
 tag2 tag3

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                }
            ]
        }
    ]
}
```

Connessione al database globale di Amazon Aurora

Ogni Aurora Global Database è dotato di un endpoint writer che viene aggiornato automaticamente da Aurora per instradare le richieste all'istanza di scrittura corrente del cluster DB primario. Con l'endpoint writer, non è necessario modificare la stringa di connessione dopo aver cambiato la posizione della regione primaria utilizzando le funzionalità gestite di switchover e failover di Aurora Global Database. Per ulteriori informazioni sull'utilizzo dell'endpoint writer insieme allo switchover e al failover di Aurora Global Database, vedere. Utilizzo dello switchover o del failover in Amazon Aurora Global Database Per informazioni sulla connessione a un database globale Aurora con proxy RDS, vedere Utilizzo del proxy RDS con database globali Aurora.

Argomenti

- Scelta dell'endpoint che soddisfa le esigenze dell'applicazione
- Visualizzazione degli endpoint di un database globale Amazon Aurora
- Considerazioni sull'utilizzo degli endpoint Global writer

Scelta dell'endpoint che soddisfa le esigenze dell'applicazione

La connessione a un database globale Aurora dipende dalla necessità di leggere o scrivere dal database e dalla AWS regione verso cui si desidera indirizzare le richieste. Ecco alcuni casi d'uso tipici:

- Instradamento delle richieste all'istanza writer: Connettiti all'endpoint di scrittura Aurora Global Database se devi eseguire istruzioni DML (Data Manipulation Language) e DDL (Data Definition Language) o se hai bisogno di una forte coerenza tra letture e scritture. Tale endpoint indirizza le richieste all'istanza writer nel cluster primario del database globale. Questo endpoint viene aggiornato automaticamente per indirizzare le richieste all'istanza di writer, eliminando la necessità di aggiornare l'applicazione ogni volta che si modifica la posizione del writer nel cluster globale. Puoi anche utilizzare l'endpoint globale per inviare read/write richieste interregionali al tuo writer.

  > **Note**
  >
  > Se hai configurato il database globale prima che l'endpoint Aurora Global Database writer fosse disponibile, l'applicazione potrebbe connettersi all'endpoint cluster del cluster primario. In questo caso, consigliamo di modificare le impostazioni di connessione per utilizzare invece l'endpoint global writer. In questo modo si evita la necessità di modificare le impostazioni di connessione dopo ogni switchover o failover di Aurora Global Database.

La prima parte del nome dell'endpoint writer è il nome dell'Aurora Global Database. Pertanto, se rinomini il tuo Aurora Global Database, il nome dell'endpoint writer cambia e qualsiasi codice che lo utilizza deve essere aggiornato con il nuovo nome.

- Ridimensionamento delle letture più vicino alla regione dell'applicazione: per ridimensionare le richieste di sola lettura nella stessa AWS regione o nelle vicinanze dell'applicazione, connettiti all'endpoint di lettura dei cluster Aurora primari o secondari.
- Ridimensionamento delle letture con scritture occasionali tra aree geografiche: per istruzioni DML occasionali, ad esempio per la manutenzione e la pulizia dei dati, connettiti all'endpoint di lettura di un cluster secondario con l'inoltro di scrittura abilitato. Con l'inoltro di scrittura, Aurora inoltra automaticamente le istruzioni di scrittura all'autore nella regione principale del database globale di Aurora. L'inoltro delle scritture offre i seguenti vantaggi:
  - Non è necessario fare il lavoro pesante per stabilire la connettività tra i cluster secondari e primari per inviare scritture tra regioni.
  - Non è necessario dividere le richieste di lettura e scrittura nell'applicazione.
  - Non è necessario sviluppare una logica complessa per gestire la coerenza read-after-write delle richieste.

  Tuttavia, con l'inoltro di scrittura, è necessario aggiornare il codice o la configurazione dell'applicazione per connettersi all'endpoint di lettura principale della nuova regione promossa dopo aver eseguito un failover o uno switchover tra regioni. Ti consigliamo di monitorare la latenza delle operazioni eseguite tramite l'inoltro di scrittura per controllare il sovraccarico di elaborazione delle richieste di scrittura. Infine, l'inoltro della scrittura non supporta determinate operazioni di MySQL o PostgreSQL, come apportare modifiche o istruzioni al Data Definition Language (DDL). `SELECT FOR UPDATE`

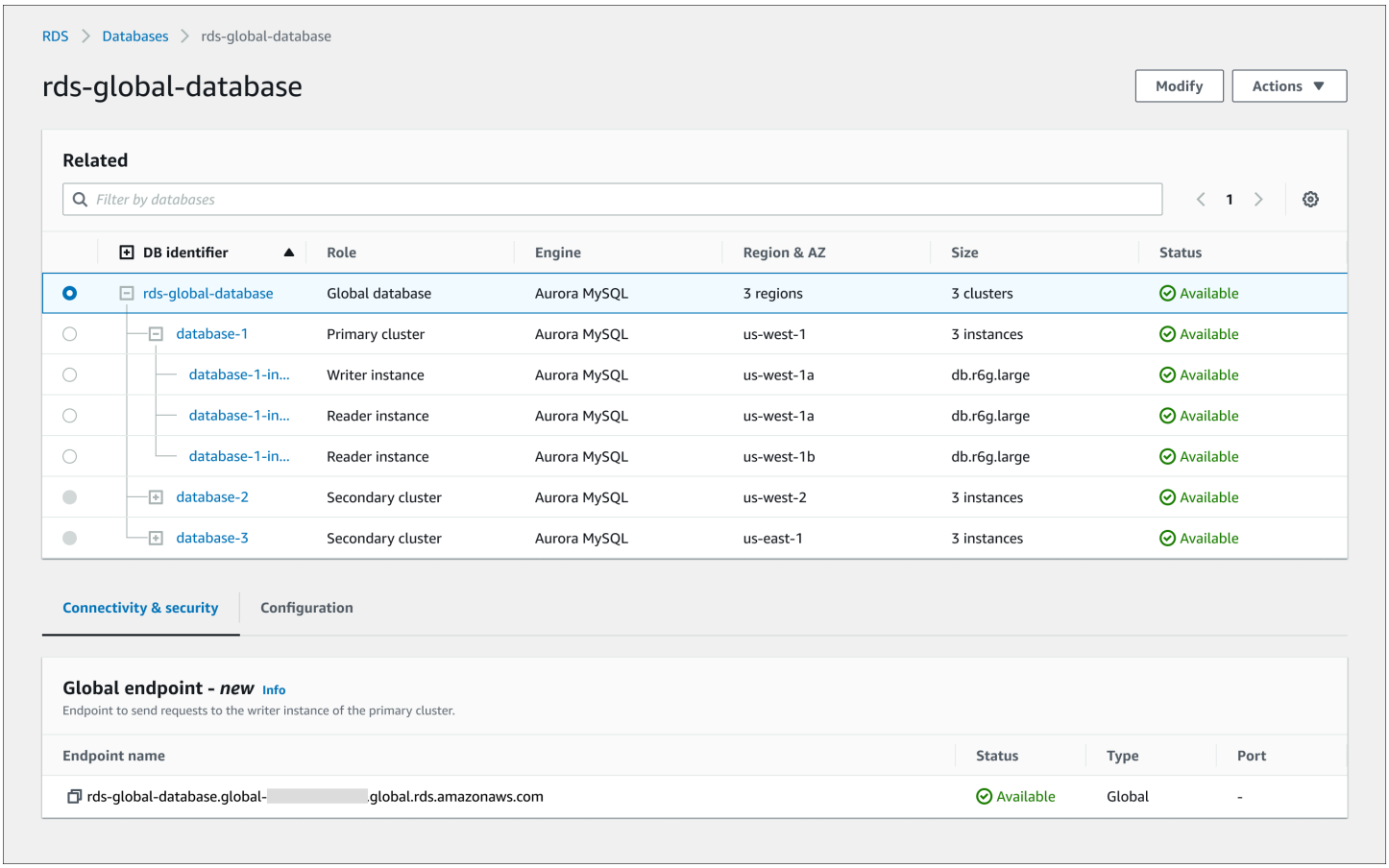
  Per ulteriori informazioni sull'utilizzo dell'inoltro di scrittura tra regioni, consulta. AWS Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora

Per dettagli sui diversi tipi di endpoint Aurora, consulta. Connessione a un cluster database Amazon Aurora

Visualizzazione degli endpoint di un database globale Amazon Aurora

Quando visualizzi un Aurora Global Database nella console, puoi vedere tutti gli endpoint associati a tutti i suoi cluster. La figura seguente mostra un esempio dei tipi di endpoint visualizzati quando si visualizzano i dettagli del cluster DB primario:

- Global writer: il singolo read/write endpoint che punta sempre all'istanza DB Writer corrente per il cluster di database globale.
- Writer: l'endpoint di connessione per read/write le richieste al cluster DB primario nel cluster di database globale.
- Reader: l'endpoint di connessione per le richieste di sola lettura a un cluster DB primario o secondario nel cluster di database globale. Per ridurre al minimo la latenza, scegli l'endpoint di lettura presente nel tuo dispositivo o quello più vicino a te Regione AWS . Regione AWS



Console

Per visualizzare gli endpoint di un database globale

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).

3. Nell'elenco, scegli il database globale o il cluster DB primario o secondario di cui desideri visualizzare gli endpoint.
4. Scegli la scheda Connettività e sicurezza per visualizzare i dettagli dell'endpoint. Gli endpoint visualizzati dipendono dal tipo di cluster selezionato, come segue:

   - Database globale: l'endpoint Global Writer.
   - Cluster DB primario: l'endpoint global writer e l'endpoint cluster e l'endpoint di lettura per il cluster primario.
   - Cluster DB secondario: l'endpoint del cluster e l'endpoint di lettura per il cluster secondario. In un cluster secondario, l'endpoint del cluster mostra lo stato di inattivo perché non gestisce le richieste di scrittura. È ancora possibile connettersi all'endpoint del cluster, ma solo per le query di lettura.

AWS CLI

Per visualizzare l'endpoint writer del cluster globale, utilizzate il AWS CLI [describe-global-clusters](#)comando, come nell'esempio seguente.

```
aws rds describe-global-clusters --region aws_region
{
    "GlobalClusters": [
        {
            "GlobalClusterIdentifier": "global_cluster_id",
            "GlobalClusterResourceId": "cluster-unique_string",
            "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:global_cluster_id",
            "Status": "available",
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",
            "GlobalClusterMembers": [

              ...
            ],
            "Endpoint":
 "global_cluster_id.global-unique_string.global.rds.amazonaws.com"
        }
    ]
}
```

Per visualizzare gli endpoint del cluster e del lettore per i cluster DB membri del cluster globale, utilizzate il AWS CLI describe-db-clusterscomando, come nell'esempio seguente. I valori restituiti per `Endpoint` e `ReaderEndpoint` sono rispettivamente gli endpoint cluster e reader.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier db_cluster_id
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "az_1",
                "az_2",
                "az_3"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "db_cluster_id",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-08-01T18:21:11.301Z",
            "Endpoint":
 "db_cluster_id.cluster-unique_string.primary_region.rds.amazonaws.com",
            "ReaderEndpoint": "db_cluster_id.cluster-
ro-unique_string.primary_region.rds.amazonaws.com",
            "MultiAZ": false,
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",

            "ReadReplicaIdentifiers": [
                "arn:aws:rds:secondary_region:123456789012:cluster:db_cluster_id"
            ],
            "DBClusterMembers": [
                {
                    "DBInstanceIdentifier": "db_instance_id",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                }
            ],

            ...
            "TagList": [],
```

```
            "GlobalWriteForwardingRequested": false
        }
    ]
}
```

API RDS

Per visualizzare l'endpoint writer del cluster globale, utilizzate l'operazione API RDS. DescribeGlobalClusters Per visualizzare gli endpoint del cluster e dei lettori per i cluster DB membri del cluster globale, utilizzate l'operazione RDS API Descrive. DBClusters

Considerazioni sull'utilizzo degli endpoint Global writer

Puoi fare un uso efficace degli endpoint Aurora Global Database Writer seguendo queste linee guida e best practice:

- Per ridurre al minimo le interruzioni dopo un failover o uno switchover tra regioni, puoi configurare la connettività VPC tra l'elaborazione dell'applicazione e le regioni primarie e secondarie. AWS Ad esempio, supponiamo di avere applicazioni o sistemi client in esecuzione nello stesso VPC del cluster primario. Se viene promosso il cluster secondario, l'endpoint global writer cambia automaticamente per puntare a quel cluster. Sebbene l'endpoint global writer consenta di evitare di modificare le impostazioni di connessione per l'applicazione, le applicazioni non possono accedere agli indirizzi IP nel VPC della AWS regione primaria appena promossa finché non si configura la rete tra i due. VPCs Consulta le opzioni di connettività di Amazon VPC-to-Amazon VPC per valutare diverse opzioni per configurare questa connettività.
- L'aggiornamento dell'endpoint global writer dopo un failover o uno switchover del database globale può richiedere molto tempo a seconda della durata della memorizzazione nella cache DNS (Domain Name Service). Per ulteriori informazioni, consulta il Manuale dell'amministratore del database MySQL di Amazon Aurora. Aurora Global Database emette un evento RDS quando rileva la modifica del DNS sull'endpoint Global Writer. È possibile utilizzare l'evento per elaborare strategie per garantire che la cache DNS non si estenda oltre il periodo successivo alla generazione dell'evento. Per ulteriori informazioni, consulta Eventi di cluster di database.
- Aurora Global Database replica i dati in modo asincrono. I metodi di failover interregionale possono far sì che alcuni dati delle transazioni di scrittura non siano stati replicati sul secondario scelto prima dell'inizio del failover. Sebbene Aurora tenti con il massimo impegno di bloccare le scritture nella AWS regione primaria originale, il failover può essere soggetto a problemi di split-brain. Le considerazioni per ridurre al minimo la perdita di dati e il rischio di split-brain si applicano anche agli

endpoint Aurora Global Database Writer. Per ulteriori informazioni, consulta Esecuzione di failover gestiti per database globali Aurora.

Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora

È possibile ridurre il numero di endpoint che è necessario gestire per le applicazioni in esecuzione nel database globale Aurora utilizzando l' inoltro in scrittura. Quando la funzionalità di inoltro di scrittura è abilitata, i cluster secondari di un database globale Aurora inoltrano le istruzioni SQL che eseguono operazioni di scrittura al cluster primario. Il cluster primario aggiorna l'origine e quindi propaga le modifiche risultanti a tutte le aree secondarie AWS.

La funzione di inoltro di scrittura consente di evitare di implementare il proprio meccanismo per inviare operazioni di scrittura da una regione AWS secondaria alla regione principale. Aurora gestisce la configurazione di rete tra regioni. Aurora trasmette anche tutte le sessioni necessarie e il contesto transazionale per ogni dichiarazione. I dati vengono sempre modificati prima nel cluster primario e quindi replicati nuovamente nei cluster secondari nel database globale Aurora. In questo modo, il cluster primario è sempre la fonte di attendibilità con la copia più aggiornata di tutti i dati.

Argomenti

- Utilizzo dell'inoltro di scrittura in un database globale Aurora MySQL
- Utilizzo dell'inoltro di scrittura in un database globale Aurora PostgreSQL

Utilizzo dell'inoltro di scrittura in un database globale Aurora MySQL

Argomenti

- Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora MySQL
- Abilitazione dell'inoltro di scrittura in Aurora MySQL
- Verifica se un cluster secondario ha abilitato l'inoltro di scrittura
- Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora MySQL
- Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL
- Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora MySQL
- Transazioni con inoltro di scrittura in Aurora MySQL
- Parametri di configurazione per l'inoltro di scrittura in Aurora MySQL

- CloudWatch Parametri Amazon per l'inoltro della scrittura in Aurora MySQL
- Variabili di stato Aurora MySQL per l'inoltro della scrittura

Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora MySQL

L'inoltro della scrittura è supportato con Aurora MySQL 2.08.1 e versioni successive in ogni regione in cui sono disponibili database globali basati su Aurora MySQL.

Per ulteriori informazioni sulla disponibilità di versioni e regioni dei database globali Aurora MySQL consulta Database globali Aurora con Aurora MySQL.

Abilitazione dell'inoltro di scrittura in Aurora MySQL

Per impostazione predefinita, l'inoltro di scrittura non è abilitato quando si aggiunge un cluster secondario a un Aurora Global Database.

Per abilitare l'inoltro di scrittura utilizzando AWS Management Console, seleziona la casella di controllo Attiva l'inoltro globale di scrittura in Leggi l'inoltro di scrittura della replica quando aggiungi una regione per un database globale. Per un cluster secondario esistente, modifica il cluster in Attiva l'inoltro di scrittura globale. Per disattivare l'inoltro di scrittura, deseleziona la casella di controllo Attiva l'inoltro di scrittura globale durante l'aggiunta della regione o la modifica del cluster secondario.

Per abilitare l'inoltro di scrittura utilizzando l'opzione. AWS CLI`--enable-global-write-forwarding` Questa opzione funziona quando si crea un nuovo cluster secondario utilizzando il comando `create-db-cluster`. Funziona anche quando si modifica un cluster secondario esistente utilizzando il comando `modify-db-cluster`. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. È possibile disattivare l'inoltro di scrittura utilizzando l'opzione `--no-enable-global-write-forwarding` con questi stessi comandi CLI.

Per abilitare l'inoltro di scrittura utilizzando l'API Amazon RDS, impostare il parametro `EnableGlobalWriteForwarding` su `true`. Questo parametro funziona quando si crea un nuovo cluster secondario utilizzando l'operazione `CreateDBCluster`. Funziona anche quando si modifica un cluster secondario esistente utilizzando l'operazione `ModifyDBCluster`. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. È possibile disattivare l'inoltro di scrittura impostando il parametro `EnableGlobalWriteForwarding` su `false`.

> **Note**
>
> Per utilizzare l'inoltro in scrittura in una sessione di database, specifica un'impostazione per il parametro di configurazione `aurora_replica_read_consistency`. Eseguire questa operazione in ogni sessione che utilizza la funzionalità di inoltro di scrittura. Per informazioni su questo parametro, consulta Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL.
>
> La funzionalità RDS Proxy non supporta il valore `SESSION` della variabile `aurora_replica_read_consistency`, la cui impostazione può causare un comportamento imprevisto.

Negli esempi seguenti di CLI viene illustrato come impostare un Aurora Global Database con l'inoltro di scrittura abilitato o disabilitato. Gli elementi evidenziati rappresentano i comandi e le opzioni che è importanti specificare e mantenere coerenti quando si imposta l'infrastruttura per un Aurora Global Database.

Nell'esempio seguente viene creato un Aurora Global Database, un cluster primario e un cluster secondario con l'inoltro di scrittura abilitato. Sostituire il nome utente, la password e le regioni AWS primarie e secondarie.

```
# Create overall global database.
aws rds create-global-cluster --global-cluster-identifier write-forwarding-test \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create primary cluster, in the same AWS Region as the global database.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-1 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --master-username user_name --master-user-password password \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
```

```
  --db-instance-identifier write-forwarding-test-cluster-1-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create secondary cluster, in a different AWS Region than the global database,
# with write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2 \
  --enable-global-write-forwarding

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2
```

L'esempio seguente continua dal precedente. Crea un cluster secondario senza l'inoltro di scrittura abilitato, quindi abilita l'inoltro di scrittura. Al termine di questo esempio, tutti i cluster secondari del database globale hanno attivato l'inoltro di scrittura.

```
# Create secondary cluster, in a different AWS Region than the global database,
# without write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
```

```
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds modify-db-cluster --db-cluster-identifier write-forwarding-test-cluster-2 \
  --region us-east-2 \
  --enable-global-write-forwarding
```

Verifica se un cluster secondario ha abilitato l'inoltro di scrittura

Per determinare se è possibile utilizzare l'inoltro di scrittura da un cluster secondario, è possibile verificare se il cluster dispone dell'attributo `"GlobalWriteForwardingStatus": "enabled"`.

Nella AWS Management Console scheda Configurazione della pagina dei dettagli del cluster, viene visualizzato lo stato Abilitato per l'inoltro globale di lettura, replica e scrittura.

Per visualizzare lo stato dell'impostazione globale di inoltro della scrittura per tutti i cluster, esegui il comando seguente. AWS CLI

Un cluster secondario mostra il valore `"enabled"` o `"disabled"` per indicare se l'inoltro di scrittura è attivato o disattivato. Il valore `null` indica che l'inoltro di scrittura non è disponibile per il cluster. Il cluster non fa parte di un database globale oppure è il cluster primario anziché un cluster secondario. Il valore può anche essere `"enabling"` o `"disabling"` se l'inoltro di scrittura è in procinto di essere attivato o disattivato.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu

[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
```

```
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Per trovare tutti i cluster secondari per i quali è abilitato l'inoltro di scrittura globale, eseguire il comando seguente. Questo comando restituisce anche l'endpoint di lettura del cluster. È possibile utilizzare l'endpoint di lettura del cluster secondario quando si utilizza l'inoltro di scrittura dal cluster secondario al primario nel database globale Aurora.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora MySQL

È possibile utilizzare i seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`. Esistono alcune restrizioni sulle proprietà di queste istruzioni che è possibile utilizzare con l'inoltro di scrittura, come descritto di seguito.
- Istruzioni `SELECT ... LOCK IN SHARE MODE` e `SELECT FOR UPDATE`.
- Istruzioni `PREPARE` e `EXECUTE`.

Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in un database globale con inoltro di scrittura. Pertanto, l'impostazione `EnableGlobalWriteForwarding` è disattivata in modo predefinito per i cluster secondari. Prima di attivarlo, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.

Le seguenti restrizioni si applicano alle istruzioni SQL utilizzate con l'inoltro di scrittura. In alcuni casi, è possibile utilizzare le istruzioni su cluster secondari con l'inoltro di scrittura abilitato a livello

di cluster. Questo approccio funziona se l'inoltro di scrittura non è attivato all'interno della sessione dal parametro di configurazione `aurora_replica_read_consistency`. Se si tenta di utilizzare un'istruzione quando non è consentito a causa dell'inoltro di scrittura, viene generato un messaggio di errore con il formato seguente.

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

DDL (Data Definition Language)

Connettersi al cluster primario per eseguire le istruzioni DDL. Non è possibile eseguirle da istanze database di lettura.

Aggiornamento di una tabella permanente utilizzando i dati di una tabella temporanea

È possibile utilizzare tabelle temporanee in cluster secondari con l'inoltro di scrittura abilitato. Tuttavia, non è possibile utilizzare un'istruzione DML per modificare una tabella permanente se l'istruzione fa riferimento a una tabella temporanea. Ad esempio, non è possibile utilizzare un'istruzione `INSERT ... SELECT` che accetta i dati da una tabella temporanea. La tabella temporanea esiste nel cluster secondario e non è disponibile quando l'istruzione viene eseguita nel cluster primario.

Transazioni XA

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario quando l'inoltro di scrittura è attivato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster secondari per i quali non è abilitato l'inoltro di scrittura o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Prima di attivare l'inoltro di scrittura all'interno di una sessione, verificare se il codice utilizza queste istruzioni.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Istruzioni LOAD per tabelle permanenti

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario con l'inoltro di scrittura abilitato.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
        LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

È possibile caricare i dati in una tabella temporanea in un cluster secondario. Tuttavia, assicurarsi di eseguire tutte le istruzioni LOAD che fanno riferimento a tabelle permanenti solo nel cluster primario.

Istruzioni plugin

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario con l'inoltro di scrittura abilitato.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Istruzioni SAVEPOINT

Non è possibile utilizzare le istruzioni seguenti in un cluster secondario quando l'inoltro di scrittura è attivato all'interno della sessione. È possibile utilizzare queste istruzioni su cluster secondari per i quali non è abilitato l'inoltro di scrittura o all'interno di sessioni in cui l'impostazione `aurora_replica_read_consistency` è vuota. Controlla se il tuo codice utilizza queste istruzioni prima di attivare l'inoltro di scrittura all'interno di una sessione.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo il livello di isolamento `REPEATABLE READ`. Sebbene sia anche possibile utilizzare il livello di isolamento `READ COMMITTED` con cluster di sola lettura nelle regioni AWS secondarie, tale livello di isolamento non funziona con l'inoltro di scrittura. Per informazioni sui livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`, consulta Livelli di isolamento di Aurora MySQL.

È possibile controllare il grado di coerenza di lettura in un cluster secondario. Il livello di coerenza di lettura determina la quantità di attesa di un cluster secondario prima di ogni operazione di lettura per garantire che alcune o tutte le modifiche vengano replicate dal cluster primario. È possibile

regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nel cluster secondario prima di qualsiasi query successiva. È inoltre possibile utilizzare questa impostazione per garantire che le query sul cluster secondario visualizzino sempre gli aggiornamenti più recenti dal cluster primario. Ciò si verifica anche per quelli inviati da altre sessioni o altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegliere un valore per il parametro a livello di sessione `aurora_replica_read_consistency`.

> **Important**
>
> Impostare sempre il parametro `aurora_replica_read_consistency` per qualsiasi sessione per la quale si desidera inoltrare le scritture. In caso contrario, Aurora non abilita l'inoltro di scrittura per quella sessione. Questo parametro ha un valore vuoto per impostazione predefinita, quindi scegli un valore specifico quando utilizzi questo parametro. Il parametro `aurora_replica_read_consistency` ha effetto solo sui cluster secondari in cui è abilitato l'inoltro di scrittura.
> Per Aurora MySQL versione 2 e versione 3 inferiori alla 3.04, usa `aurora_replica_read_consistency` come variabile di sessione. Per Aurora MySQL versione 3.04 e successive, è possibile usare `aurora_replica_read_consistency` come variabile di sessione o come un parametro del cluster database.

Per il parametro `aurora_replica_read_consistency`, è possibile specificare i valori `EVENTUAL`, `SESSION` e `GLOBAL`.

Aumentando il livello di coerenza, l'applicazione impiega più tempo ad aspettare che le modifiche vengano propagate tra le regioni. AWS È possibile scegliere il bilanciamento tra tempi di risposta rapidi e garantire che le modifiche apportate in altre posizioni siano completamente disponibili prima dell'esecuzione delle query.

Con la coerenza di lettura impostata su`EVENTUAL`, le query in una AWS regione secondaria che utilizza l'inoltro di scrittura potrebbero visualizzare dati leggermente obsoleti a causa del ritardo di replica. I risultati delle operazioni di scrittura nella stessa sessione non sono visibili fino a quando l'operazione di scrittura non viene eseguita nella regione primaria e replicata nella regione corrente. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati meno recenti o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica.

Con la coerenza di lettura impostata su`SESSION`, tutte le query in un' AWS area secondaria che utilizza l'inoltro di scrittura visualizzano i risultati di tutte le modifiche apportate in quella sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati nell'area corrente. Non attende i risultati aggiornati delle operazioni di scrittura eseguite in altre regioni o in altre sessioni all'interno dell'area corrente.

Con la coerenza di lettura impostata su`GLOBAL`, in una sessione in un' AWS area secondaria vengono visualizzate le modifiche apportate da quella sessione. Vengono inoltre visualizzate tutte le modifiche impegnate sia dalla AWS regione primaria che da altre AWS regioni secondarie. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando il cluster secondario contiene tutti up-to-date i dati salvati dal cluster primario, a partire dal momento in cui è iniziata la query.

Per ulteriori informazioni su tutti i parametri coinvolti nell'inoltro di scrittura, consulta Parametri di configurazione per l'inoltro di scrittura in Aurora MySQL.

Esempi di utilizzo dell'inoltro di scrittura

Questi esempi utilizzano `aurora_replica_read_consistency` come una variabile di sessione. Per Aurora MySQL versione 3.04 e successive, è possibile usare `aurora_replica_read_consistency` come variabile di sessione o come un parametro del cluster database.

Nell'esempio seguente, il cluster primario si trova nella regione US East (N. Virginia). Il cluster secondario si trova nella regione Stati Uniti orientali (Ohio). L'esempio mostra gli effetti delle istruzioni `INSERT` in esecuzione seguite da istruzioni `SELECT`. A seconda del valore dell'impostazione `aurora_replica_read_consistency`, i risultati potrebbero differire a seconda della tempistica delle istruzioni. Per ottenere una maggiore coerenza, è possibile attendere brevemente prima di rilasciare l'istruzione `SELECT`. Oppure Aurora può attendere automaticamente il completamento della replica dei risultati prima di procedere con `SELECT`.

In questo esempio, è disponibile un'impostazione di coerenza di lettura di `eventual`. L'esecuzione di una istruzione `INSERT` immediatamente seguita da una istruzione `SELECT` restituisce ancora il valore di `COUNT(*)`. Questo valore riflette il numero di righe prima dell'inserimento della nuova riga. Se poco dopo si esegue nuovamente `SELECT`, viene restituito il conteggio delle righe aggiornato. Le istruzioni `SELECT` non attendono.

```
mysql> set aurora_replica_read_consistency = 'eventual';
mysql> select count(*) from t1;
```

```
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Con un'impostazione di coerenza di lettura di `session`, un'istruzione `SELECT` immediatamente dopo un `INSERT` attende fino a quando le modifiche dall'istruzione `INSERT` diventano visibili. Le istruzioni `SELECT` successive non attendono.

```
mysql> set aurora_replica_read_consistency = 'session';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)
mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
```

```
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Con l'impostazione di coerenza di lettura ancora impostata su `session`, l'introduzione di una breve attesa dopo l'esecuzione di un'istruzione `INSERT` rende disponibile il conteggio delle righe aggiornate dal momento dell'esecuzione dell'istruzione `SELECT` successiva.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Con un'impostazione di coerenza di lettura di `global`, ogni istruzione `SELECT` attende di assicurarsi che tutte le modifiche ai dati a partire dall'ora di inizio dell'istruzione siano visibili prima di eseguire la query. La quantità di attesa per ogni istruzione `SELECT` varia a seconda della quantità di ritardo di replica tra i cluster primario e secondario.

```
mysql> set aurora_replica_read_consistency = 'global';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
```

```
+----------+
1 row in set (0.37 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora MySQL

Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata al cluster primario ed eseguita lì.

Transazioni con inoltro di scrittura in Aurora MySQL

Se la transazione viene inoltrata al cluster primario dipende dalla modalità di accesso della transazione. È possibile specificare la modalità di accesso per la transazione utilizzando l'istruzione `SET TRANSACTION` o `START TRANSACTION`. È anche possibile specificare la modalità di accesso alle transazioni modificando il valore della variabile di sessione transaction_read_only. È possibile modificare questo valore di sessione solo quando si è connessi a un cluster database in cui l'inoltro di scrittura è abilitato.

Se una transazione di lunga durata non emette alcuna dichiarazione per un periodo di tempo considerevole, potrebbe superare il periodo di timeout inattivo. Questo periodo ha un valore predefinito di un minuto. Puoi aumentarlo fino a un giorno. Una transazione che supera il timeout di inattività viene annullata dal cluster primario. L'istruzione successiva inviata riceve un errore di timeout. Quindi Aurora esegue il rollback della transazione.

Questo tipo di errore può verificarsi in altri casi in cui l'inoltro di scrittura non diventa disponibile. Ad esempio, Aurora annulla tutte le transazioni che utilizzano l'inoltro di scrittura se si riavvia il cluster primario o se si disattiva l'impostazione di configurazione dell'inoltro di scrittura.

Parametri di configurazione per l'inoltro di scrittura in Aurora MySQL

I gruppi di parametri del cluster Aurora contengono delle impostazioni per la funzionalità di inoltro di scrittura. Poiché si tratta di parametri cluster, tutte le istanze database in ogni cluster hanno gli stessi valori per queste variabili. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Nome	Ambito	Tipo	Valore predefinito	Valori validi
`aurora_fwd_master_idle_timeout` (Aurora MySQL versione 2)	Globale	unsigned integer	60	1–86.400
`aurora_fwd_master_max_connections_pct` (Aurora MySQL versione 2)	Globale	intero lungo senza segno	10	0–90
`aurora_fwd_writer_idle_timeout` (Aurora MySQL version 3)	Globale	unsigned integer	60	1–86.400
`aurora_fwd_writer_max_connections_pct` (Aurora MySQL version 3)	Globale	intero lungo senza segno	10	0–90
`aurora_replica_read_consistency`	Sessione per la versione 2 e la versione 3 precedenti alla 3.04, Global per la versione 3.04 e successive	Enum	" (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Per controllare le richieste di scrittura in ingresso da cluster secondari, utilizza le seguenti impostazioni nel cluster primario:

- `aurora_fwd_master_idle_timeout`, `aurora_fwd_writer_idle_timeout`: il numero di secondi in cui il cluster primario attende l'attività su una connessione inoltrata da un cluster

secondario prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, la sessione viene annullata da Aurora.

- `aurora_fwd_master_max_connections_pct`, `aurora_fwd_writer_max_connections_pct`: il limite superiore delle connessioni al database che può essere utilizzato su un'istanza database writer per gestire le query inoltrate dai lettori. Viene espresso come percentuale dell'impostazione `max_connections` per l'istanza database writer nel cluster primario. Ad esempio, se `max_connections` è 800 e `aurora_fwd_master_max_connections_pct` o `aurora_fwd_writer_max_connections_pct` è 10, allora lo scrittore consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione `max_connections`.

  Questa impostazione si applica solo al cluster primario, quando uno o più cluster secondari hanno abilitato l'inoltro di scrittura. Se si riduce il valore, le connessioni esistenti non vengono influenzate. Aurora tiene in considerazione il nuovo valore dell'impostazione quando si prova a creare una nuova connessione da un cluster secondario. Il valore predefinito è 10, che rappresenta il 10% del valore `max_connections`. Se si abilita l'inoltro di query su uno qualsiasi dei cluster secondari, questa impostazione deve avere un valore diverso da zero affinché le operazioni di scrittura dai cluster secondari abbiano esito positivo. Se il valore è zero, le operazioni di scrittura ricevono il codice di errore `ER_CON_COUNT_ERROR` con il messaggio `Not enough connections on writer to handle your request`.

Il `aurora_replica_read_consistency` parametro abilita l'inoltro della scrittura. Lo si utilizza in ogni sessione. È possibile specificare `EVENTUAL`, `SESSION` o `GLOBAL` per il livello di coerenza di lettura. Per ulteriori informazioni sui livelli di coerenza, consulta Isolamento e coerenza per l'inoltro di scrittura in Aurora MySQL. A questo parametro si applicano le seguenti regole:

- Il valore predefinito è '' (empty).
- L'inoltro di scrittura è disponibile in una sessione solo se `aurora_replica_read_consistency` è impostato su `EVENTUAL` o `SESSION` o `GLOBAL`. Questo parametro è rilevante solo nelle istanze di lettura di cluster secondari che hanno l'inoltro di scrittura abilitato e che si trovano in un Aurora Global Database.
- Non è possibile impostarlo come variabile (quando vuoto) o unset (se già impostato) all'interno di una transazione multiistruzione. Tuttavia, è possibile modificarlo da un valore valido (`EVENTUALSESSION`, o `GLOBAL`) a un altro valore valido (`EVENTUALSESSION`, o `GLOBAL`) durante tale transazione.

- La variabile non può essere SET quando l'inoltro di scrittura non è abilitato nel cluster secondario.

CloudWatch Parametri Amazon per l'inoltro della scrittura in Aurora MySQL

I seguenti CloudWatch parametri Amazon si applicano al cluster primario quando utilizzi l'inoltro di scrittura su uno o più cluster secondari. Questi parametri sono tutti misurati sull'istanza database writer nel cluster primario.

CloudWatch parametro	Unità	Descrizione
`AuroraDMLRejectedMasterFull`	Conteggio	Il numero di query inoltrate che vengono rifiutate perché la sessione è piena sull'istanza DB di Writer. Aurora MySQL versione 2
`AuroraDMLRejectedWriterFull`	Conteggio	Il numero di query inoltrate che vengono rifiutate perché la sessione è piena sull'istanza Writer DB. Per Aurora MySQL versione 3.
`ForwardingMasterDMLLatency`	Millisecondi	Tempo medio per elaborare ogni istruzione DML inoltrata sull'istanza database writer. Non include il tempo impiegato dal cluster secondario per inoltrare la richiesta di scrittura o il tempo necessario per replicare le modifiche al cluster secondario. Aurora MySQL versione 2
`ForwardingMasterDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate ogni

CloudWatch parametro	Unità	Descrizione
		secondo da questa istanza database writer. Aurora MySQL versione 2
`ForwardingMasterOpenSessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Aurora MySQL versione 2
`ForwardingWriterDMLLatency`	Millisecondi	Tempo medio per elaborare ogni istruzione DML inoltrata sull'istanza database writer. Non include il tempo impiegato dal cluster secondario per inoltrare la richiesta di scrittura o il tempo necessario per replicare le modifiche al cluster secondario. Per Aurora MySQL versione 3.
`ForwardingWriterDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer. Per Aurora MySQL versione 3.
`ForwardingWriterOpenSessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Per Aurora MySQL versione 3.

Le seguenti CloudWatch metriche si applicano a ogni cluster secondario. Questi parametri vengono misurati su ogni istanza database del lettore in un cluster secondario con l'inoltro di scrittura abilitato.

CloudWatch parametro	Unità	Descrizione
`ForwardingReplicaDMLLatency`	Millisecondi	Tempo di risposta medio inoltrato DMLs sulla replica.
`ForwardingReplicaDMLThroughput`	Conteggio al secondo	Numero di istruzioni DML inoltrate elaborate al secondo.
`ForwardingReplicaOpenSessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`ForwardingReplicaReadWaitLatency`	Millisecondi	Tempo medio di attesa di un'istruzione `SELECT` su un'istanza del database di scrittura prima di raggiungere il cluster primario. Il grado di attesa dell'istanza database del lettore prima di elaborare una query dipende dall'impostazione `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Conteggio al secondo	Numero totale di istruzioni `SELECT` elaborate ogni secondo in tutte le sessioni che inoltrano scritture.
`ForwardingReplicaSelectLatency`	Millisecondi	Forwarded`SELECT`latenza, media su tutte le inoltrate `SELECT`dichiarazioni entro il periodo di monitoraggio.
`ForwardingReplicaSelectThroughput`	Conteggio al secondo	Velocità di trasmissione effettiva per

CloudWatch parametro	Unità	Descrizione
		secondo `SELECT` inoltrata media all'interno del periodo di monitoraggio.

Variabili di stato Aurora MySQL per l'inoltro della scrittura

Le seguenti variabili di stato Aurora MySQL si applicano al cluster primario quando si utilizza l'inoltro di scrittura su uno o più cluster secondari. Questi parametri sono tutti misurati sull'istanza database writer nel cluster primario.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_master_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate a questa istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_master_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate a questa istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_master_open_sessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_master_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate a questa istanza database writer. Aurora MySQL versione 2

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_master_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate a questa istanza database writer. Aurora MySQL versione 2
`Aurora_fwd_writer_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate a questa istanza database writer. Per Aurora MySQL versione 3.
`Aurora_fwd_writer_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate a questa istanza database writer.
`Aurora_fwd_writer_open_sessions`	Conteggio	Numero di sessioni inoltrate sull'istanza database writer. Per Aurora MySQL versione 3.
`Aurora_fwd_writer_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate a questa istanza database writer. Per Aurora MySQL versione 3.
`Aurora_fwd_writer_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate a questa istanza database writer. Per Aurora MySQL versione 3.

Le seguenti variabili di stato Aurora MySQL si applicano a ciascun cluster secondario. Questi parametri vengono misurati su ogni istanza database del lettore in un cluster secondario con l'inoltro di scrittura abilitato.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_replica_dml_stmt_count`	Conteggio	Numero totale di istruzioni DML inoltrate da questa istanza database del lettore.
`Aurora_fwd_replica_dml_stmt_duration`	Microsecondi	Durata totale delle istruzioni DML inoltrate da questa istanza database del lettore.
`Aurora_fwd_replica_errors_session_limit`	Conteggio	Numero di sessioni rifiutate dal cluster primario a causa di una delle seguenti condizioni di errore: • istanza di scrittura piena • Troppe istruzioni inoltrate in corso.
`Aurora_fwd_replica_open_sessions`	Conteggio	Numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza database di lettura.
`Aurora_fwd_replica_read_wait_count`	Conteggio	Numero totale di read-after-write attese su questa istanza DB del lettore.
`Aurora_fwd_replica_read_wait_duration`	Microsecondi	Durata totale delle attese a causa dell'impostazione di coerenza di lettura su questa istanza database del lettore.
`Aurora_fwd_replica_select_stmt_count`	Conteggio	Numero totale di istruzioni `SELECT` inoltrate dall'istanza database del lettore.

Variabile di stato Aurora MySQL	Unità	Descrizione
`Aurora_fwd_replica_select_stmt_duration`	Microsecondi	Durata totale delle istruzioni `SELECT` inoltrate da questa istanza database del lettore.

Utilizzo dell'inoltro di scrittura in un database globale Aurora PostgreSQL

Argomenti

- Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora PostgreSQL
- Abilitazione dell'inoltro di scrittura in Aurora PostgreSQL
- Verifica se un cluster secondario ha abilitato l'inoltro di scrittura in Aurora PostgreSQL
- Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora PostgreSQL
- Isolamento e coerenza per l'inoltro di scrittura in Aurora PostgreSQL
- Modalità di accesso alle transazioni con inoltro di scrittura
- Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora PostgreSQL
- Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL
- CloudWatch Parametri Amazon per l'inoltro delle scritture in Aurora PostgreSQL
- Eventi di attesa per l'inoltro di scrittura in Aurora PostgreSQL

Disponibilità di regioni e versioni dell'inoltro di scrittura in Aurora PostgreSQL

In Aurora PostgreSQL versione 16 e versioni principali successive, l'inoltro globale di scrittura è supportato in tutte le versioni secondarie. Per le versioni precedenti di Aurora PostgreSQL, l'inoltro globale della scrittura è supportato dalla versione 15.4 e dalle versioni secondarie successive e dalla versione 14.9 e versioni secondarie successive. L'inoltro di scrittura è disponibile in tutte le AWS regioni in cui sono disponibili database globali basati su Aurora PostgreSQL.

Per ulteriori informazioni sulla disponibilità di versioni e regioni dei database globali Aurora PostgreSQL consulta Database globali Aurora con Aurora PostgreSQL.

Abilitazione dell'inoltro di scrittura in Aurora PostgreSQL

Per impostazione predefinita, l'inoltro di scrittura non è abilitato quando si aggiunge un cluster secondario a un Aurora Global Database. È possibile abilitare l'inoltro di scrittura per il cluster di database secondario durante la creazione o in qualsiasi momento dopo averlo creato. Se necessario, puoi disabilitarlo in un secondo momento. L'attivazione o la disabilitazione dell'inoltro di scrittura non causa tempi di inattività o il riavvio.

> **Note**
>
> È possibile utilizzare l'inoltro locale di scrittura per le applicazioni che richiedono operazioni di scrittura occasionali e richiedono read-after-write coerenza, ovvero la capacità di leggere l'ultima scrittura in una transazione.

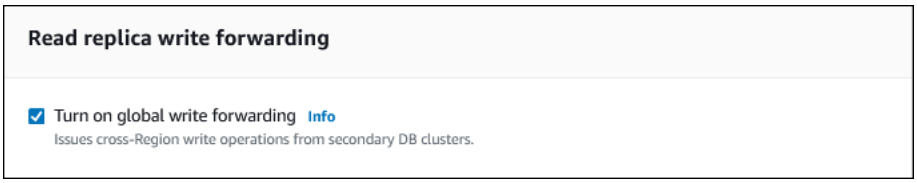
Console

Nella console è possibile attivare o disattivare l'inoltro di scrittura quando si crea o si modifica un cluster di database secondario.

Abilitazione o disabilitazione dell'inoltro di scrittura durante la creazione di un cluster di database secondario

Quando crei un nuovo cluster di database secondario, abiliti l'inoltro di scrittura selezionando la casella di controllo Attiva l'inoltro di scrittura globale in Abilita inoltro in scrittura della replica in lettura. Oppure deseleziona la casella di controllo per disabilitarlo. Per creare un cluster di database secondario, segui le istruzioni in Creazione di un cluster database Amazon Aurora.

Lo screenshot seguente mostra la sezione Abilita inoltro in scrittura della replica in lettura con la casella di controllo Attiva l'inoltro di scrittura globale selezionata.



Abilitazione o disabilitazione dell'inoltro di scrittura durante la modifica di un cluster di database secondario

È possibile modificare un cluster di database secondario nella console per abilitare o disabilitare l'inoltro di scrittura.

Per abilitare o disabilitare l'inoltro di scrittura per un cluster di database secondario utilizzando la console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database.
3. Scegli il cluster di database secondario e scegli Modifica.
4. Nella sezione Abilita inoltro in scrittura della replica in lettura, seleziona o deseleziona la casella di controllo Attiva l'inoltro di scrittura globale.
5. Scegli Continua.
6. In Pianificazione delle modifiche, scegli Applica immediatamente. Se scegli Applica durante la prossima finestra di manutenzione pianificata, Aurora ignora questa impostazione e attiva immediatamente l'inoltro di scrittura.
7. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per abilitare l'inoltro di scrittura utilizzando AWS CLI, usa l'opzione. `--enable-global-write-forwarding` Questa opzione funziona quando si crea un nuovo cluster secondario utilizzando il comando create-db-cluster. Funziona anche quando si modifica un cluster secondario esistente utilizzando il comando modify-db-cluster. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. Puoi disattivare l'inoltro di scrittura mediante l'opzione `--no-enable-global-write-forwarding` con questi stessi comandi CLI.

Le seguenti procedure descrivono come abilitare o disabilitare l'inoltro di scrittura per un cluster di database secondario nel cluster globale utilizzando la AWS CLI.

Per abilitare o disabilitare l'inoltro di scrittura per un cluster di database secondario esistente

- Chiamate il modify-db-cluster AWS CLI comando e fornite i seguenti valori:
  - `--db-cluster-identifier`: il nome del cluster di database.
  - `--enable-global-write-forwarding` per attivare o `--no-enable-global-write-forwarding` per disattivare.

  L'esempio seguente mostra come abilitare l'inoltro di scrittura per un cluster di database `sample-secondary-db-cluster`.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-secondary-db-cluster \
    --enable-global-write-forwarding
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-secondary-db-cluster ^
    --enable-global-write-forwarding
```

API RDS

Per abilitare l'inoltro di scrittura utilizzando l'API Amazon RDS, impostare il parametro `EnableGlobalWriteForwarding` su `true`. Questo parametro funziona quando si crea un nuovo cluster secondario utilizzando l'DBClusteroperazione Create. Funziona anche quando si modifica un cluster secondario esistente utilizzando l'DBClusteroperazione Modifica. Richiede che il database globale utilizzi una versione Aurora che supporti l'inoltro di scrittura. È possibile disattivare l'inoltro di scrittura impostando il parametro `EnableGlobalWriteForwarding` su `false`.

Verifica se un cluster secondario ha abilitato l'inoltro di scrittura in Aurora PostgreSQL

Per determinare se è possibile utilizzare l'inoltro di scrittura da un cluster secondario, è possibile verificare se il cluster dispone dell'attributo `"GlobalWriteForwardingStatus": "enabled"`.

Nella AWS Management Console scheda Configurazione viene visualizzata la pagina dei dettagli del cluster. `Read replica write forwarding` Per visualizzare lo stato dell'impostazione globale di inoltro della scrittura per tutti i cluster, esegui il comando seguente. AWS CLI

Un cluster secondario mostra il valore `"enabled"` o `"disabled"` per indicare se l'inoltro di scrittura è attivato o disattivato. Il valore `null` indica che l'inoltro di scrittura non è disponibile per il cluster. Il cluster non fa parte di un database globale oppure è il cluster primario anziché un cluster secondario. Il valore può anche essere `"enabling"` o `"disabling"` se l'inoltro di scrittura è in procinto di essere attivato o disattivato.

Example

```
aws rds describe-db-clusters --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Per trovare tutti i cluster secondari per i quali è abilitato l'inoltro di scrittura globale, eseguire il comando seguente. Questo comando restituisce anche l'endpoint di lettura del cluster. È possibile utilizzare l'endpoint di lettura del cluster secondario quando si utilizza l'inoltro di scrittura dal cluster secondario al primario nel database globale Aurora.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilità delle applicazioni e di SQL con l'inoltro di scrittura in Aurora PostgreSQL

Alcune istruzioni non sono consentite o possono produrre risultati non aggiornati quando vengono utilizzate in un database globale con inoltro di scrittura. Inoltre, le funzioni e le procedure definite

dall'utente non sono supportate. Pertanto, l'impostazione `EnableGlobalWriteForwarding` è disattivata in modo predefinito per i cluster secondari. Prima di attivarlo, verificare che il codice dell'applicazione non sia interessato da nessuna di queste restrizioni.

È possibile utilizzare i seguenti tipi di istruzioni SQL con l'inoltro di scrittura:

- Istruzioni DML (Data Manipulation Language), ad esempio `INSERT`, `DELETE` e `UPDATE`
- Istruzioni `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`
- Istruzioni `PREPARE` e `EXECUTE`.
- Istruzioni `EXPLAIN` con le istruzioni in questo elenco

I seguenti tipi di istruzioni SQL non sono supportati con l'inoltro di scrittura:

- Istruzioni DDL (Data Definition Language)
- `ANALYZE`
- `CLUSTER`
- `COPY`
- Cursori: i cursori non sono supportati, quindi assicurati di chiuderli prima di utilizzare l'inoltro di scrittura.
- `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
- `LOCK`
- Istruzioni `SAVEPOINT`
- `SELECT INTO`
- `SET CONSTRAINTS`
- `TRUNCATE`
- `VACUUM`

Isolamento e coerenza per l'inoltro di scrittura in Aurora PostgreSQL

Nelle sessioni che utilizzano l'inoltro di scrittura, è possibile utilizzare solo i livelli di isolamento `REPEATABLE READ` e `READ COMMITTED`. Tuttavia, il livello di isolamento `SERIALIZABLE` non è supportato.

È possibile controllare il grado di coerenza di lettura in un cluster secondario. Il livello di coerenza di lettura determina la quantità di attesa di un cluster secondario prima di ogni operazione di lettura

per garantire che alcune o tutte le modifiche vengano replicate dal cluster primario. È possibile regolare il livello di coerenza di lettura per garantire che tutte le operazioni di scrittura inoltrate dalla sessione siano visibili nel cluster secondario prima di qualsiasi query successiva. È inoltre possibile utilizzare questa impostazione per garantire che le query sul cluster secondario visualizzino sempre gli aggiornamenti più recenti dal cluster primario. Ciò si verifica anche per quelli inviati da altre sessioni o altri cluster. Per specificare questo tipo di comportamento per l'applicazione, scegliete il valore appropriato per il `apg_write_forward.consistency_mode` parametro. Il parametro `apg_write_forward.consistency_mode` ha effetto solo sui cluster secondari in cui è abilitato l'inoltro di scrittura.

> **Note**
>
> Per il parametro `apg_write_forward.consistency_mode`, è possibile specificare i valori `SESSION`, `EVENTUAL`, `GLOBAL` o `OFF`. Per impostazione predefinita, il valore è impostato su `SESSION`. L'impostazione del valore su `OFF` disabilita l'inoltro di scrittura nella sessione.

Aumentando il livello di coerenza, l'applicazione impiega più tempo ad aspettare che le modifiche vengano propagate tra le AWS regioni. È possibile scegliere il bilanciamento tra tempi di risposta rapidi e garantire che le modifiche apportate in altre posizioni siano completamente disponibili prima dell'esecuzione delle query.

Ogni impostazione della modalità di coerenza disponibile, produce un effetto come descritto di seguito:

- `SESSION`— Tutte le query in una AWS regione secondaria che utilizza l'inoltro di scrittura visualizzano i risultati di tutte le modifiche apportate in quella sessione. Le modifiche sono visibili indipendentemente dal fatto che la transazione sia stata impegnata. Se necessario, la query attende che i risultati delle operazioni di scrittura inoltrate vengano replicati nell'area corrente. Non attende i risultati aggiornati delle operazioni di scrittura eseguite in altre regioni o in altre sessioni all'interno dell'area corrente.
- `EVENTUAL`— Le query in un' AWS area secondaria che utilizza l'inoltro di scrittura potrebbero visualizzare dati leggermente obsoleti a causa del ritardo di replica. I risultati delle operazioni di scrittura nella stessa sessione non sono visibili fino a quando l'operazione di scrittura non viene eseguita nella regione primaria e replicata nella regione corrente. La query non attende la disponibilità dei risultati aggiornati. Pertanto, potrebbe recuperare i dati meno recenti o i dati aggiornati, a seconda della tempistica delle istruzioni e della quantità di ritardo di replica.

- `GLOBAL`— In una sessione in un' AWS area secondaria vengono visualizzate le modifiche apportate da quella sessione. Visualizza anche tutte le modifiche impegnate sia dalla AWS regione primaria che da altre AWS regioni secondarie. Ogni query potrebbe attendere un periodo che varia a seconda della quantità di ritardo della sessione. La query procede quando il cluster secondario contiene tutti up-to-date i dati salvati dal cluster primario, a partire dal momento in cui è iniziata la query.
- `OFF`: l'inoltro di scrittura durante la sessione è disabilitato.

Per ulteriori informazioni su tutti i parametri coinvolti nell'inoltro di scrittura, consulta Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL.

Modalità di accesso alle transazioni con inoltro di scrittura

Se la modalità di accesso alle transazioni è impostata su sola lettura, l'inoltro di scrittura non viene utilizzato. È possibile impostare la modalità di accesso in modalità di sola lettura e scrittura mentre si è connessi a un cluster di DB e a una sessione con l'inoltro di scrittura abilitato.

Per ulteriori informazioni sulle modalità di accesso alle transazioni, consulta SET TRANSACTION.

Esecuzione di istruzioni a più parti con inoltro scrittura in Aurora PostgreSQL

Un'istruzione DML può essere costituita da più parti, ad esempio un'istruzione `INSERT ... SELECT` o `DELETE ... WHERE`. In questo caso, l'intera istruzione viene inoltrata al cluster primario ed eseguita lì.

Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL

I gruppi di parametri del cluster Aurora contengono delle impostazioni per la funzionalità di inoltro di scrittura. Poiché si tratta di parametri cluster, tutte le istanze database in ogni cluster hanno gli stessi valori per queste variabili. I dettagli su questi parametri sono riepilogati nella tabella seguente, con note di utilizzo dopo la tabella.

Nome	Ambito	Tipo	Valore predefinito	Valori validi
`apg_write_forward.connect_timeout`	Sessione	secondi	30	0–2147483647

Nome	Ambito	Tipo	Valore predefinito	Valori validi
apg_write_forward.consistency_mode	Sessione	enum	Sessione	SESSION, EVENTUAL, GLOBAL, OFF
apg_write_forward.idle_in_transaction_session_timeout	Sessione	millisecondi	86400000	0–2147483647
apg_write_forward.idle_session_timeout	Sessione	millisecondi	300000	0–2147483647
apg_write_forward.max_forwarding_connections_percent	Globale	int	25	1-100

Il parametro apg_write_forward.max_forwarding_connections_percent rappresenta il limite superiore degli slot di connessione al database che possono essere utilizzati per gestire le query inoltrate dalle istanze di lettura. Viene espresso come percentuale dell'impostazione max_connections per l'istanza database di scrittura nel cluster primario. Ad esempio, se max_connections è 800 e apg_write_forward.max_forwarding_connections_percent è 10, allora l'istanza di scrittura consente un massimo di 80 sessioni inoltrate simultanee. Queste connessioni provengono dallo stesso pool di connessioni gestito dall'impostazione max_connections. Questa impostazione si applica solo al cluster primario, quando almeno un cluster ha abilitato l'inoltro di scrittura.

Utilizza le seguenti impostazioni sul cluster secondario:

- apg_write_forward.consistency_mode: un parametro a livello di sessione che controlla il grado di coerenza di lettura sul cluster secondario. I valori validi sono SESSION, EVENTUAL, GLOBAL o OFF. Per impostazione predefinita, il valore è impostato su SESSION. L'impostazione del valore su OFF disabilita l'inoltro di scrittura nella sessione. Per ulteriori informazioni sui livelli di coerenza, consulta Isolamento e coerenza per l'inoltro di scrittura in Aurora PostgreSQL. Questo parametro è rilevante solo nelle istanze di lettura di cluster secondari che hanno l'inoltro di scrittura abilitato e che si trovano in un Aurora Global Database.

- `apg_write_forward.connect_timeout`: il numero massimo di secondi che il cluster secondario attende per stabilire una connessione al cluster primario prima di rinunciare. Il valore `0` indica un tempo di attesa infinito.
- `apg_write_forward.idle_in_transaction_session_timeout`: il numero di millisecondi che il cluster primario attende l'attività su una connessione inoltrata da un cluster secondario con una transazione aperta prima di chiuderla. Se la sessione continua ad avere una transazione inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.
- `apg_write_forward.idle_session_timeout`: il numero di millisecondi che il cluster primario attende l'attività su una connessione inoltrata da un cluster secondario prima di chiuderla. Se la sessione rimane inattiva oltre questo periodo, Aurora la chiude. Il valore `0` disabilita il timeout.

CloudWatch Parametri Amazon per l'inoltro delle scritture in Aurora PostgreSQL

I seguenti CloudWatch parametri Amazon si applicano al cluster primario quando utilizzi l'inoltro di scrittura su uno o più cluster secondari. Questi parametri sono tutti misurati sull'istanza database writer nel cluster primario.

CloudWatch Parametro	Unità e descrizione
`AuroraForwardingWriterDMLThroughput`	Conteggio (al secondo) Numero di istruzioni DML inoltrate elaborate ogni secondo da questa istanza database writer.
`AuroraForwardingWriterOpenSessions`	Conteggio Numero di sessioni aperte su questa istanza database di scrittura che elabora le query inoltrate.
`AuroraForwardingWriterTotalSessions`	Conteggio Numero totale di sessioni inoltrate sull'istanza database di scrittura.

Le seguenti CloudWatch metriche si applicano a ciascun cluster secondario. Questi parametri vengono misurati su ogni istanza database del lettore in un cluster secondario con l'inoltro di scrittura abilitato.

CloudWatch Parametro	Unità e descrizione
`AuroraForwardingReplicaCommitThroughput`	Conteggio (al secondo) Numero di commit in sessioni inoltrate da questa replica ogni secondo.
`AuroraForwardingReplicaDMLLatency`	Millisecondi Tempo di risposta medio in millisecondi di inoltro sulla replica. DMLs
`AuroraForwardingReplicaDMLThroughput`	Conteggio (al secondo) Numero di istruzioni DML inoltrate sulla replica elaborate ogni secondo.
`AuroraForwardingReplicaErrorSessionsLimit`	Conteggio Numero di sessioni rifiutate dal cluster primario quando viene raggiunto il limite massimo di connessioni o di connessioni create per l'inoltro di scrittura.
`AuroraForwardingReplicaOpenSessions`	Conteggio Il numero di sessioni che utilizzano l'inoltro di scrittura su un'istanza di replica.
`AuroraForwardingReplicaReadWaitLatency`	Millisecondi Tempo medio di attesa in millisecondi della replica per garantire la coerenza con l'LSN del cluster primario. Il grado di attesa dell'istanza database in lettura dipende dall'impostazione `apg_write_forward.consistency_mode` . Per ulteriori informazioni su questa impostazione, consulta the section called “Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL”.

Eventi di attesa per l'inoltro di scrittura in Aurora PostgreSQL

Amazon Aurora genera i seguenti eventi di attesa quando si utilizza l'inoltro di scrittura con Aurora PostgreSQL.

Argomenti

- IPC: AuroraWriteForwardConnect
- IPC: AuroraWriteForwardConsistencyPoint
- IPC: AuroraWriteForwardExecute
- IPC: AuroraWriteForwardGetGlobalConsistencyPoint
- IPC: AuroraWriteForwardXactAbort
- IPC: AuroraWriteForwardXactCommit
- IPC: AuroraWriteForwardXactStart

IPC: AuroraWriteForwardConnect

L'evento `IPC:AuroraWriteForwardConnect` si verifica quando un processo di backend sul cluster di database secondario è in attesa dell'apertura della connessione del cluster di database primario al nodo di scrittura.

Probabili cause di aumento delle attese

Questo evento diventa più frequente all'aumentare del numero dei tentativi di connessione dal nodo di lettura di una regione secondaria al nodo di scrittura del cluster di database primario.

Azioni

Riduci il numero di connessioni simultanee da un nodo secondario al nodo di scrittura della regione primaria.

IPC: AuroraWriteForwardConsistencyPoint

L'evento `IPC:AuroraWriteForwardConsistencyPoint` descrive il tempo di attesa di una query generata da un nodo sul cluster di database secondario affinché i risultati delle operazioni di scrittura inoltrate vengano replicati nella regione attuale. Questo evento viene generato solo se il parametro `apg_write_forward.consistency_mode` a livello di sessione è impostato su uno dei seguenti:

- `SESSION`: le query su un nodo secondario attendono i risultati di tutte le modifiche apportate in quella sessione.
- `GLOBAL`: le query su un nodo secondario attendono i risultati delle modifiche apportate da quella sessione, oltre a tutte le modifiche richieste dalla regione primaria e dalle altre regioni secondarie del cluster globale.

Per ulteriori informazioni sull'impostazione del parametro `apg_write_forward.consistency_mode`, consulta the section called “Parametri di configurazione per l'inoltro di scrittura in Aurora PostgreSQL”.

Probabili cause di aumento delle attese

Alcune cause comuni dei tempi di attesa più lunghi sono:

- Aumento del ritardo di replica, misurato dalla metrica di Amazon CloudWatch `ReplicaLag`. Per ulteriori informazioni su questa metrica, consulta Monitoraggio della replica Aurora PostgreSQL..
- Aumento del carico sul nodo di scrittura della regione primaria o sul nodo secondario.

Azioni

Modifica la modalità di coerenza in base ai requisiti dell'applicazione.

IPC: AuroraWriteForwardExecute

L'evento `IPC:AuroraWriteForwardExecute` si verifica quando un processo di backend sul cluster di database secondario è in attesa del completamento di una query inoltrata e di ottenere risultati dal nodo di scrittura del cluster di database primario.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- Recupero di un numero elevato di righe dal nodo di scrittura primario della regione.
- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Ottimizza le query per recuperare solo i dati necessari.

- Ottimizza le operazioni DML (Data Manipulation Language) per modificare solo i dati necessari.
- Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC: AuroraWriteForwardGetGlobalConsistencyPoint

L'evento `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` si verifica quando un processo di backend sul cluster di database secondario che utilizza la modalità di coerenza GLOBAL è in attesa di ottenere il punto di coerenza globale dal nodo di scrittura prima di eseguire una query.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Modifica la modalità di coerenza in base ai requisiti dell'applicazione.
- Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC: AuroraWriteForwardXactAbort

L'evento `IPC:AuroraWriteForwardXactAbort` si verifica quando un processo di backend sul cluster di database secondario è in attesa del risultato di una query di pulizia remota. Le query di pulizia vengono emesse per riportare il processo allo stato ottimale dopo l'interruzione di una transazione di scrittura inoltrata. Amazon Aurora le esegue perché è stato rilevato un errore o perché un utente ha chiamato un comando `ABORT` esplicito o ha annullato una query in esecuzione.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query di pulizia dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Consigliamo azioni diverse a seconda delle cause dell'evento di attesa.

- Indaga la causa della transazione interrotta.
- Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC: AuroraWriteForwardXactCommit

L'evento `IPC:AuroraWriteForwardXactCommit` si verifica quando un processo di backend sul cluster di database secondario è in attesa del risultato di un comando di transazione di commit inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

IPC: AuroraWriteForwardXactStart

L'evento `IPC:AuroraWriteForwardXactStart` si verifica quando un processo di backend sul cluster di database secondario è in attesa del risultato di un comando di avvio della transazione inoltrato.

Probabili cause di aumento delle attese

Alcune cause comuni dell'aumento dei tempi di attesa sono:

- L'aumento della latenza di rete tra il nodo secondario e il nodo di scrittura della regione primaria ritarda la ricezione dei dati del nodo di scrittura da parte del nodo secondario.
- L'aumento del carico sul nodo secondario può ritardare la trasmissione della richiesta di query dal nodo secondario al nodo di scrittura della regione primaria.
- Un carico maggiore sul nodo di scrittura della regione primaria può ritardare la trasmissione dei dati dal nodo di scrittura al nodo secondario.

Azioni

Se il nodo secondario o il nodo di scrittura della regione primaria è limitato dalla CPU o dalla larghezza di banda di rete, valuta la possibilità di modificarlo in un tipo di istanza con maggiore capacità di CPU o maggiore larghezza di banda di rete.

Utilizzo dello switchover o del failover in Amazon Aurora Global Database

La funzionalità Aurora Global Database offre una maggiore protezione di continuità aziendale e disaster recovery (BCDR) rispetto all'elevata disponibilità standard fornita da un cluster Aurora DB in un unico cluster. Regione AWS Utilizzando Aurora Global Database, è possibile pianificare un ripristino più rapido da disastri regionali rari e non pianificati o interruzioni complete dei livelli di servizio in tempi rapidi.

È possibile consultare le seguenti linee guida e procedure per pianificare, testare e implementare la strategia BCDR utilizzando la funzionalità Aurora Global Database.

Argomenti

- Pianificazione della continuità aziendale e del disaster recovery
- Esecuzione di switchover per database globali Amazon Aurora
- Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata
- Gestione RPOs di database globali basati su Aurora PostgreSQL
- Resilienza interregionale per i cluster secondari di Global Database

Pianificazione della continuità aziendale e del disaster recovery

Per pianificare la strategia di continuità aziendale e disaster recovery, è utile comprendere la seguente terminologia di settore e il modo in cui tali termini si riferiscono alle funzionalità di Aurora Global Database.

Il ripristino di emergenza è in genere determinato dai due obiettivi aziendali seguenti:

- Obiettivo del tempo di ripristino (RTO): il tempo necessario a un sistema per tornare a uno stato funzionante dopo un guasto o un'interruzione del servizio. In altre parole, l'RTO misura i tempi di inattività. Per Aurora Global Database, l'RTO può essere dell'ordine dei minuti.
- Obiettivo del punto di ripristino (RPO): la quantità di dati che può venire persa (misurata nel tempo) dopo un guasto o un'interruzione del servizio. Questa perdita di dati è in genere dovuta al ritardo della replica asincrona. Per un database globale Aurora, l'RPO viene in genere misurato in secondi. Con un database globale basato su Aurora PostgreSQL–, puoi utilizzare il parametro `rds.global_db_rpo` per impostare e tenere traccia del limite superiore su RPO, ma ciò potrebbe influire sull'elaborazione delle transazioni sul nodo writer del cluster primario. Per ulteriori informazioni, consulta Gestione RPOs di database globali basati su Aurora PostgreSQL.

L'esecuzione di uno switchover o di un failover con Aurora Global Database implica la promozione di un cluster DB secondario come cluster DB principale. Il termine "interruzione a livello regionale" viene spesso utilizzato per descrivere una serie di scenari di errore. Lo scenario peggiore potrebbe essere un'interruzione generalizzata causata da un evento catastrofico che interessa un'area geografica particolarmente estesa. Tuttavia, la maggior parte delle interruzioni è molto più localizzata e riguarda solo un piccolo sottoinsieme di servizi cloud o sistemi dei clienti. Valuta l'ambito dell'interruzione nel suo insieme per assicurarti che il failover tra regioni rappresenti la soluzione più appropriata e scegli il metodo di failover più adatto alla situazione. L'utilizzo del failover o dello switchover dipende dallo scenario di interruzione specifico:

- Failover: utilizza questo approccio per il ripristino dopo un'interruzione non pianificata. Con questo approccio, puoi eseguire un failover tra regioni su uno dei cluster database secondari del database globale Aurora. Il valore RPO, espresso in secondi, per questo approccio è in genere diverso da zero. La quantità di perdita di dati dipende dal ritardo di replica del database globale di Aurora Regioni AWS al momento dell'errore. Per ulteriori informazioni, consulta Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.
- Switchover: questa operazione era precedentemente denominata «failover pianificato gestito». Utilizza questo approccio per scenari controllati, come la manutenzione operativa e altre procedure operative pianificate, in cui tutti i cluster Aurora e gli altri servizi con cui interagiscono sono integri. Poiché questa funzionalità sincronizza i cluster di database secondari con il primario prima di apportare altre modifiche, l'RPO è 0 (nessuna perdita di dati). Per ulteriori informazioni, consulta Esecuzione di switchover per database globali Amazon Aurora.

> **Note**
>
> Prima di poter eseguire uno switchover o un failover su un cluster Aurora DB secondario headless, è necessario aggiungervi un'istanza DB. Per ulteriori informazioni sui cluster database headless, consulta Creazione di un cluster database Aurora headless in una regione secondaria.

Esecuzione di switchover per database globali Amazon Aurora

> **Note**
>
> Gli switchover erano precedentemente denominati failover pianificati gestiti.

Utilizzando gli switchover, è possibile modificare la regione del cluster primario su base regolare. Questo approccio è destinato agli scenari controllati, ad esempio durante la manutenzione operativa e altre procedure operative pianificate.

Esistono tre casi d'uso comuni per l'utilizzo degli switchover.

- Per i requisiti relativi alla "rotazione regionale" imposti a settori specifici. Ad esempio, le normative sui servizi finanziari potrebbero imporre che i sistemi di livello 0 passino a un'altra regione per diversi mesi per garantire l'esecuzione regolare delle procedure di ripristino di emergenza.

- Per applicazioni "" multiregionali. follow-the-sun Ad esempio, un'azienda potrebbe voler fornire scritture con latenza inferiore in diverse regioni in base all'orario di lavoro nei vari fusi orari.
- Come zero-data-loss metodo per tornare alla regione principale originale dopo un failover.

> **Note**
>
> Gli switchover sono progettati per essere utilizzati su un database globale Aurora in cui tutti i cluster Aurora e gli altri servizi con cui interagiscono sono integri. Per eseguire il ripristino da un'interruzione non pianificata, puoi eseguire la procedura appropriata descritta in Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata.
> È possibile eseguire switchover gestiti tra regioni con Aurora Global Database solo se i cluster DB primari e secondari hanno le stesse versioni del motore principale e secondario. A seconda del motore e delle versioni del motore, potrebbe essere necessario che i livelli di patch siano identici oppure i livelli di patch possono essere diversi. Per un elenco dei motori e delle versioni del motore che consentono queste operazioni tra cluster primari e secondari con diversi livelli di patch, vedereCompatibilità del livello di patch per switchover e failover gestiti tra regioni. Prima di iniziare lo switchover, controlla le versioni del motore nel cluster globale per assicurarti che supportino lo switchover gestito tra regioni e, se necessario, aggiornale.

Durante uno switchover, Aurora imposta il cluster nella regione secondaria prescelta come cluster principale. Il meccanismo di switchover mantiene la topologia di replica esistente del database globale: ha ancora lo stesso numero di cluster Aurora nelle stesse regioni. Prima di avviare il processo di switchover, Aurora attende che i cluster Region secondari di destinazione siano completamente sincronizzati con il Region cluster primario. Quindi, il cluster DB nella regione primaria diventa di sola lettura. Il cluster secondario scelto promuove uno dei suoi nodi di sola lettura allo stato di scrittore completo, il che consente a tale cluster secondario di assumere il ruolo di cluster primario. Poiché il cluster secondario di destinazione è stato sincronizzato con quello primario all'inizio del processo, il nuovo cluster primario continua le operazioni per il database globale Aurora senza perdere alcun dato. Il database non è disponibile per un breve periodo, mentre i cluster primario e secondario selezionati assumono i loro nuovi ruoli.

> **Note**
>
> Per gestire gli slot di replica per Aurora PostgreSQL dopo aver eseguito uno switchover, vedere. Gestione degli slot logici per Aurora Postgre SQL

Per ottimizzare la disponibilità delle applicazioni, è consigliabile eseguire le seguenti operazioni prima di utilizzare questa funzionalità:

- Esegui questa operazione durante le ore non di punta o in un altro momento quando le scritture nel cluster DB primario sono minime.
- Controllare i tempi di ritardo per tutti i cluster di database Aurora secondari nel database globale Aurora. Per tutti i database globali basati su Aurora PostgreSQL e per i database globali basati su Aurora MySQL a partire dalle versioni del motore 3.04.0 e successive o 2.12.0 e successive, usa Amazon per visualizzare la metrica per tutti i cluster DB secondari. CloudWatch `AuroraGlobalDBRPOLag` Per le versioni minori precedenti dei database globali basati su Aurora MySQL, puoi invece visualizzare la metrica `AuroraGlobalDBReplicationLag`. Questa metrica indica il ritardo (in millisecondi) della replica tra un cluster secondario e il cluster database primario. Il suo valore è direttamente proporzionale al tempo necessario ad Aurora per completare lo switchover. Di conseguenza, maggiore è il valore del ritardo, maggiore sarà il tempo necessario per lo switchover. Quando esamini queste metriche, fallo dal cluster primario corrente.

  Per ulteriori informazioni sulle CloudWatch metriche per Aurora, consulta. Parametri a livello di cluster per Amazon Aurora
- Il cluster DB secondario promosso durante uno switchover potrebbe avere impostazioni di configurazione diverse rispetto al vecchio cluster DB primario. Si consiglia di mantenere coerenti i seguenti tipi di impostazioni di configurazione in tutti i cluster dei cluster di database globali Aurora. In questo modo è possibile ridurre al minimo i problemi di prestazioni, le incompatibilità dei carichi di lavoro e altri comportamenti anomali dopo uno switchover.
  - Configura il gruppo di parametri del cluster Aurora DB per il nuovo primario, se necessario: quando promuovi un cluster DB secondario per assumere il ruolo principale, il gruppo di parametri del secondario potrebbe essere configurato in modo diverso rispetto a quello primario. In tal caso, modifica il gruppo di parametri del cluster di database secondario promosso in modo che sia conforme alle impostazioni del cluster primario. Per scoprire come, consulta Modifica dei parametri per un database globale Aurora.

- Configura strumenti e opzioni di monitoraggio, come Amazon CloudWatch Events e allarmi: configura il cluster DB promosso con la stessa capacità di registrazione, gli allarmi e così via necessari per il database globale. Come per i gruppi di parametri, la configurazione di queste funzionalità non viene ereditata dal ruolo primario durante il processo di switchover. Alcune CloudWatch metriche, come il ritardo di replica, sono disponibili solo per le regioni secondarie. Pertanto, uno switchover modifica il modo in cui visualizzare tali metriche e impostare i relativi allarmi e potrebbe richiedere modifiche da apportare a qualsiasi dashboard predefinito. Per ulteriori informazioni sui cluster e sul monitoraggio di Aurora DB, vedere. Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch
- Configura le integrazioni con altri AWS servizi: se il tuo database globale Aurora si integra AWS con servizi AWS Secrets Manager come AWS Identity and Access Management Amazon S3 AWS Lambda e, assicurati di configurare le integrazioni con questi servizi secondo necessità. Per ulteriori informazioni sull'integrazione dei database globali Aurora con IAM, Amazon S3 e Lambda, consulta Utilizzo di Amazon Aurora global database con altri servizi AWS. Per ulteriori informazioni su Secrets Manager, vedi Come automatizzare la replica dei segreti in AWS Secrets Manager across. Regioni AWS

Se si utilizza l'endpoint Aurora Global Database writer, non è necessario modificare le impostazioni di connessione nell'applicazione. Verifica che le modifiche al DNS si siano propagate e che sia possibile connettersi ed eseguire operazioni di scrittura sul nuovo cluster primario. Potrai quindi riprendere la piena operatività dell'applicazione.

Supponiamo che le connessioni alle applicazioni utilizzino l'endpoint cluster del vecchio cluster primario, anziché l'endpoint global writer. In tal caso, assicuratevi di modificare le impostazioni di connessione dell'applicazione per utilizzare l'endpoint cluster del nuovo cluster primario. Se hai accettato i nomi forniti al momento della creazione del database globale Aurora, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster promosso nell'applicazione. Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando tale cluster viene promosso a primario.

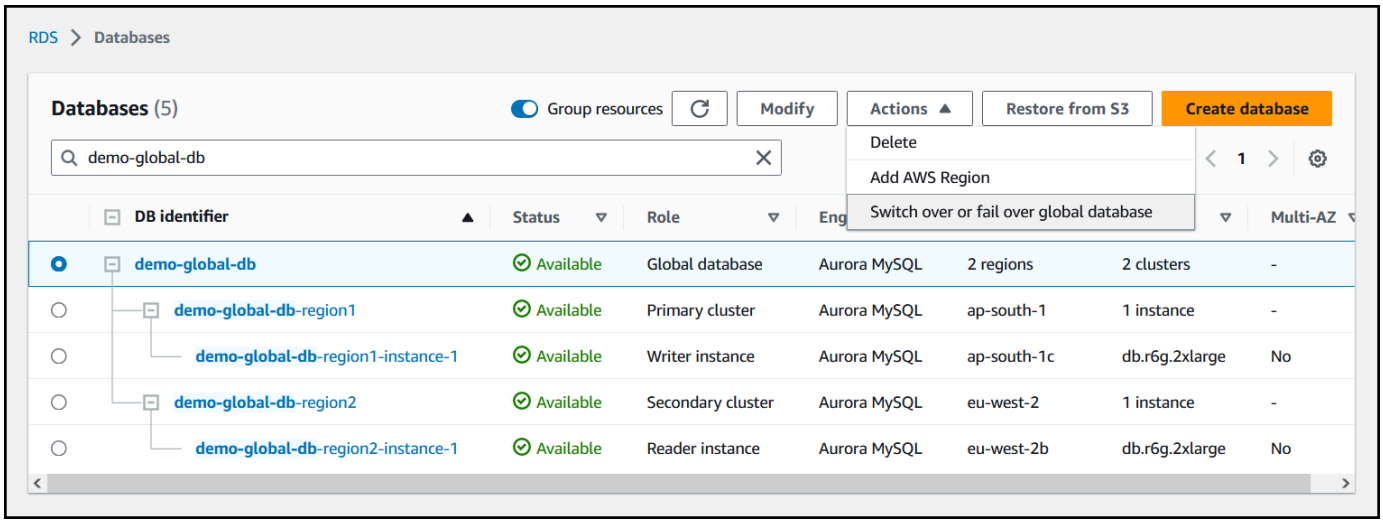
Se utilizzi RDS Proxy, assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint appropriato. read/write endpoint of the proxy that's associated with the new primary cluster. This proxy endpoint might be the default endpoint or a custom read/write Per ulteriori informazioni, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

È possibile eseguire uno switchover di Aurora Global Database utilizzando l'API AWS Management Console AWS CLI, the o RDS.

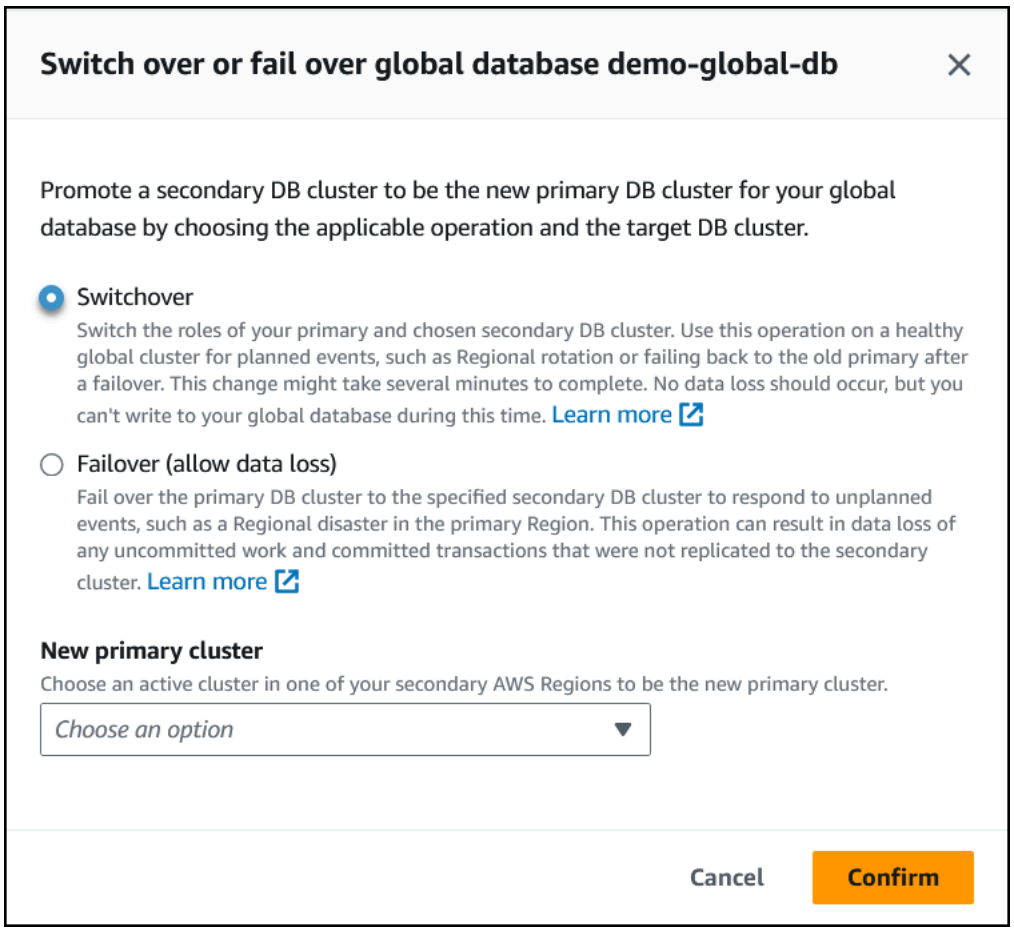
Console

Esecuzione dello switchover nel database globale Aurora

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database e trova il database globale Aurora in cui intendi eseguire lo switchover.
3. Scegli Switchover o failover database globale nel menu Operazioni.

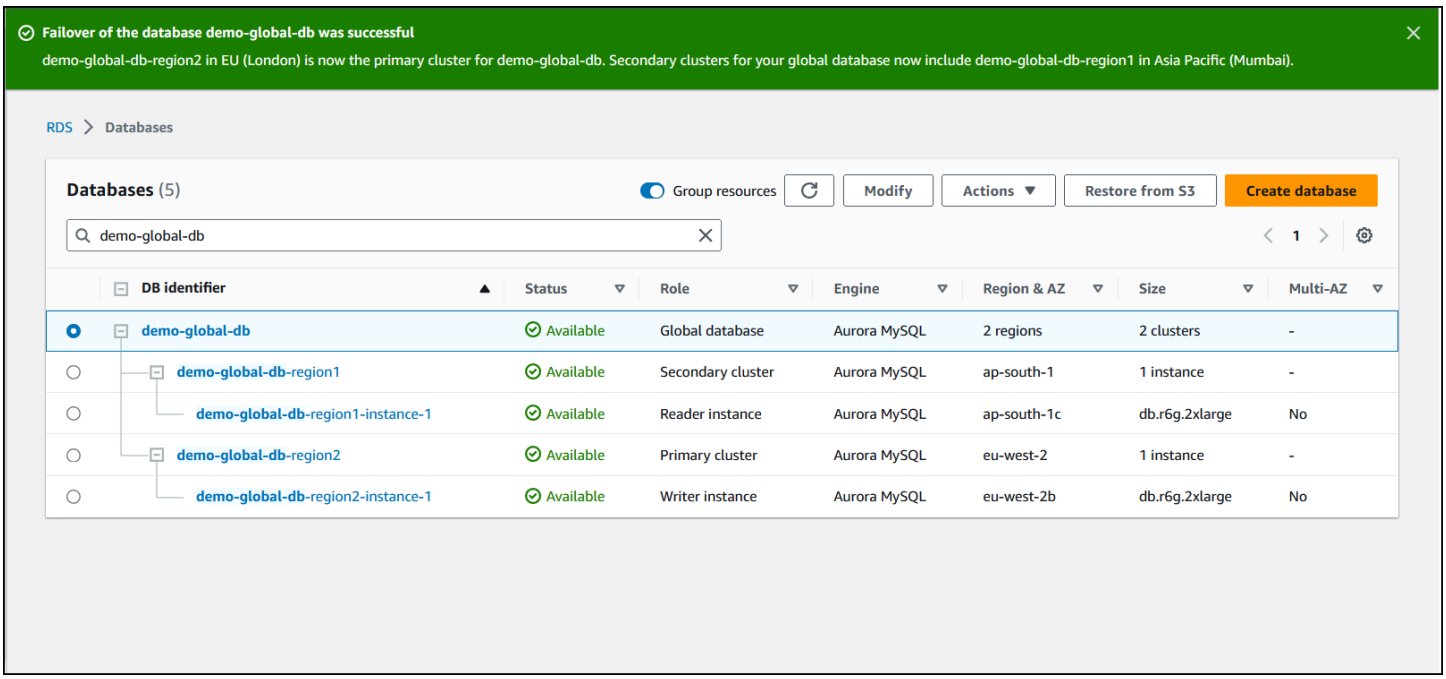


4. Scegli Switchover.



5. In Nuovo cluster primario, scegli un cluster attivo in una delle Regioni AWS secondarie da promuovere a nuovo cluster primario.
6. Scegli Conferma.

Al termine dello switchover, potrai visualizzare i cluster database Aurora e il relativo stato corrente nell'elenco Database, come illustrato nella figura seguente.



AWS CLI

Esecuzione dello switchover nel database globale Aurora

Utilizzate il comando `switchover-global-cluster` CLI per eseguire uno switchover per Aurora Global Database. Questo comando consente di passare i valori per i seguenti parametri.

- `--region`— Specificare Regione AWS dove è in esecuzione il cluster DB primario del database globale Aurora.
- `--global-cluster-identifier` – Specifica il nome del database globale Aurora.
- `--target-db-cluster-identifier` – Specifica l'Amazon Resource Name (ARN) del cluster di database Aurora che si desidera promuovere come principale per il database globale Aurora.

Per LinuxmacOS, oUnix:

```
aws rds --region region_of_primary \
   switchover-global-cluster --global-cluster-identifier global_database_id \
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

Per Windows:

```
aws rds --region region_of_primary ^
   switchover-global-cluster --global-cluster-identifier global_database_id ^
```

```
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

API RDS

Per eseguire uno switchover per Aurora Global Database, esegui l'operazione API. SwitchoverGlobalCluster

Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata

In rare occasioni, il database globale Aurora potrebbe subire un'interruzione imprevista del database primario. Regione AWS In questo caso, il cluster database Aurora primario e il relativo nodo di scrittura non sono disponibili e la replica tra il cluster database primario e secondari viene interrotta. Per ridurre al minimo i tempi di inattività (RTO) e la perdita di dati (RPO), puoi eseguire un failover tra regioni.

Aurora Global Database dispone di due metodi di failover che è possibile utilizzare in una situazione di disaster recovery:

- Failover gestito: questo metodo è consigliato in situazioni che prevedono il ripristino di emergenza. Quando si utilizza questo metodo, Aurora aggiunge automaticamente la vecchia regione primaria al database globale come regione secondaria quando diventa nuovamente disponibile. Pertanto, viene mantenuta la topologia originale del cluster globale. Per informazioni su come utilizzare questo metodo, consulta Esecuzione di failover gestiti per database globali Aurora.
- Failover manuale: questo metodo alternativo può essere utilizzato quando il failover gestito non è un'opzione, ad esempio quando le regioni primarie e secondarie utilizzano versioni del motore non compatibili. Per informazioni su come utilizzare questo metodo, consulta Esecuzione di failover gestiti per database globali Aurora.

> **Important**
>
> Entrambi i metodi di failover possono comportare la perdita dei dati delle transazioni di scrittura che non sono stati replicati sul dispositivo secondario scelto prima che si verificasse l'evento di failover. Tuttavia, il processo di ripristino che promuove un'istanza database sul cluster database secondario scelto come istanza database di scrittura principale garantisce che i dati si trovino in uno stato transazionale coerente. I failover sono inoltre suscettibili a problemi di split-brain.

Esecuzione di failover gestiti per database globali Aurora

Questo approccio è destinato alla continuità aziendale in caso di una reale emergenza a livello regionale o di un'interruzione completa del livello di servizio.

Durante un failover gestito, il cluster secondario nella regione secondaria prescelta diventa il nuovo cluster primario. Il cluster secondario scelto promuove uno dei suoi nodi di sola lettura allo stato di istanza di scrittura completa. Questo passaggio consente al cluster di assumere il ruolo di cluster primario. Il database non sarà disponibile per un breve periodo di tempo mentre il cluster sta assumendo il suo nuovo ruolo. Non appena la vecchia regione primaria è di nuovo integra e disponibile, Aurora la aggiunge automaticamente al cluster globale come regione secondaria. Pertanto, viene mantenuta la topologia di replica esistente del database globale Aurora.

> **Note**
>
> Per gestire gli slot di replica per Aurora PostgreSQL dopo aver eseguito un failover, vedere. Gestione degli slot logici per Aurora Postgre SQL

> **Note**
>
> È possibile eseguire failover gestiti tra regioni con Aurora Global Database solo se i cluster DB primari e secondari hanno le stesse versioni del motore principale e secondario. A seconda del motore e delle versioni del motore, potrebbe essere necessario che i livelli di patch siano identici oppure i livelli di patch possono essere diversi. Per un elenco dei motori e delle versioni del motore che consentono queste operazioni tra cluster primari e secondari con diversi livelli di patch, vedereCompatibilità del livello di patch per switchover e failover gestiti tra regioni. Prima di iniziare il failover, controllate le versioni del motore nel vostro cluster globale per assicurarvi che supportino lo switchover gestito tra regioni e, se necessario, aggiornatele. Se le versioni del motore richiedono livelli di patch identici ma utilizzano livelli di patch diversi, è possibile eseguire il failover manualmente seguendo la procedura riportata di seguito. Esecuzione di failover gestiti per database globali Aurora

Il failover gestito non attende la sincronizzazione dei dati tra la regione secondaria scelta e la regione principale corrente. Poiché Aurora Global Database replica i dati in modo asincrono, è possibile che non tutte le transazioni siano state replicate AWS nella regione secondaria scelta prima che questa venga promossa ad accettare funzionalità complete di lettura/scrittura.

Per garantire che i dati siano in uno stato coerente, Aurora crea un nuovo volume di archiviazione per la vecchia regione primaria dopo il ripristino. Prima di creare il nuovo volume di archiviazione nella AWS regione, Aurora tenta di scattare un'istantanea del vecchio volume di archiviazione nel punto in cui si è verificato l'errore. In questo modo, è possibile ripristinare l'istantanea e recuperare i dati mancanti in essa contenuti. Se questa operazione ha esito positivo, Aurora inserisce questa istantanea denominata `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`* nella sezione snapshot di. AWS Management ConsoleÈ inoltre possibile utilizzare il `describe-db-cluster-snapshots` AWS CLI comando o l'operazione `DescribeDBClusterSnapshots` API per visualizzare i dettagli dell'istantanea.

Quando si avvia un failover gestito, Aurora tenta anche di arrestare il traffico di scrittura attraverso il livello di archiviazione Aurora ad alta disponibilità. Ci riferiamo a questo meccanismo come «write fencing». Se il processo ha esito positivo, Aurora emette un evento RDS che indica che le scritture sono state interrotte. Nell'improbabile eventualità che si verifichino più errori AZ in una regione, è possibile che il processo di write fencing non riesca tempestivamente. In tal caso, Aurora emette un evento RDS che informa l'utente che il processo di interruzione delle scritture è scaduto. Se il vecchio cluster primario è raggiungibile sulla rete, Aurora registra questi eventi lì. In caso contrario, Aurora registra gli eventi sul nuovo cluster primario. Per ulteriori informazioni su questi eventi, consultaEventi di cluster di database. Poiché la scrittura di scherma è il miglior tentativo possibile, è possibile che le scritture vengano momentaneamente accettate nella vecchia regione primaria, con conseguenti problemi cerebrali divisi.

Si consiglia di completare le seguenti attività prima di eseguire un failover con Aurora Global Database. In questo modo si riduce al minimo la possibilità di problemi di split-brain o il ripristino di dati non replicati dall'istantanea del vecchio cluster primario.

- Per evitare che le scritture vengano inviate al cluster primario di Aurora Global Database, metti le applicazioni offline.
- Assicurati che tutte le applicazioni che si connettono al cluster DB primario utilizzino l'endpoint global writer. Questo endpoint ha un valore che rimane invariato anche quando una nuova regione diventa il cluster principale a causa dello switchover o del failover. Aurora implementa misure di protezione aggiuntive per ridurre al minimo la possibilità di perdita di dati per le operazioni di scrittura inviate tramite l'endpoint globale. Per ulteriori informazioni sugli endpoint di Global Writer, consulta. Connessione al database globale di Amazon Aurora
- Se utilizzi l'endpoint global writer e i livelli applicativi o di rete memorizzano nella cache i valori DNS, riduci il time-to-live (TTL) della cache DNS a un valore basso, ad esempio 5 secondi. In questo modo, l'applicazione registra rapidamente le modifiche DNS con l'endpoint global writer.

Sebbene Aurora tenti di bloccare le scritture nella vecchia regione primaria, non è garantito che l'azione abbia successo. La riduzione della durata della cache DNS riduce ulteriormente la probabilità di problemi di split-brain. In alternativa, puoi verificare la presenza dell'evento RDS che ti informa quando Aurora ha osservato le modifiche al DNS per l'endpoint global writer. In questo modo, puoi verificare che l'applicazione abbia registrato anche la modifica DNS prima di riavviare il traffico di scrittura dell'applicazione.

- Controlla i tempi di ritardo per tutti i cluster Aurora DB secondari nell'Aurora Global Database. La scelta della regione secondaria con il minor ritardo di replica può ridurre al minimo la perdita di dati relativamente all'attuale regione primaria in stato di errore.

  Per tutte le versioni dei database globali basati su Aurora PostgreSQL e per i database globali basati su Aurora MySQL a partire dalle versioni del motore 3.04.0 e successive o 2.12.0 e successive, usa Amazon per visualizzare la metrica per tutti i cluster DB secondari. CloudWatch `AuroraGlobalDBRPOLag` Per le versioni minori precedenti dei database globali basati su Aurora MySQL, puoi invece visualizzare la metrica `AuroraGlobalDBReplicationLag`. Questa metrica indica il ritardo (in millisecondi) della replica tra un cluster secondario e il cluster database primario.

  Per ulteriori informazioni sulle CloudWatch metriche per Aurora, consulta. Parametri a livello di cluster per Amazon Aurora

Durante un failover gestito, il cluster database secondario scelto viene promosso al nuovo ruolo primario. Tuttavia, non eredita le varie opzioni di configurazione del cluster di database primario. Una mancata corrispondenza nella configurazione può causare problemi di prestazioni, incompatibilità dei carichi di lavoro e altri comportamenti anomali. Per evitare tali problemi, è consigliabile risolvere le differenze tra i cluster di database globali Aurora per quanto segue:

- Configura il gruppo di parametri del cluster Aurora DB per il nuovo cluster primario, se necessario: puoi configurare i gruppi di parametri del cluster Aurora DB in modo indipendente per ogni cluster Aurora nell'Aurora Global Database. Pertanto, quando si promuove un cluster database secondario perché assuma il ruolo primario, il gruppo di parametri dal cluster secondario potrebbe essere configurato in modo diverso rispetto al cluster primario. In tal caso, modifica il gruppo di parametri del cluster di database secondario promosso in modo che sia conforme alle impostazioni del cluster primario. Per scoprire come, consulta Modifica dei parametri per un database globale Aurora.
- Configura strumenti e opzioni di monitoraggio, come Amazon CloudWatch Events e allarmi: configura il cluster DB promosso con la stessa capacità di registrazione, gli allarmi e così via necessari per il database globale. Come per i gruppi di parametri, la configurazione di queste

funzionalità non viene ereditata dal primario durante il processo di failover. Alcune CloudWatch metriche, come il ritardo di replica, sono disponibili solo per le regioni secondarie. Pertanto, un failover modifica il modo in cui visualizzare tali metriche e impostare i relativi allarmi e potrebbe richiedere modifiche da apportare a qualsiasi dashboard predefinito. Per ulteriori informazioni sul monitoraggio dei cluster Aurora DB, vedere. Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch

- Configura le integrazioni con altri AWS servizi: se Aurora Global Database si integra con AWS altri servizi, AWS Secrets Manager come AWS Identity and Access Management Amazon S3 AWS Lambda e, devi assicurarti che questi siano configurati come richiesto per l'accesso da qualsiasi regione secondaria. Per ulteriori informazioni sull'integrazione dei database globali Aurora con IAM, Amazon S3 e Lambda, consulta Utilizzo di Amazon Aurora global database con altri servizi AWS. Per ulteriori informazioni su Secrets Manager, vedi Come automatizzare la replica dei segreti in AWS Secrets Manager across. Regioni AWS

In genere, il cluster secondario scelto assume il ruolo primario entro pochi minuti. Non appena la nuova istanza Writer DB di Region principale è disponibile, puoi connettere le tue applicazioni e riprendere i tuoi carichi di lavoro. Dopo aver promosso il nuovo cluster primario, Aurora ricostruisce automaticamente tutti i cluster secondari regionali aggiuntivi.

Poiché i database globali Aurora utilizzano la replica asincrona, il ritardo di replica in ciascuna regione secondaria può variare. Aurora ricostruisce queste regioni secondarie in modo che abbiano esattamente gli stessi point-in-time dati del nuovo cluster di regioni primario. La durata dell'attività di ricostruzione completa può richiedere da alcuni minuti a diverse ore, a seconda delle dimensioni del volume di archiviazione e della distanza tra regioni. Quando i cluster regionali secondari terminano la ricostruzione in base alla nuova regione primaria, diventano disponibili per l'accesso in lettura.

Non appena la nuova istanza di scrittura primaria viene promossa e risulta disponibile, il cluster della nuova regione primaria può gestire le operazioni di lettura e scrittura per il database globale Aurora.

Se si utilizza l'endpoint globale, non è necessario modificare le impostazioni di connessione nell'applicazione. Verifica che le modifiche al DNS si siano propagate e che sia possibile connettersi ed eseguire operazioni di scrittura sul nuovo cluster primario. Potrai quindi riprendere la piena operatività dell'applicazione.

Se non stai utilizzando l'endpoint globale, assicurati di modificare l'endpoint dell'applicazione in modo che utilizzi l'endpoint del cluster per il cluster DB primario appena promosso. Se hai accettato i nomi forniti al momento della creazione del database globale Aurora, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster promosso nell'applicazione.

Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando tale cluster viene promosso a primario.

Se utilizzi RDS Proxy, assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint appropriato. read/write endpoint of the proxy that's associated with the new primary cluster. This proxy endpoint might be the default endpoint or a custom read/write Per ulteriori informazioni, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

Per ripristinare la topologia originale del cluster database globale, Aurora monitora la disponibilità della vecchia regione primaria. Non appena tale regione è di nuovo integra e disponibile, Aurora la aggiunge automaticamente al cluster globale come regione secondaria. Prima di creare il nuovo volume di archiviazione nella vecchia regione primaria, Aurora tenta di acquisire uno snapshot del vecchio volume di archiviazione nel punto in cui si è verificato l'errore. Ciò consente di usare lo snapshot per recuperare i dati mancanti. Se questa operazione ha esito positivo, Aurora crea un'istantanea denominata. `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`* È possibile trovare questa istantanea nella sezione Istantanee di. AWS Management Console Puoi anche vedere questa istantanea elencata nelle informazioni restituite dall'operazione dell'API Descrivi DBCluster Snapshots.

> **Note**
>
> Lo snapshot del vecchio volume di archiviazione è uno snapshot del sistema soggetto al periodo di conservazione del backup configurato sul vecchio cluster primario. Per conservare questo snapshot oltre il periodo di conservazione, puoi copiarlo e salvarlo come snapshot manuale. Per ulteriori informazioni sulla copia degli snapshot, inclusi i prezzi, consulta Copia di snapshot del cluster DB.

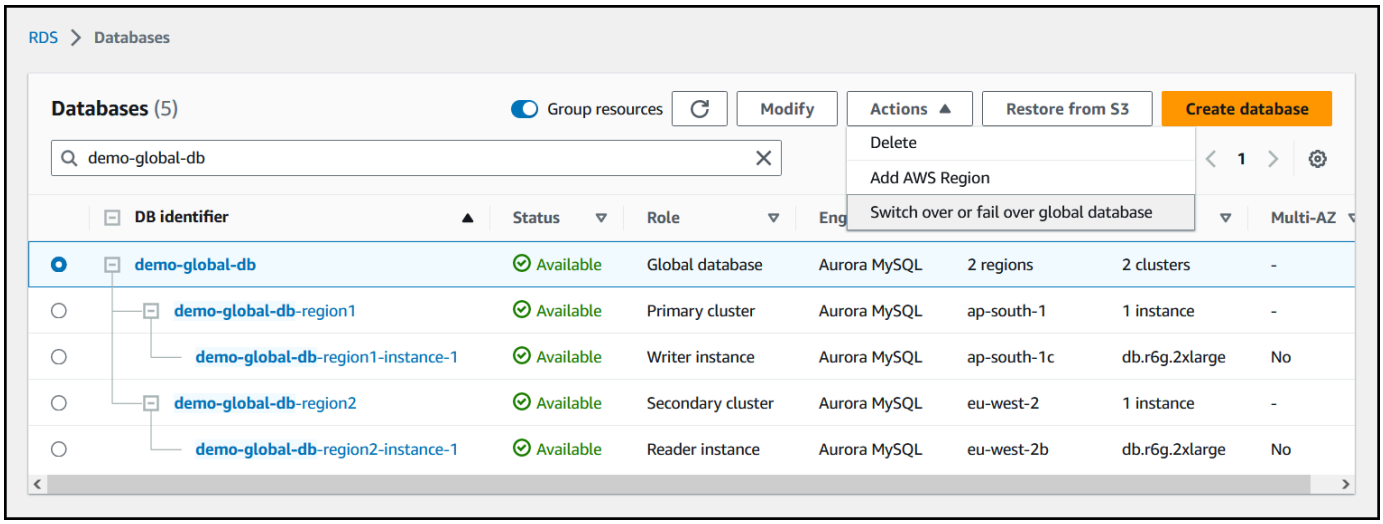
Dopo il ripristino della topologia originale, puoi eseguire il failback del database globale nella regione primaria originale eseguendo un'operazione di switchover nel momento più opportuno per l'azienda e il carico di lavoro. A tale scopo, segui la procedura in Esecuzione di switchover per database globali Amazon Aurora.

È possibile eseguire un failover con Aurora Global Database utilizzando AWS Management Console l'API, AWS CLI the o RDS.

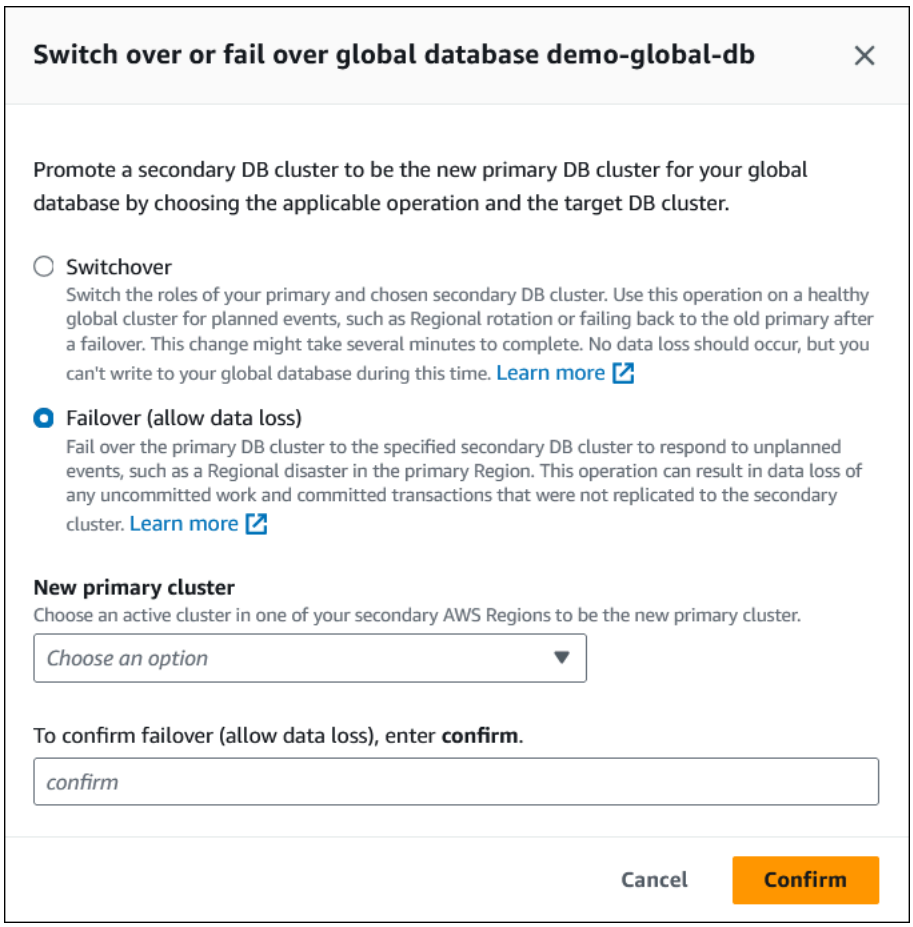
Console

Esecuzione del failover gestito nel database globale Aurora

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Database e trova il database globale Aurora in cui desideri eseguire il failover.
3. Scegli Switchover o failover database globale nel menu Operazioni.

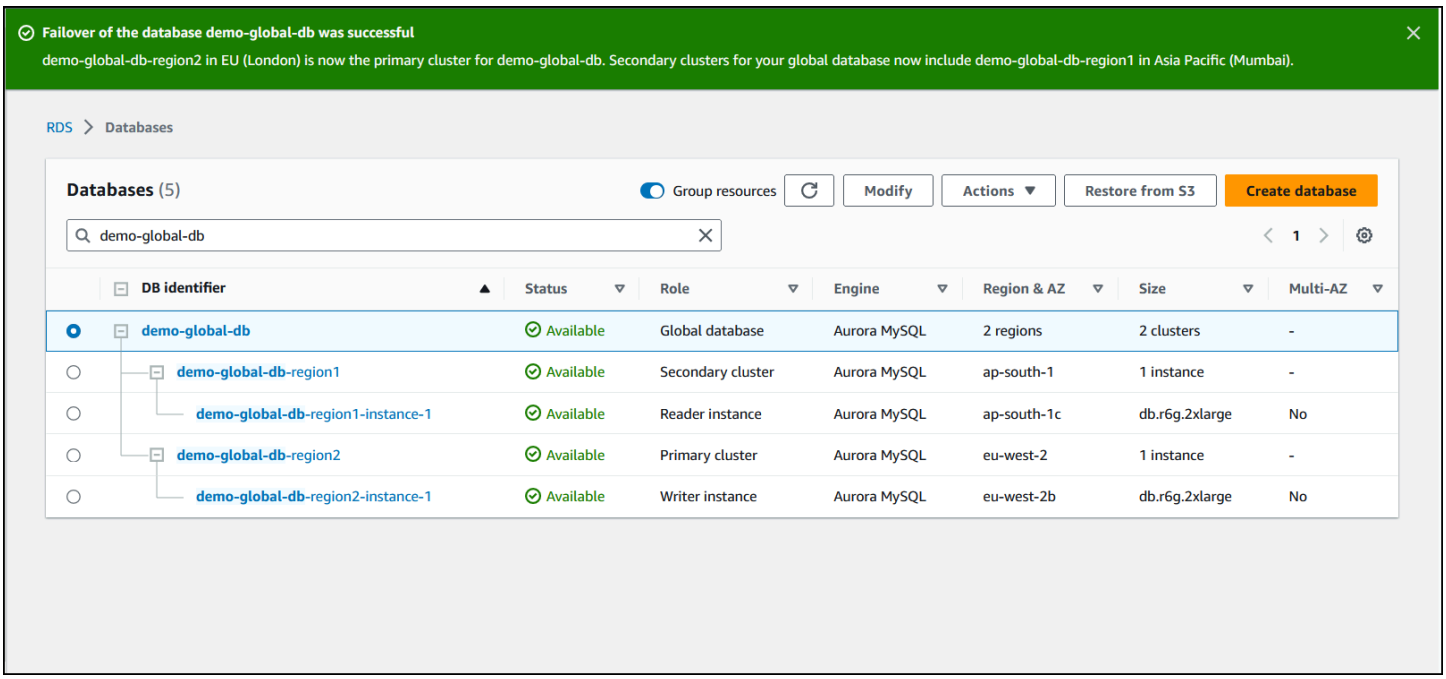


4. Scegli Failover (consenti perdita di dati).



5. In Nuovo cluster primario, scegli un cluster attivo in una delle Regioni AWS secondarie da promuovere a nuovo cluster primario.
6. Immetti **`confirm`**, quindi scegli Conferma.

Al termine del failover, è possibile visualizzare i cluster Aurora DB e il loro stato corrente nell'elenco Database, come mostrato nell'immagine seguente.



AWS CLI

Esecuzione del failover gestito nel database globale Aurora

Utilizzate il comando `failover-global-cluster` CLI per eseguire un failover con Aurora Global Database. Questo comando consente di passare i valori per i seguenti parametri.

- `--region`— Specificare la posizione Regione AWS in cui è in esecuzione il cluster DB secondario che si desidera utilizzare come nuovo primario per il database globale Aurora.
- `--global-cluster-identifier` – Specifica il nome del database globale Aurora.
- `--target-db-cluster-identifier`: specifica il nome della risorsa Amazon (ARN) del cluster database Aurora che desideri promuovere a nuovo cluster primario per il database globale Aurora.
- `--allow-data-loss`: imposta in modo esplicito un'operazione di failover anziché un'operazione di switchover. Un'operazione di failover può causare una perdita di dati se i componenti della replica asincrona non hanno completato l'invio di tutti i dati replicati alla regione secondaria.

Per LinuxmacOS, oUnix:

```
aws rds --region region_of_selected_secondary \
   failover-global-cluster --global-cluster-identifier global_database_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Per Windows:

```
aws rds --region region_of_selected_secondary ^
   failover-global-cluster --global-cluster-identifier global_database_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

API RDS

Per eseguire un failover con Aurora Global Database, esegui FailoverGlobalClusterl'operazione API.

Esecuzione di failover gestiti per database globali Aurora

In alcuni scenari, puoi non essere in grado di utilizzare il processo di failover gestito. Un esempio è quando i cluster database primario e secondari non utilizzano versioni del motore compatibili. In questo caso, è possibile seguire questa procedura manuale per eseguire un failover nella regione secondaria di destinazione.

> **Tip**
>
> Si consiglia di comprendere questo processo prima di utilizzarlo. Prepara un piano per procedere rapidamente al primo segno di un problema a livello regionale. Puoi essere pronto a identificare la regione secondaria con il minor ritardo di replica utilizzando Amazon CloudWatch regolarmente per tenere traccia dei tempi di ritardo per i cluster secondari. Assicurati di testare il piano per verificare che le procedure siano complete e accurate e che il personale sia addestrato per eseguire un failover in caso di ripristino di emergenza prima che ciò avvenga realmente.

Per eseguire un failover manuale su un cluster secondario dopo un'interruzione non pianificata nella regione principale

1. Interrompi l'emissione di istruzioni DML e altre operazioni di scrittura sul cluster di database Aurora primario nella Regione AWS con l'interruzione.
2. Identifica un cluster Aurora DB da un secondario da Regione AWS utilizzare come nuovo cluster DB primario. Se hai due o più file secondari Regioni AWS nel tuo database globale Aurora, scegli il cluster secondario con il minor ritardo di replica.
3. Scollega il cluster di database secondario scelto dal database globale Aurora.

La rimozione di un cluster di database secondario da un database globale Aurora interrompe immediatamente la replica dal primario al secondario e la promuove a cluster di database Aurora con provisioning autonomo con funzionalità di lettura/scrittura complete. Tutti gli altri cluster di database Aurora secondari associati al cluster primario nella regione con interruzione sono ancora disponibili e possono accettare chiamate dall'applicazione. Inoltre consumano risorse. Poiché si sta ricreando il database globale Aurora, rimuovi gli altri cluster di database secondari prima di creare il nuovo database globale Aurora nei passaggi seguenti. In questo modo, si evitano incongruenze di dati tra i cluster di database nel database globale Aurora (problemi di split-brain).

Per i passaggi dettagliati per lo scollegamento, consulta Rimozione di un cluster da un database globale Amazon Aurora.

4. Riconfigura l'applicazione per inviare tutte le operazioni di scrittura a questo cluster di database Aurora ora autonomo utilizzando il nuovo endpoint. Se sono stati accettati i nomi forniti al momento della creazione del database globale Aurora, puoi modificare l'endpoint rimuovendo `-ro` dalla stringa endpoint del cluster nell'applicazione.

   Ad esempio, l'endpoint del cluster secondario `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` diventa `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` quando tale cluster viene scollegato dal database globale Aurora.

   Questo cluster di database Aurora diventa il cluster primario di un nuovo database globale Aurora quando si inizia ad aggiungere regioni nel passaggio successivo.

   Se utilizzi RDS Proxy, assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint appropriato. read/write endpoint of the proxy that's associated with the new primary cluster. This proxy endpoint might be the default endpoint or a custom read/write Per ulteriori informazioni, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

5. Aggiungi un file Regione AWS al cluster DB. Quando esegui questa operazione, inizia il processo di replica da primario a secondario. Per i passaggi dettagliati per aggiungere una regione, consulta Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

6. Aggiungine altro Regioni AWS se necessario per ricreare la topologia necessaria per supportare l'applicazione.

Assicurati che le scritture delle applicazioni vengano inviate al cluster di database Aurora corretto prima, durante e dopo aver apportato queste modifiche. In questo modo, si evitano incongruenze di dati tra i cluster di database nel database globale Aurora (problemi di split-brain).

Se la riconfigurazione è avvenuta in risposta a un'interruzione in un Regione AWS Regione AWS A tale scopo, aggiungete il vecchio Regione AWS al nuovo database globale e quindi utilizzate il processo di passaggio per cambiarne il ruolo. Il database globale Aurora deve utilizzare una versione di Aurora PostgreSQL o Aurora MySQL che supporti gli switchover. Per ulteriori informazioni, consulta Esecuzione di switchover per database globali Amazon Aurora.

Gestione RPOs di database globali basati su Aurora PostgreSQL

Con un database globale basato su Aurora PostgreSQL, puoi gestire l'obiettivo del punto di ripristino (RPO) per il database globale Aurora utilizzando il parametro `rds.global_db_rpo`. RPO rappresenta la quantità massima di dati che possono essere persi in caso di interruzione.

Quando si imposta un RPO per il database globale basato su Aurora PostgreSQL–, Aurora controlla il tempo di ritardo RPO di tutti i cluster secondari per assicurarsi che almeno un cluster secondario rimanga all'interno della finestra RPO di destinazione. Il tempo di ritardo RPO è un altro parametro basato sul tempo.

L'RPO viene utilizzato quando il database riprende le operazioni in un nuovo database dopo un failover. Regione AWS Aurora valuta i tempi di ritardo RPO e RPO per eseguire il commit (o il blocco) delle transazioni sulla regione principale come segue:

- Conferma la transazione se almeno un cluster di database secondario ha un tempo di ritardo RPO inferiore rispetto all'RPO.
- Blocca la transazione se tutti i cluster di database secondari hanno tempi di ritardo RPO superiori all'RPO. Registra inoltre l'evento nel file di log di PostgreSQL ed emette eventi di “attesa” che mostrano le sessioni bloccate.

In altre parole, se tutti i cluster secondari sono dietro l'RPO di destinazione, Aurora sospende le transazioni sul cluster primario fino al raggiungimento di almeno uno dei cluster secondari. Le transazioni in pausa vengono nuovamente eseguite non appena il tempo di ritardo di almeno un cluster di database secondario diventa inferiore all'RPO. Il risultato è che nessuna transazione può eseguire il commit fino al raggiungimento dell'RPO.

Il parametro `rds.global_db_rpo` è dinamico. Se decidi che non vuoi che tutte le transazioni di scrittura si blocchino fino a quando il ritardo non diminuisce a un livello sufficiente, puoi ripristinarlo rapidamente. In questo caso, Aurora riconosce e implementa la modifica dopo un breve ritardo.

> **Important**
>
> In un database globale con solo due AWS regioni, si consiglia di mantenere il valore predefinito del `rds.global_db_rpo` parametro nel gruppo di parametri della regione secondaria. In caso contrario, l'esecuzione di un failover dovuto alla perdita della AWS regione principale potrebbe causare la sospensione delle transazioni da parte di Aurora. Attendi invece che Aurora completi la ricostruzione del cluster nella vecchia AWS regione in cui si è verificata l'errore prima di modificare questo parametro per imporre un RPO massimo.

Se imposti questo parametro come indicato di seguito, puoi anche monitorare le metriche che genera. Puoi eseguire questa operazione utilizzando `psql` o un altro strumento per interrogare il cluster di database primario del database globale Aurora e ottenere informazioni dettagliate sulle operazioni del database globale basato su Aurora PostgreSQL–. Per scoprire come, consulta Monitoraggio dei database globali basati su Aurora SQL Postgre.

Argomenti

- Impostazione dell'obiettivo del punto di ripristino
- Visualizzazione dell'obiettivo del punto di ripristino
- Disattivazione dell'obiettivo del punto di ripristino

Impostazione dell'obiettivo del punto di ripristino

Il parametro `rds.global_db_rpo` controlla l'impostazione RPO per un database PostgreSQL. Questo parametro è supportato da Aurora PostgreSQL. I valori validi per `rds.global_db_rpo` vanno da 20 secondi a 2.147.483.647 secondi (68 anni). Scegli un valore realistico per soddisfare le tue esigenze aziendali e il caso d'uso. Ad esempio, puoi consentire fino a 10 minuti per l'RPO, nel qual caso si imposta il valore su 600.

Puoi impostare questo valore per il database globale basato su Aurora PostgreSQL utilizzando la AWS Management Console, la AWS CLI o l'API RDS.

Console

Per impostare l'RPO

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Sceglie il cluster primario del database Aurora globale e apri la scheda Configurazione per trovare il relativo gruppo di parametri del cluster di database. Ad esempio, il gruppo di parametri predefinito per un cluster di database primario che esegue Aurora PostgreSQL 11.7 è `default.aurora-postgresql11`.

   I gruppi di parametri non possono essere modificati direttamente. Puoi invece procedere come descritto di seguito:

   - Crea un gruppo di parametri cluster di database personalizzato utilizzando il gruppo di parametri predefinito appropriato come punto di partenza. Ad esempio, crea un gruppo di parametri cluster di database personalizzato basato su `default.aurora-postgresql11`.
   - Nel gruppo di parametri database personalizzato, imposta il valore del parametro rds.global_db_rpo in modo da soddisfare il caso d'uso. I valori validi vanno da 20 secondi fino al valore intero massimo di 2.147.483.647 (68 anni).
   - Applica il gruppo di parametri del cluster di database modificato al cluster di database Aurora.

Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.

AWS CLI

Per impostare il `rds.global_db_rpo` parametro, utilizzare il comando CLI modify-db-cluster-parameter-group. Nel comando specifica il nome del gruppo di parametri del cluster primario e i valori per il parametro RPO.

Nell'esempio seguente l'RPO viene impostato su 600 secondi per il gruppo di parametri cluster di database primario denominato `my_custom_global_parameter_group`.

PerLinux, omacOS: Unix

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
```

```
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Per Windows:

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

API RDS

Per modificare il `rds.global_db_rpo` parametro, utilizza l'operazione dell'DBClusterParameterGroupAPI Amazon RDS Modify.

Visualizzazione dell'obiettivo del punto di ripristino

L'obiettivo del punto di ripristino (RPO) di un database globale viene memorizzato nel parametro `rds.global_db_rpo`. Puoi connettersi all'endpoint per il cluster secondario che si desidera visualizzare e utilizzare per `psql` eseguire una query sull'istanza per questo valore.

```
db-name=>show rds.global_db_rpo;
```

Se questo parametro non è impostato, la query restituirà quanto segue:

```
rds.global_db_rpo
-------------------
 -1
(1 row)
```

Questa risposta successiva proviene da un cluster di database secondario con impostazione RPO di 1 minuto.

```
rds.global_db_rpo
-------------------
 60
(1 row)
```

Puoi inoltre utilizzare la CLI per ottenere valori per scoprire se `rds.global_db_rpo` è attivo su uno qualsiasi dei cluster di database Aurora utilizzando l'interfaccia a riga di comando per ottenere i valori di tutti i parametri `user` per il cluster.

Per LinuxmacOS, oUnix:

```
aws rds describe-db-cluster-parameters \
 --db-cluster-parameter-group-name lab-test-apg-global \
 --source user
```

Per Windows:

```
aws rds describe-db-cluster-parameters ^
 --db-cluster-parameter-group-name lab-test-apg-global *
 --source user
```

Il comando restituisce un output simile al seguente per tutti i parametri `user` che non sono parametri del cluster di database `default-engine` o `system`.

```
{
    "Parameters": [
        {
            "ParameterName": "rds.global_db_rpo",
            "ParameterValue": "60",
            "Description": "(s) Recovery point objective threshold, in seconds, that
 blocks user commits when it is violated.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "20-2147483647",
            "IsModifiable": true,
            "ApplyMethod": "immediate",
            "SupportedEngineModes": [
                "provisioned"
            ]
        }
    ]
}
```

Per ulteriori informazioni sulla visualizzazione dei parametri del gruppo di parametri del cluster, consulta Visualizzazione dei valori dei parametri per un gruppo di parametri del cluster DB in Amazon Aurora.

Disattivazione dell'obiettivo del punto di ripristino

Per disabilitare l'RPO, reimpostare il parametro `rds.global_db_rpo`. È possibile reimpostare i parametri utilizzando AWS Management Console, AWS CLI o l'API RDS.

Console

Per disabilitare l'RPO

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegliere Parameter groups (Gruppi di parametri).
3. Nell'elenco scegliere il gruppo di parametri cluster DB primario.
4. Scegliere Edit parameters (Modifica parametri).
5. Scegliere la casella accanto al parametro rds.global_db_rpo.
6. Scegliere Reimposta.
7. Quando la schermata mostra Reimposta parametri nel gruppo di parametri DB, scegliere Reimposta parametri.

Per ulteriori informazioni su come reimpostare un parametro con la console, vedere Modifica dei parametri in un gruppo di parametri del cluster DB in Amazon Aurora.

AWS CLI

Per reimpostare il `rds.global_db_rpo` parametro, usa il comando reset-db-cluster-parameter-group.

Per LinuxmacOS, oUnix:

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name global_db_cluster_parameter_group \
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

Per Windows:

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name global_db_cluster_parameter_group ^
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

API RDS

Per reimpostare il `rds.global_db_rpo` parametro, utilizza l'DBClusterParameterGroupoperazione di ripristino dell'API di Amazon RDS.

Resilienza interregionale per i cluster secondari di Global Database

Le versioni 16.6, 15.10, 14.15, 13.18, 12.22 o successive di Aurora PostgreSQL e le versioni 3.09 o successive di Aurora MySQL contengono miglioramenti della disponibilità che consentono alle repliche di lettura regionali secondarie di mantenere la continuità del servizio durante eventi non pianificati come guasti hardware, interruzioni della rete tra regioni, grandi volumi di trasferimenti di dati tra i cluster e altro. AWS

Sebbene le repliche di lettura rimangano disponibili per le richieste delle applicazioni, il ritardo di replica potrebbe continuare ad aumentare fino alla risoluzione dell'evento non pianificato. È possibile monitorare il ritardo tra i cluster primari e secondari utilizzando la metrica. `AuroraGlobalDBProgressLag` CloudWatch Per misurare il end-to-end ritardo, incluso qualsiasi ritardo tra il volume del cluster e le istanze DB del cluster secondario, aggiungi i valori delle metriche e. `AuroraGlobalDBProgressLag` `AuroraReplicaLag` CloudWatch Per ulteriori informazioni sulle metriche, fare riferimento a. Riferimento alle metriche per

La disponibilità di lettura del Global Database per Aurora MySQL e le versioni precedenti di Aurora PostgreSQL potrebbe risentire di tali eventi non pianificati.

Per ulteriori informazioni sulle nuove funzionalità di Aurora PostgreSQL 16.6, 15.10, 14.15, 13.18 e 12.22, vedere PostgreSQL 16.6 nelle Note di rilascio di Aurora PostgreSQL.

Per ulteriori informazioni sulle nuove funzionalità nelle versioni 3.09 e successive di Aurora MySQL, consulta Aggiornamenti del motore di database per Amazon Aurora MySQL versione 3 nelle Note di rilascio di Aurora MySQL.

Monitoraggio di un database globale Amazon Aurora

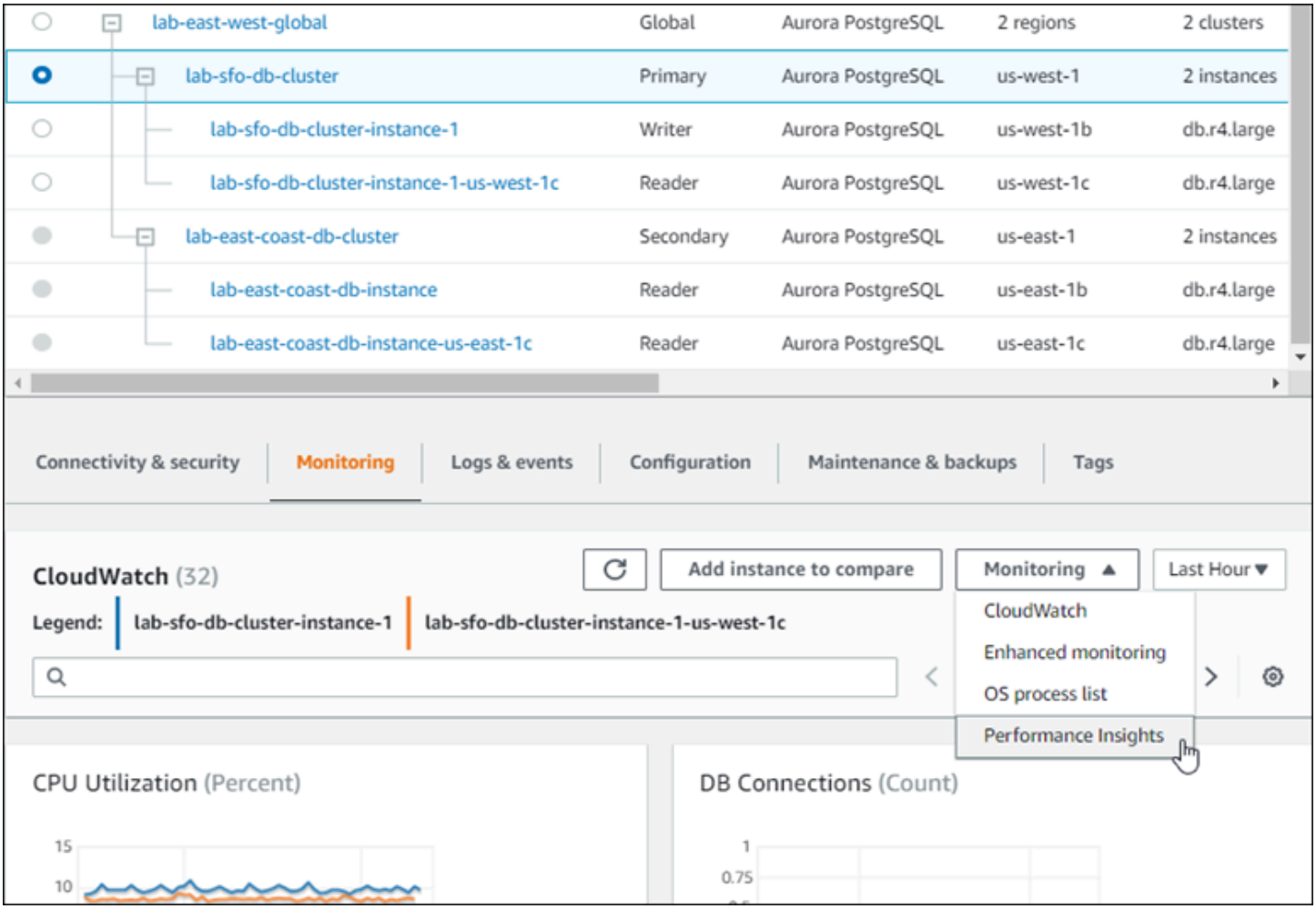
Quando si creano i cluster di database Aurora che costituiscono il database globale Aurora, è possibile scegliere molte opzioni che consentono di monitorare le prestazioni del cluster di database. Queste opzioni includono:

- Amazon RDS Performance Insights: abilita lo schema delle prestazioni nel motore di database Aurora sottostante. Per ulteriori informazioni su Performance Insights e database globali Aurora,

consulta Monitoraggio di un database globale Amazon Aurora con Amazon Performance Insights RDS.

- Monitoraggio avanzato: genera metriche per l'utilizzo di processi o thread su. CPU Per ulteriori informazioni sul monitoraggio avanzato, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.
- Amazon CloudWatch Logs: pubblica tipi di log specifici in Logs. CloudWatch I log degli errori vengono pubblicati per impostazione predefinita, ma è possibile scegliere altri log specifici per il motore di database Aurora.
  - Per i cluster Aurora DB SQL basati su Aurora My, è possibile esportare il registro di controllo, il registro generale e il registro delle query lente.
  - Per i cluster Aurora DB SQL basati su Aurora Postgre, è possibile esportare il registro Postgre. SQL
- Per i database globali SQL basati su Aurora My, è possibile eseguire query su `information_schema` tabelle specifiche per verificare lo stato del database globale Aurora e delle relative istanze. Per scoprire come, consulta Monitoraggio dei database globali SQL basati su Aurora My.
- Per i database globali SQL basati su Aurora Postgre, è possibile utilizzare funzioni specifiche per controllare lo stato del database globale di Aurora e delle relative istanze. Per scoprire come, consulta Monitoraggio dei database globali basati su Aurora SQL Postgre.

La seguente schermata mostra alcune delle opzioni disponibili nella scheda Monitoraggio di un cluster di database Aurora primario in un database globale Aurora.



Per ulteriori informazioni, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Monitoraggio di un database globale Amazon Aurora con Amazon Performance Insights RDS

Puoi usare Amazon RDS Performance Insights per i tuoi database globali Aurora. È possibile abilitare questa funzionalità singolarmente, per ogni cluster di database Aurora nel database globale Aurora. A tale scopo, scegliere Enable Performance Insights (Abilita Performance Insights) nella sezione Additional configuration (Configurazioni aggiuntive) della pagina Crea database. In alternativa, una volta operativi, è possibile modificare i cluster di database Aurora per utilizzare questa funzionalità. È possibile abilitare o disattivare Performance Insights per ciascun cluster che fa parte del database globale Aurora.

I report creati da Performance Insights si applicano a ciascun cluster del database globale. Quando aggiungi un nuovo secondario Regione AWS a un database globale Aurora che utilizza già

Performance Insights, assicurati di abilitare Performance Insights nel cluster appena aggiunto. Non eredita l'impostazione Performance Insights dal database globale esistente.

È possibile passare da un'istanza DB a un database globale Regioni AWS mentre si visualizza la pagina Performance Insights. Tuttavia, potresti non visualizzare le informazioni sulle prestazioni subito dopo la commutazione Regioni AWS. Sebbene le istanze DB possano avere nomi identici in ognuna Regione AWS, il Performance Insights associato URL è diverso per ogni istanza DB. Dopo la commutazione Regioni AWS, scegli nuovamente il nome dell'istanza DB nel riquadro di navigazione Performance Insights.

Per le istanze database associate a un database globale, i fattori che influiscono sulle prestazioni potrebbero essere diversi in ciascuna Regione AWS. Ad esempio, le istanze DB di ciascuna istanza Regione AWS potrebbero avere una capacità diversa.

Per ulteriori informazioni sull'utilizzo di Performance Insights, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Monitoraggio dei database globali Aurora con i flussi di attività di database

Con i flussi di attività del database è possibile monitorare e impostare gli allarmi per l'attività di audit nei cluster database del database globale. Avvia un flusso di attività del database su ciascun cluster database separatamente. Ciascun cluster fornisce i dati di audit al proprio flusso Kinesis all'interno della propria Regione AWS. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

Monitoraggio dei database globali SQL basati su Aurora My

Per visualizzare lo stato di un database globale SQL basato su Aurora My, interroga le tabelle information_schema.aurora_global_db_status andinformation_schema.aurora_global_db_instance_status.

> **Note**
>
> Le `information_schema.aurora_global_db_instance_status` tabelle `information_schema.aurora_global_db_status` e sono disponibili solo con i database globali di Aurora My SQL 3.04.0 e versioni successive.

Per monitorare un database globale SQL basato su Aurora My

1. Connect all'endpoint del cluster primario del database globale utilizzando un SQL client My. Per ulteriori informazioni su come connettersi, consulta Connessione al database globale di Amazon Aurora.
2. Esegui la query sulla tabella `information_schema.aurora_global_db_status` in un comando mysql per elencare i volumi primari e secondari. Questa query restituisce i tempi di ritardo dei cluster database secondari del database globale, come nell'esempio seguente.

```
mysql> select * from information_schema.aurora_global_db_status;
```

```
AWS_REGION | HIGHEST_LSN_WRITTEN | DURABILITY_LAG_IN_MILLISECONDS |
 RPO_LAG_IN_MILLISECONDS | LAST_LAG_CALCULATION_TIMESTAMP | OLDEST_READ_VIEW_TRX_ID
-----------+---------------------+--------------------------------
+-------------------------+---------------------------------
+------------------------
us-east-1  |           183537946 |                              0   |
      0 |  1970-01-01 00:00:00.000000     |               0
us-west-2  |           183537944 |                              428 |
      0 |  2023-02-18 01:26:41.925000     |               20806982
(2 rows)
```

L'output include una riga per ogni cluster DB del database globale contenente le seguenti colonne:

- AWS_REGION— In Regione AWS cui si trova questo cluster DB. Per l'elenco delle tabelle Regioni AWS per motore, vedereDisponibilità nelle regioni.
- HIGHEST_ LSN _ WRITTEN — Il numero di sequenza di log più alto (LSN) attualmente scritto su questo cluster DB.

  Un numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. LSNssono ordinati in modo tale che un LSN valore più grande rappresenti una transazione successiva.
- DURABILITY_ LAG _IN_ MILLISECONDS — La differenza nei valori di timestamp tra il `HIGHEST_LSN_WRITTEN` cluster DB secondario e il `HIGHEST_LSN_WRITTEN` cluster DB primario. Questo valore è sempre 0 sul cluster database primario del database globale Aurora.
- RPO_ LAG _IN_ MILLISECONDS — Il ritardo dell'obiettivo del punto di ripristino (). RPO Il RPO ritardo è il tempo necessario per COMMIT archiviare la transazione utente più recente

su un cluster DB secondario dopo che è stata archiviata nel cluster DB primario del database globale Aurora. Questo valore è sempre 0 sul cluster database primario del database globale Aurora.

In termini semplici, questa metrica calcola l'obiettivo del punto di ripristino per ogni cluster Aurora My SQL DB nel database globale Aurora, ovvero la quantità di dati che potrebbero andare persi in caso di interruzione. Come nel caso del ritardo, viene misurato in termini di tempo. RPO

- LAST_ _ LAG CALCULATION _ TIMESTAMP — Il timestamp che specifica quando i valori sono stati calcolati l'ultima volta per e. `DURABILITY_LAG_IN_MILLISECONDS` `RPO_LAG_IN_MILLISECONDS` Il valore temporale `1970-01-01 00:00:00+00` indica che questo è il cluster di database primario.
- OLDEST_ _ READ VIEW _ TRX _ID — L'ID della transazione più vecchia a cui l'istanza Writer DB può eliminare.

3. Esegui una query sulla tabella `information_schema.aurora_global_db_instance_status` per elencare tutte le istanze database secondarie per il cluster database primario e i cluster database secondari.

```
mysql> select * from information_schema.aurora_global_db_instance_status;
```

```
SERVER_ID            |              SESSION_ID              | AWS_REGION
 | DURABLE_LSN | HIGHEST_LSN_RECEIVED | OLDEST_READ_VIEW_TRX_ID |
 OLDEST_READ_VIEW_LSN | VISIBILITY_LAG_IN_MSEC
---------------------+--------------------------------------+------------
+-------------+----------------------+------------------------
+----------------------+------------------------
ams-gdb-primary-i2   | MASTER_SESSION_ID                    | us-east-1  |
 183537698   |                    0 |                      0 |
 0 |                     0
ams-gdb-secondary-i1 | cc43165b-bdc6-4651-abbf-4f74f08bf931 | us-west-2  |
 183537689   |            183537692 |                20806928 |
 183537682 |                      0
ams-gdb-secondary-i2 | 53303ff0-70b5-411f-bc86-28d7a53f8c19 | us-west-2  |
 183537689   |            183537692 |                20806928 |
 183537682 |                    677
ams-gdb-primary-i1   | 5af1e20f-43db-421f-9f0d-2b92774c7d02 | us-east-1  |
 183537697   |            183537698 |                20806930 |
 183537691 |                     21
(4 rows)
```

L'output include una riga per ogni istanza DB del database globale contenente le colonne seguenti:

- SERVER_ID — L'identificatore del server per l'istanza DB.
- SESSION_ID — Un identificatore univoco per la sessione corrente. Il valore di `MASTER_SESSION_ID` identifica l'istanza database di lettura (primaria).
- AWS_REGION— In Regione AWS cui si trova questa istanza DB. Per l'elenco delle tabelle Regioni AWS per motore, vedereDisponibilità nelle regioni.
- DURABLE_ LSN — Il LSN prodotto è resistente allo stoccaggio.
- HIGHEST_ LSN _ RECEIVED — Il valore più alto LSN ricevuto dall'istanza DB dell'istanza Writer DB.
- OLDEST_ READ _ VIEW _ TRX _ID — L'ID della transazione più vecchia a cui l'istanza Writer DB può eliminare.
- OLDEST_ READ _ VIEW _ LSN — La più vecchia LSN utilizzata dall'istanza DB per la lettura dallo storage.
- VISIBILITY_ LAG _IN_ MSEC — Per i lettori del cluster DB primario, in che misura questa istanza DB è in ritardo rispetto all'istanza DB di scrittura in millisecondi. Per istanze di lettura in un cluster database secondario, il ritardo in millisecondi di questa istanza database rispetto al volume secondario.

Per vedere come questi valori cambiano nel tempo, considerare il seguente blocco di transazioni in cui un inserimento di tabella richiede un'ora:

```
mysql> BEGIN;
mysql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
mysql> COMMIT;
```

In alcuni casi, potrebbe esserci una disconnessione di rete tra il cluster DB primario e il cluster DB secondario dopo l'istruzione `BEGIN`. In tal caso, il valore _ _IN_ del cluster DB secondario inizia ad aumentare. DURABILITY LAG MILLISECONDS Alla fine dell'`INSERT`istruzione, il MILLISECONDS valore DURABILITY_ LAG _IN_ è di 1 ora. Tuttavia, il MILLISECONDS valore RPO_ LAG _IN_ è 0 perché tutti i dati utente salvati tra il cluster DB primario e il cluster DB secondario sono sempre gli stessi. Non appena l'`COMMIT`istruzione viene completata, il valore RPO_ LAG MILLISECONDS _IN_ aumenta.

Monitoraggio dei database globali basati su Aurora SQL Postgre

Per visualizzare lo stato di un database globale SQL basato su Aurora Postgre, usa le funzioni and. `aurora_global_db_status` `aurora_global_db_instance_status`

> **Note**
>
> Solo Aurora Postgre SQL supporta le funzioni and. `aurora_global_db_status` `aurora_global_db_instance_status`

Per monitorare un database globale basato su Aurora SQL Postgre

1. Connect all'endpoint del cluster primario del database globale utilizzando un'SQLutilità Postgre come psql. Per ulteriori informazioni su come connettersi, consulta Connessione al database globale di Amazon Aurora.
2. Utilizzare la funzione `aurora_global_db_status` in un comando psql per elencare i volumi primari e secondari. Mostra i tempi di ritardo dei cluster DB secondari del database globale.

```
postgres=> select * from aurora_global_db_status();
```

```
aws_region | highest_lsn_written | durability_lag_in_msec | rpo_lag_in_msec |
 last_lag_calculation_time  | feedback_epoch | feedback_xmin
------------+---------------------+------------------------+-----------------
+----------------------------+----------------+--------------
us-east-1  |         93763984222 |                     -1 |              -1 |
 1970-01-01 00:00:00+00     |              0 |              0
us-west-2  |         93763984222 |                    900 |            1090 |
 2020-05-12 22:49:14.328+00 |              2 |    3315479243
(2 rows)
```

L'output include una riga per ogni cluster DB del database globale contenente le seguenti colonne:

- aws_region — In cui si trova questo cluster DB Regione AWS . Per l'elenco delle tabelle per motore Regioni AWS , vedi. Disponibilità nelle regioni
- highest_lsn_written — Il numero di sequenza di log più alto (LSN) attualmente scritto su questo cluster DB.

Un numero di sequenza di log (LSN) è un numero sequenziale univoco che identifica un record nel log delle transazioni del database. LSNssono ordinati in modo tale che un LSN valore più grande rappresenti una transazione successiva.

- durability_lag_in_msec – La differenza di timestamp tra il numero di sequenza di log più alto scritto su un cluster DB secondario (`highest_lsn_written`) e il `highest_lsn_written` sul cluster DB primario.
- rpo_lag_in_msec — Il ritardo dell'obiettivo del punto di ripristino (). RPO Questo ritardo è la differenza di tempo tra il commit delle transazioni utente più recenti memorizzate in un cluster DB secondario e il commit delle transazioni utente più recenti memorizzate nel cluster DB primario.
- last_lag_calculation_time – Il timestamp in cui sono stati calcolati i valori per `durability_lag_in_msec` e `rpo_lag_in_msec`.
- feedback_epoch – L'epoca utilizzata da un cluster di database secondario quando genera informazioni di standby a caldo.

  Hot Standby – Si verifica quando un cluster DB può connettersi e interrogare mentre il server è in modalità di ripristino o standby. Il feedback hot standby è costituito da informazioni sul cluster DB quando è in standby. Per ulteriori informazioni, consulta Hot standby nella documentazione di Postgre. SQL
- feedback_xmin – L'ID minimo della transazione attiva (meno recente) utilizzato da un cluster di database secondario.

3. Utilizzare la funzione `aurora_global_db_instance_status` per elencare tutte le istanze database secondarie sia per il cluster DB primario che per i cluster DB secondari.

```
postgres=> select * from aurora_global_db_instance_status();
```

```
server_id                                    |             session_id
  | aws_region | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------------+-------------------------------------
+------------+-------------+------------------+----------------+--------------
+----------------------+------------------------
apg-global-db-rpo-mammothrw-elephantro-1-n1 | MASTER_SESSION_ID
 | us-east-1  | 93763985102 |                  |                |              |
                    |
```

```
apg-global-db-rpo-mammothrw-elephantro-1-n2 | f38430cf-6576-479a-b296-dc06b1b1964a
 | us-east-1  | 93763985099 |      93763985102 |              2 |    3315479243 |
       93763985095 |                     10
apg-global-db-rpo-elephantro-mammothrw-n1   | 0d9f1d98-04ad-4aa4-8fdd-e08674cbbbfe
 | us-west-2  | 93763985095 |      93763985099 |              2 |    3315479243 |
       93763985089 |                   1017
(3 rows)
```

L'output include una riga per ogni istanza DB del database globale contenente le colonne seguenti:

- server_id – Identificatore del server per l'istanza DB.
- session_id – Un identificatore univoco per la sessione corrente.
- aws_region — La cartella in cui si trova Regione AWS questa istanza DB. Per l'elenco delle tabelle per motore Regioni AWS , vedi. Disponibilità nelle regioni
- durable_lsn — Il LSN prodotto è durevole nello stoccaggio.
- highest_lsn_rcvd — Il valore più alto ricevuto dall'istanza DB dall'istanza DB writer. LSN
- feedback_epoch – L'epoca utilizzata dall'istanza DB quando genera informazioni di hot standby.

  Standby a caldo è quando un'istanza database può connettersi ed eseguire query mentre il server è in modalità di ripristino o standby. Il feedback hot standby consiste in informazioni sull'istanza DB quando è in hot standby. Per ulteriori informazioni, consulta la documentazione di Postgre su Hot standby. SQL
- feedback_xmin – L'ID della transazione attiva minimo (meno recente) utilizzato dall'istanza DB.
- oldest_read_view_lsn — Il più vecchio utilizzato dall'istanza DB per leggere dallo storage. LSN
- visibility_lag_in_msec – Quanto questa istanza DB è in ritardo rispetto all'istanza DB di scrittura.

Per vedere come questi valori cambiano nel tempo, considerare il seguente blocco di transazioni in cui un inserimento di tabella richiede un'ora:

```
psql> BEGIN;
psql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
psql> COMMIT;
```

In alcuni casi, potrebbe esserci una disconnessione di rete tra il cluster DB primario e il cluster DB secondario dopo l'istruzione `BEGIN`. In tal caso, il valore `durability_lag_in_msec` del cluster DB secondario inizia ad aumentare. Alla fine dell'istruzione `INSERT`, il valore `durability_lag_in_msec` è 1 ora. Tuttavia, il valore `rpo_lag_in_msec` è 0 perché tutti i dati utente impegnati tra il cluster DB primario e il cluster DB secondario sono ancora gli stessi. Non appena l'istruzione `COMMIT` viene completata, il valore `rpo_lag_in_msec` aumenta.

Utilizzo di Amazon Aurora global database con altri servizi AWS

Puoi usare i tuoi database globali Aurora con altri AWS servizi, come Amazon S3 e. AWS Lambda Ciò richiede che tutti i cluster di database Aurora nel database globale abbiano gli stessi privilegi, funzioni esterne e così via nelle rispettive Regioni AWS. Poiché un cluster di database Aurora secondario di sola lettura in un database globale Aurora può essere promosso al ruolo primario, è consigliabile impostare i privilegi di scrittura in anticipo su tutti i cluster di database Aurora per tutti i servizi che si intende utilizzare con il database globale Aurora.

Le seguenti procedure riassumono le azioni da intraprendere per ciascuna di esse. Servizio AWS

Per richiamare AWS Lambda funzioni da un database globale Aurora

1. Per tutti i cluster Aurora che costituiscono l'Aurora Global Database, esegui le procedure in Richiamo di una funzione Lambda da un cluster Amazon Aurora My DB SQL.
2. Per ogni cluster nel database globale di Aurora, imposta il ruolo (ARN) del ruolo new IAM (IAM).
3. Per consentire agli utenti di database in un Aurora Global Database di richiamare funzioni Lambda, associa il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS a ogni cluster nell'Aurora Global Database.
4. Configura ogni cluster nell'Aurora Global Database per consentire le connessioni in uscita a Lambda. Per istruzioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

Caricare i dati da Amazon S3.

1. Per tutti i cluster Aurora che costituiscono l'Aurora Global Database, esegui le procedure in Caricamento di dati in un cluster Amazon Aurora My SQL DB da file di testo in un bucket Amazon S3.
2. Per ogni cluster Aurora nel database globale, imposta il parametro `aurora_load_from_s3_role` o il cluster `aws_default_s3_role` DB sull'Amazon

Resource Name (ARN) del nuovo IAM ruolo. Se non è specificato un IAM ruolo per `aurora_load_from_s3_role`, Aurora utilizza il IAM ruolo specificato in. `aws_default_s3_role`

3. Per consentire agli utenti di database in un Aurora Global Database di accedere ad S3, associa il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS a ogni cluster Aurora nel database globale.
4. Configura ogni cluster nell'Aurora Global Database per consentire le connessioni in uscita a S3. Per istruzioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

Per salvare i dati interrogati in Amazon S3

1. Per tutti i cluster Aurora che compongono il database globale di Aurora, esegui le procedure in o. Salvataggio di dati da un cluster Amazon Aurora My SQL DB in file di testo in un bucket Amazon S3 Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3
2. Per ogni cluster Aurora nel database globale, imposta il parametro `aurora_select_into_s3_role` o il cluster `aws_default_s3_role` DB sull'Amazon Resource Name (ARN) del nuovo IAM ruolo. Se non è specificato un IAM ruolo per `aurora_select_into_s3_role`, Aurora utilizza il IAM ruolo specificato in. `aws_default_s3_role`
3. Per consentire agli utenti di database in un Aurora Global Database di accedere ad S3, associa il ruolo creato in Creazione di un ruolo IAM per consentire ad Amazon Aurora di accedere ai servizi AWS a ogni cluster Aurora nel database globale.
4. Configura ogni cluster nell'Aurora Global Database per consentire le connessioni in uscita a S3. Per istruzioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

Aggiornamento di un database globale Amazon Aurora

L'aggiornamento di un database globale Aurora segue le stesse procedure dell'aggiornamento dei cluster di database Aurora. Tuttavia, di seguito sono riportate alcune importanti differenze di cui prendere nota prima di iniziare il processo.

Ti consigliamo di eseguire l'aggiornamento dei cluster di database primario e secondario alla stessa versione. È possibile eseguire un failover gestito del database tra regioni su un database globale

Aurora solo se i cluster di database primario e secondario hanno la stessa versione principale e secondaria e lo stesso livello di patch del motore. Tuttavia, i livelli di patch possono essere diversi, a seconda della versione secondaria del motore. Per ulteriori informazioni, consulta Compatibilità del livello di patch per switchover e failover gestiti tra regioni.

Aggiornamenti di una versione principale

Quando esegui un aggiornamento della versione principale di un database globale Amazon Aurora, aggiorni il cluster di database globale invece dei singoli cluster in esso contenuti.

Per informazioni su come aggiornare un database globale Aurora PostgreSQL a una versione principale superiore, consulta Principali aggiornamenti per database globali.

> **Note**
>
> Con un database globale Aurora basato su Aurora PostgreSQL, non è possibile eseguire un aggiornamento della versione principale del motore Aurora DB se la caratteristica Recovery point objective (RPO) (Obiettivo del punto di ripristino (RPO)) è attivata. Per ulteriori informazioni sulla caratteristica RPO, consulta Gestione RPOs di database globali basati su Aurora PostgreSQL.

Per informazioni su come aggiornare un database globale Aurora MySQL a una versione principale superiore, consulta Principali aggiornamenti in loco per database globali.

> **Note**
>
> Con un database globale Aurora basato su Aurora MySQL, è possibile eseguire un aggiornamento sul posto da Aurora MySQL versione 2 alla versione 3 solo se si imposta il parametro su default e si riavvia il database globale. `lower_case_table_names`
> Per eseguire un aggiornamento della versione principale ad Aurora MySQL versione 3 usando `lower_case_table_names`, puoi utilizzare la seguente procedura:
>
> 1. Rimuovi tutte le regioni secondarie dal cluster globale. Seguire la procedura riportata in Rimozione di un cluster da un database globale Amazon Aurora.
> 2. Esegui l'aggiornamento della versione del motore della regione principale ad Aurora MySQL versione 3. Seguire la procedura riportata in Come eseguire un aggiornamento in loco.

3. Aggiungi le regioni secondarie al cluster globale. Seguire la procedura riportata in Aggiungere un file Regione AWS a un database globale di Amazon Aurora.

Puoi anche utilizzare il metodo di ripristino snapshot. Per ulteriori informazioni, consulta Ripristino da uno snapshot cluster database.

Aggiornamenti della versione secondaria

Per un aggiornamento secondario su un database globale Aurora, è necessario aggiornare tutti i cluster secondari prima di aggiornare il cluster primario.

Per informazioni su come aggiornare un database globale Aurora PostgreSQL a una versione secondaria superiore, consulta Come eseguire aggiornamenti della versione secondaria e applicare patch. Per informazioni su come aggiornare un database globale Aurora MySQL a una versione secondaria superiore, consulta Aggiornamento di Aurora MySQL modificando la versione del motore.

Prima di eseguire l'aggiornamento, esamina le seguenti considerazioni:

- L'aggiornamento della versione secondaria di un cluster secondario non influisce in alcun modo sulla disponibilità o sull'utilizzo del cluster primario.
- Un cluster secondario deve disporre almeno di un'istanza database per eseguire un aggiornamento a una versione secondaria.
- Se si aggiorna un database globale Aurora MySQL alla versione 2.11.*, è necessario aggiornare i cluster DB primari e secondari esattamente alla stessa versione, incluso il livello di patch.
- Se si aggiorna un database globale Aurora PostgreSQL, è necessario aggiornare i cluster DB primari e secondari esattamente alla stessa versione e allo stesso livello di patch. Per aggiornare il livello di patch, applica tutte le azioni di manutenzione in sospeso sul cluster secondario.
- Per supportare switchover o failover gestiti tra regioni, potrebbe essere necessario aggiornare i cluster DB primari e secondari alla stessa identica versione, incluso il livello di patch. Questo requisito si applica ad Aurora MySQL e ad alcune versioni di Aurora PostgreSQL. Per un elenco di versioni che consentono lo switchover e il failover tra cluster che eseguono livelli di patch diversi, consulta. Compatibilità del livello di patch per switchover e failover gestiti tra regioni

Compatibilità del livello di patch per switchover e failover gestiti tra regioni

Se Aurora Global Database esegue una delle seguenti versioni secondarie del motore, è possibile eseguire switchover o failover gestiti tra regioni anche se i livelli di patch dei cluster DB primari e secondari non corrispondono. Per le versioni secondarie del motore inferiori a quelle in questo elenco, i cluster DB primari e secondari devono eseguire gli stessi livelli di patch principale, secondario e per eseguire switchover o failover gestiti tra aree geografiche. Assicurati di leggere le informazioni sulla versione e le note nella tabella seguente quando pianifichi gli aggiornamenti per il cluster primario, i cluster secondari o entrambi.

> **Note**
>
> Per i failover manuali tra regioni, è possibile eseguire il processo di failover purché il cluster database secondario di destinazione esegua la stessa versione principale e secondaria del motore del cluster database primario. In questo caso, i livelli di patch non devono necessariamente corrispondere.
> Se le versioni del motore richiedono livelli di patch identici, è possibile eseguire il failover manualmente seguendo la procedura riportata di seguito. Esecuzione di failover gestiti per database globali Aurora

Motore del database	Versioni secondarie del motore	Note
Aurora MySQL	Nessuna versione secondaria	Nessuna delle versioni secondarie di Aurora MySQL consente switchover o failover gestiti tra regioni con livelli di patch diversi tra i cluster DB primari e secondari.
Aurora PostgreSQL	• Versione 15 o versioni principali successive • Versione 14.5 o versioni secondarie successive • Versione 13.8 o versioni secondarie successive	Con le versioni del motore elencate nella colonna precedente, è possibile eseguire switchover o failover gestiti tra diverse regioni da un cluster DB primario con un livello di patch a un cluster DB secondario con un livello di patch diverso.

Motore del database	Versioni secondarie del motore	Note
	• Versione 12.12 o versioni secondarie successive • Versione 11.17 o versioni secondarie successive	Con versioni minori inferiori a queste, è possibile eseguire switchover o failover gestiti tra aree geografiche solo se i livelli di patch dei cluster DB primari e secondari corrispondono. **Warning** Quando aggiorni un cluster nel tuo database globale a una delle seguenti versioni di patch, non sarai in grado di eseguire switchover o failover tra aree geografiche finché tutti i cluster del tuo database globale non eseguiranno una di queste versioni di patch o una più recente. • Versioni di patch 16.1.6, 16.2.4, 16.3.2 e 16.4.2 • Versioni di patch 15.3.8, 15.4.9, 15.5.6, 15.6.4, 15.7.2 e 15.8.2 • Versioni di patch 14.8.8, 14.9.9, 14.10.6, 14.11.4, 14.12.2 e 14.13.2

Proxy Amazon RDS per Aurora

Con Amazon RDS Proxy, puoi consentire alle tue applicazioni di eseguire il pool e condividere connessioni di database per migliorare la loro capacità di dimensionamento. RDS Proxy rende le applicazioni più resilienti agli errori del database connettendosi automaticamente a un'istanza database di standby, mantenendo al contempo le connessioni delle applicazioni. Utilizzando RDS Proxy, puoi anche applicare l'autenticazione AWS Identity and Access Management (IAM) per i database e archiviare in modo sicuro le credenziali in. AWS Secrets Manager

Con Server proxy per RDS, puoi gestire picchi imprevedibili nel traffico del database In caso contrario, questi picchi potrebbero causare problemi a causa di un numero eccessivo di connessioni o della creazione di nuove connessioni a una velocità elevata. Server proxy per RDS stabilisce un pool di connessioni al database e riutilizza le connessioni di questo pool. Questo approccio evita il sovraccarico della memoria e della CPU per aprire ogni volta una nuova connessione al database. Per proteggere un database dall'eccesso di sottoscrizioni, è possibile controllare il numero di connessioni al database che vengono create.

Il proxy RDS mette in coda o limita le connessioni alle applicazioni che non possono essere servite immediatamente dal pool di connessioni. Sebbene le latenze possano aumentare, l'applicazione può continuare a dimensionare senza errori improvvisi o senza sovraccaricare il database. Se le richieste di connessione superano i limiti specificati, RDS Proxy rifiuta le connessioni dell'applicazione (genera il carico). Allo stesso tempo, mantiene prestazioni prevedibili per il carico che RDS è in grado di gestire con la capacità disponibile.

Puoi ridurre il sovraccarico per elaborare le credenziali e stabilire una connessione sicura per ogni nuova connessione. RDS Proxy può gestire alcune di queste operazioni per conto del database.

RDS Proxy è compatibile con le versioni di motore supportate. È possibile abilitare RDS Proxy per la maggior parte delle applicazioni senza modifiche al codice. Per un elenco delle versioni di motore supportate, consulta Regioni supportate e motori Aurora DB per Amazon RDS Proxy.

Argomenti

- Disponibilità di regioni e versioni
- Quote e limiti per RDS Proxy
- Pianificazione sull'utilizzo di RDS Proxy
- RDSConcetti e terminologia relativi ai proxy
- Guida introduttiva a RDS Proxy

- Gestione di un RDS proxy
- Utilizzo degli endpoint Amazon RDS Proxy
- Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch
- Utilizzo degli eventi RDS Proxy
- RDSEsempi della riga di comando del proxy
- Risoluzione dei problemi per RDS Proxy
- Utilizzo di RDS Proxy con AWS CloudFormation
- Utilizzo del Server proxy per RDS con i database globali Aurora

Disponibilità di regioni e versioni

Per informazioni sul supporto della versione del motore di database e sulla disponibilità di RDS Proxy in un determinato Regione AWS periodo, vedere. Regioni supportate e motori Aurora DB per Amazon RDS Proxy

Quote e limiti per RDS Proxy

A RDS Proxy si applicano i seguenti limiti:

- Ogni Account AWS ID è limitato a 20 proxy. Se la tua applicazione richiede più proxy, richiedi un aumento tramite la pagina Service Quotas all'interno di. AWS Management Console Nella pagina Service Quotas, seleziona Amazon Relational Database Service (Amazon RDS) e individua i proxy per richiedere un aumento della quota. AWS puoi aumentare automaticamente la tua quota o è in attesa di revisione della tua richiesta entro. Supporto
- Ogni proxy può avere fino a 200 segreti Secrets Manager associati. Pertanto, ogni proxy può connettersi a un massimo di 200 account utente diversi in qualsiasi momento.
- Ogni proxy ha un endpoint predefinito. Puoi anche aggiungere fino a 20 endpoint proxy per ogni proxy. È possibile creare, visualizzare, modificare ed eliminare questi endpoint.
- In un cluster Aurora, tutte le connessioni che utilizzano l'endpoint proxy predefinito vengono gestite dall'istanza scrittore di Aurora. Per eseguire il bilanciamento del carico per carichi di lavoro a uso intensivo di lettura, puoi creare un endpoint di sola lettura per un proxy. Tale endpoint passa le connessioni all'endpoint di lettura del cluster. In questo modo, le connessioni proxy possono sfruttare i vantaggi della scalabilità di lettura di Aurora. Per ulteriori informazioni, consulta Panoramica degli endpoint proxy.

- È possibile utilizzare Server proxy per RDS con cluster Aurora Serverless v2 ma non con cluster Aurora Serverless v1.
- RDS Proxy deve essere nello stesso virtual private cloud (VPC) del database. Sebbene il database non sia accessibile pubblicamente, il proxy può esserlo. Ad esempio, se state prototipando il database su un host locale, non potete connettervi al proxy a meno che non impostiate i requisiti di rete necessari per consentire la connessione al proxy. Questo perché l'host locale si trova all'esterno del VPC del proxy.

  > **Note**
  >
  > Per i cluster database Aurora, puoi attivare l'accesso tra VPC. Per fare ciò, creare un endpoint aggiuntivo per un proxy e specificare un VPC, sottoreti e gruppi di sicurezza diversi con tale endpoint. Per ulteriori informazioni, consulta Accesso ai database Aurora RDS VPCs.

- Non è possibile utilizzare RDS Proxy con un VPC con tenancy impostato a `dedicated`.
- Se utilizzi RDS Proxy con un cluster Aurora con autenticazione IAM abilitata, verifica l'autenticazione dell'utente. Gli utenti che si connettono tramite un proxy devono eseguire l'autenticazione con le credenziali di accesso. Per ulteriori informazioni sul supporto di Secrets Manager e IAM in Server proxy per Amazon RDS, consulta Configurazione delle credenziali del database in AWS Secrets Manager RDS Proxy e Configurazione dell'autenticazione IAM per RDS Proxy.
- Non puoi utilizzare RDS Proxy con DNS personalizzati quando utilizzi la convalida del nome host SSL.
- Ogni proxy può essere associato a un singolo cluster di database di destinazione. Tuttavia, è possibile associare più proxy allo stesso cluster di database.
- Qualsiasi istruzione con una dimensione del testo maggiore di 16 KB fa sì che il proxy effettui il pinning della sessione nella connessione corrente.
- Per alcune regioni sono presenti restrizioni relative alla zona di disponibilità (AZ) da considerare durante la creazione del proxy. La Regione Stati Uniti orientali (Virginia settentrionale) non supporta Server proxy per RDS nella zona di disponibilità `use1-az3`. La Regione Stati Uniti occidentali (California settentrionale) non supporta Server proxy per RDS nella zona di disponibilità `usw1-az2`. Quando selezioni le sottoreti durante la creazione del proxy, assicurati di non scegliere sottoreti nelle zone di disponibilità sopra menzionate.
- Attualmente, RDS Proxy non supporta alcuna chiave di contesto di condizione globale.

Per ulteriori informazioni sulle chiavi di contesto delle condizioni globali, consulta Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente di IAM.

- Non è possibile utilizzare RDS Proxy con RDS Custom per SQL Server.
- Per riflettere qualsiasi modifica del gruppo di parametri del database apportata al proxy, è necessario riavviare l'istanza anche se si sceglie di applicare immediatamente le modifiche. Per i parametri a livello di cluster, è necessario un riavvio a livello di cluster.
- Il proxy crea automaticamente l'utente `rdsproxyadmin` DB quando si registra una destinazione proxy. Si tratta di un utente protetto essenziale per la funzionalità del proxy. È necessario evitare di manomettere l'`rdsproxyadmin`utente a qualsiasi titolo. L'eliminazione o la modifica `rdsproxyadmin` dell'utente o delle relative autorizzazioni può comportare la completa indisponibilità del proxy per l'applicazione.

Per le altre limitazioni di ciascun motore di database, consulta le sezioni riportate di seguito:

- Limitazioni aggiuntive per Aurora MySQL
- Limitazioni aggiuntive per Aurora PostgreSQL

Limitazioni aggiuntive per Aurora MySQL

Le seguenti limitazioni aggiuntive si applicano a Server proxy per RDS con database Aurora MySQL:

- Il supporto del proxy RDS per `caching_sha2_password` l'autenticazione richiede una connessione sicura (TLS).
- `caching_sha2_password`È noto che il supporto RDS Proxy per presenta problemi di compatibilità con alcune versioni del driver go-sql.
- Quando si utilizza il driver MySQL 8.4 C, l'API potrebbe formare pacchetti non validi se `mysql_stmt_bind_named_param` il conteggio dei parametri supera il conteggio dei segnaposto in un'istruzione preparata. Ciò si traduce in risposte errate. Per ulteriori informazioni, consulta la segnalazione di bug di MySQL.
- Attualmente, tutti i proxy sono in ascolto di MySQL sulla porta 3306. I proxy si connettono ancora al database utilizzando la porta specificata nelle impostazioni del database.
- Non è possibile utilizzare RDS Proxy con database MySQL autogestiti in istanze. EC2
- Non è possibile utilizzare RDS Proxy con un'istanza database di RDS per MySQL con il parametro `read_only` nel suo gruppo di parametri database impostato su 1.

- RDS Proxy non supporta la modalità compressa MySQL. Ad esempio, non supporta la compressione utilizzata dalle opzioni `--compress` o `-C` del comando `mysql`.
- Le connessioni al database che elaborano un comando `GET DIAGNOSTIC` potrebbero restituire informazioni imprecise quando Server proxy per RDS riutilizza la stessa connessione al database per eseguire un'altra query. Questo può accadere quando Server proxy per RDS crea multiplex delle connessioni al database.
- Alcune istruzioni e funzioni SQL, ad esempio, `SET LOCAL` possono modificare lo stato della connessione senza causare il pinning. Per il comportamento del pinning più aggiornato, consulta Evitare il blocco di un proxy RDS.
- L'utilizzo della `ROW_COUNT()` funzione in una query con più istruzioni non è supportato.
- RDS Proxy non supporta le applicazioni client che non sono in grado di gestire più messaggi di risposta in un unico record TLS.

> **Important**
>
> Per i proxy associati ai database MySQL, non impostare il parametro `sql_auto_is_null` di configurazione su `true` o un valore diverso da zero nella query di inizializzazione. Ciò potrebbe causare un comportamento non corretto dell'applicazione.

Limitazioni aggiuntive per Aurora PostgreSQL

Le seguenti limitazioni aggiuntive si applicano a Server proxy per RDS con database Aurora PostgreSQL:

- RDS Proxy non supporta i filtri di pinning della sessione per PostgreSQL.
- Attualmente, tutti i proxy sono in ascolto di PostgreSQL sulla porta 5432.
- Per PostgreSQL, RDS Proxy attualmente non supporta l'annullamento di una query da un client tramite l'emissione di una `CancelRequest`. Questo è il caso, ad esempio, di quando si annulla una query a esecuzione prolungata in una sessione psql interattiva utilizzando Ctrl+C.
- I risultati della funzione PostgreSQL lastval non sono sempre accurati. Per risolvere il problema, utilizzare l'istruzione INSERT con la clausola `RETURNING`.
- Server proxy per RDS attualmente non supporta la modalità di replica in streaming.
- Il `default` database deve esistere.

- Se si utilizza `ALTER ROLE` per modificare il ruolo dell'utente con`SET ROLE`, le connessioni successive dell'utente al proxy potrebbero non utilizzare questa impostazione del ruolo, se tali connessioni presentano un blocco. Per evitare che ciò si verifichi, quando si utilizza il proxy, è necessario utilizzarlo `SET ROLE` nella query di inizializzazione del proxy. Per ulteriori informazioni, vedere Interrogazione di inizializzazione in. Creazione di un proxy per

> **Important**
>
> Per i proxy esistenti con database PostgreSQL, se si modifica l'autenticazione del database in modo da utilizzare solo `SCRAM`, il proxy diventa non disponibile per un massimo di 60 secondi. Per evitare il problema, procedi in uno dei seguenti modi:
>
> - Assicurati che il database consenta entrambe le autenticazioni `SCRAM` e `MD5`.
> - Per utilizzare solo l'autenticazione `SCRAM`, crea un nuovo proxy, esegui la migrazione del traffico dell'applicazione sul nuovo proxy, quindi elimina il proxy precedentemente associato al database.

Pianificazione sull'utilizzo di RDS Proxy

Puoi determinare quali tra le istanze DB, i cluster e le applicazioni possono trarre maggior vantaggio dall'utilizzo di RDS Proxy. Per fare ciò, considera questi fattori:

- Qualsiasi cluster database che genera errori del tipo "Troppe connessioni" è un buon candidato per l'associazione con un proxy. Questo è spesso caratterizzato da un valore elevato della `ConnectionAttempts` CloudWatch metrica. Il proxy consente alle applicazioni di aprire molte connessioni client, mentre il proxy gestisce un numero minore di connessioni di lunga durata al cluster database.
- Per DB, i cluster che utilizzano classi di AWS istanze più piccole, come T2 o T3, l'uso di un proxy può aiutare a evitare condizioni. out-of-memory Può anche contribuire a ridurre il sovraccarico della CPU per stabilire connessioni. Queste condizioni possono verificarsi quando vi è un numero elevato di connessioni.
- Puoi monitorare determinati CloudWatch parametri di Amazon per determinare se un cluster di DB si avvicina a determinati tipi di limite. Questi limiti riguardano il numero di connessioni e la memoria associata alla gestione delle connessioni. Puoi anche monitorare determinati CloudWatch parametri per determinare se un cluster di DB gestisce molte connessioni di breve

durata. L'apertura e la chiusura di tali connessioni possono determinare un sovraccarico delle prestazioni sul database. Per informazioni sui parametri da monitorare, consulta Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch.

- AWS LambdaAnche le funzioni possono essere ben utilizzate con un proxy. Queste funzioni effettuano frequenti connessioni del database brevi che beneficiano del pool di connessioni offerto da RDS Proxy. Puoi usufruire di qualsiasi autenticazione IAM già disponibile per le funzioni Lambda, invece di gestire le credenziali del database nel codice Lambda dell'applicazione.
- Le applicazioni che in genere aprono e chiudono un numero elevato di connessioni al database e non dispongono di meccanismi di pooling delle connessioni incorporati sono ottimi candidati per l'utilizzo di un proxy.
- Le applicazioni che mantengono un numero elevato di connessioni aperte per lunghi periodi sono in genere buoni candidati per l'utilizzo con un proxy. Applicazioni in ambiti come software as a service (SaaS) o e-commerce spesso riducono al minimo la latenza per le richieste del database lasciando aperte le connessioni.
- Potrebbe non essere stata adottata l'autenticazione IAM e Secrets Manager a causa della complessità di configurazione di tale autenticazione per tutti i cluster database. In tal caso, puoi abbandonare i metodi di autenticazione esistenti e delegare l'autenticazione a un proxy. Il proxy può applicare le policy di autenticazione per le connessioni client per applicazioni particolari. Puoi usufruire di qualsiasi autenticazione IAM già disponibile per le funzioni Lambda, invece di gestire le credenziali del database nel codice Lambda dell'applicazione.
- Server proxy per RDS può contribuire a rendere le applicazioni più resilienti e trasparenti agli errori del database. Server proxy per RDS ignora le cache del sistema dei nomi di dominio (DNS) per ridurre i tempi di failover fino al 66% per i database Multi-AZ Aurora. Server proxy per RDS inoltre instrada automaticamente il traffico a una nuova istanza database, preservando al contempo le connessioni dell'applicazione. In tal modo i failover sono più trasparenti per le applicazioni.

RDSConcetti e terminologia relativi ai proxy

Puoi semplificare la gestione delle connessioni per le tue (cluster Amazon Aurora DB) utilizzando Proxy. RDS

RDSIl proxy gestisce il traffico di rete tra l'applicazione client e il database. Lo fa in modo attivo prima comprendendo il protocollo del database. Quindi regola il suo comportamento in base alle SQL operazioni dell'applicazione e ai set di risultati del database.

RDSIl proxy riduce la memoria e il CPU sovraccarico per la gestione delle connessioni sul database. Il database richiede meno memoria e CPU risorse quando le applicazioni aprono molte connessioni simultanee. Inoltre, non richiede alcuna logica nelle applicazioni al fine di chiudere e riaprire le connessioni che rimangono inattive per un lungo periodo di tempo. Allo stesso modo, richiede meno operazioni logiche dell'applicazione per ristabilire le connessioni in caso di problemi di database.

L'infrastruttura per RDS Proxy è altamente disponibile e distribuita su più zone di disponibilità ()AZs. Il calcolo, la memoria e l'archiviazione per RDS Proxy sono indipendenti dal cluster Aurora DB dell'. Questa separazione consente di ridurre il sovraccarico sui server di database, in modo da poter dedicare le risorse al servizio dei carichi di lavoro del database. Le risorse di calcolo del RDS proxy sono serverless e si ridimensionano automaticamente in base al carico di lavoro del database.

Argomenti

- Panoramica dei concetti relativi al proxy RDS
- Pooling di connessioni
- RDSSicurezza proxy
- Failover
- Transazioni

Panoramica dei concetti relativi al proxy RDS

RDSIl proxy gestisce l'infrastruttura per eseguire il pool di connessioni e le altre funzionalità descritte nelle sezioni seguenti. Puoi vedere i proxy rappresentati nella RDS console nella pagina Proxy.

Ogni proxy gestisce le connessioni a un cluster Aurora DB. Il proxy determina automaticamente l'istanza di writer corrente per i cluster con provisioning Aurora dell'.

Le connessioni che un proxy mantiene aperte e disponibili per le applicazioni di database da utilizzare costituiscono il pool di connessioni.

Per impostazione predefinita, RDS Proxy può riutilizzare una connessione dopo ogni transazione della sessione. Questo riutilizzo a livello di transazione viene definito multiplexing. Quando RDS Proxy rimuove temporaneamente una connessione dal pool di connessioni per riutilizzarla, tale operazione viene chiamata prestito della connessione. Quando è sicuro farlo, RDS Proxy restituisce la connessione al pool di connessioni.

In alcuni casi, RDS Proxy non può garantire che sia sicuro riutilizzare una connessione al database al di fuori della sessione corrente. In questi casi, mantiene la sessione sulla stessa connessione fino al termine della sessione. Questo comportamento di fallback viene definito pinning.

Un proxy ha un endpoint predefinito. Ti connetti a questo endpoint quando lavori con un' Amazon Aurora DB. invece di connetterti all'endpoint di lettura-scrittura che si connette direttamente al cluster. Gli endpoint per finalità speciali per un cluster Aurora rimangono disponibili per l'utilizzo. Per i cluster DB Aurora , puoi anche creare endpoint di lettura/scrittura e di sola lettura aggiuntivi. Per ulteriori informazioni, consulta Panoramica degli endpoint proxy.

Ad esempio, puoi comunque connetterti all'endpoint del cluster per le connessioni di lettura-scrittura senza pool di connessioni. Puoi comunque connetterti all'endpoint di lettura per le connessioni di sola lettura con bilanciamento del carico. Puoi comunque connetterti agli endpoint dell'istanza per la diagnosi e la risoluzione dei problemi di istanze database specifiche all'interno di un cluster. Se utilizzi altri AWS servizi, ad esempio AWS Lambda per connetterti ai RDS database, modifica le relative impostazioni di connessione per utilizzare l'endpoint proxy. Ad esempio, si specifica l'endpoint proxy per consentire alle funzioni Lambda di accedere al database RDS sfruttando al contempo la funzionalità Proxy.

Ogni proxy contiene un gruppo di destinazione. Questo gruppo target include il cluster Aurora DB a cui il proxy può connettersi. Per un cluster Aurora, per impostazione predefinita, il gruppo di destinazione è associato a tutte le istanze DB in tale cluster. In questo modo, il proxy può connettersi a qualsiasi istanza database Aurora venga promossa a istanza di scrittura nel cluster. Il cluster Aurora DB associato a un proxy viene chiamato target di quel proxy. Per comodità, quando si crea un proxy tramite la console, RDS Proxy crea anche il gruppo di destinazione corrispondente e registra automaticamente le destinazioni associate.

Una famiglia di motori è un insieme correlato di motori di database che utilizzano lo stesso protocollo di DB. Scegli la famiglia di motori per ogni proxy creato.

Pooling di connessioni

Ogni proxy esegue il pool di connessioni separatamente per l'istanza writer e reader del associato. Il pool di connessioni è un'ottimizzazione che riduce il sovraccarico associato all'apertura e alla chiusura delle connessioni e mantiene aperte contemporaneamente molte connessioni. Questo sovraccarico include la memoria necessaria per gestire ogni nuova connessione. Inoltre, la chiusura di ogni connessione e l'apertura di una nuova comportano un CPU sovraccarico. Gli esempi includono Transport Layer Security/Secure Sockets Layer (TLS/SSL (), handshake, autenticazione,

funzionalità di negoziazione e così via. Il pool di connessioni semplifica la logica dell'applicazione. Non devi scrivere codice dell'applicazione per ridurre al minimo il numero di connessioni aperte simultanee.

Ogni proxy esegue anche il multiplexing delle connessioni, noto anche come riutilizzo della connessione. Con il multiplexing, RDS Proxy esegue tutte le operazioni per una transazione utilizzando una connessione al database sottostante. RDSquindi può utilizzare una connessione diversa per la transazione successiva. Puoi aprire molte connessioni simultanee al proxy e il proxy mantiene un numero minore di connessioni aperte all'istanza database o al cluster. In questo modo si riduce ulteriormente il sovraccarico di memoria per le connessioni sul server di database. Questa tecnica riduce anche la possibilità di errori del tipo «troppe connessioni».

RDSSicurezza proxy

RDSIl proxy utilizza i meccanismi RDS di sicurezza esistenti comeTLS/SSLe AWS Identity and Access Management (IAM). Per informazioni generali sulle funzionalità di sicurezza, consulta Sicurezza in Amazon Aurora. Inoltre, assicurati di acquisire familiarità con il funzionamento di Aurora con l'autenticazione, l'autorizzazione e altre aree di sicurezza.

RDSIl proxy può fungere da ulteriore livello di sicurezza tra le applicazioni client e il database sottostante. Ad esempio, è possibile connettersi al proxy utilizzando TLS 1.3, anche se l'istanza DB sottostante supporta una versione precedente diTLS. È possibile connettersi al proxy utilizzando un IAM ruolo. anche se il proxy si connette al database utilizzando il metodo di autenticazione utente/password nativo. Utilizzando questa tecnica, puoi applicare requisiti di autenticazione avanzata per le applicazioni di database senza un costoso sforzo di migrazione per le istanze DB medesime.

Le credenziali del database utilizzate da RDS Proxy vengono memorizzate in AWS Secrets Manager. Ogni utente del database per il cluster Aurora DB a cui accede un proxy deve avere un segreto corrispondente in Secrets Manager. È inoltre possibile configurare IAM l'autenticazione per gli utenti di RDS Proxy. In questo modo, è possibile applicare IAM l'autenticazione per l'accesso al database anche se i database utilizzano l'autenticazione con password nativa. È consigliabile utilizzare queste funzionalità di protezione anziché incorporare le credenziali del database nel codice dell'applicazione.

Utilizzo diTLS/SSLcon Proxy RDS

È possibile connettersi a RDS Proxy utilizzando il SSL protocollo TLS /.

> **Note**
>
> RDSIl proxy utilizza i certificati di AWS Certificate Manager (ACM). Se utilizzi RDS Proxy, non è necessario scaricare RDS certificati Amazon o aggiornare applicazioni che utilizzano connessioni RDS Proxy.

TLSPer applicare tutte le connessioni tra il proxy e il database, puoi specificare un'impostazione Require Transport Layer Security quando crei o modifichi un proxy in. AWS Management Console

RDSIl proxy può anche garantire che la sessione utilizziTLS/SSLtra il client e l'endpoint RDS Proxy. Per fare in modo che RDS Proxy lo faccia, specifica il requisito sul lato client. SSLle variabili di sessione non sono impostate per SSL le connessioni a un database tramite RDS Proxy.

- Aurora My SQL, specifica il requisito sul lato client con il `--ssl-mode` parametro quando esegui il `mysql` comando.
- Per Aurora Postgre SQL, specifica `sslmode=require` come parte della `conninfo` stringa quando esegui il comando. `psql`

RDSIl proxy supporta le versioni TLS del protocollo 1.0, 1.1, 1.2 e 1.3. È possibile connettersi al proxy utilizzando una versione superiore TLS a quella utilizzata nel database sottostante.

Per impostazione predefinita, i programmi client stabiliscono una connessione crittografata con RDS Proxy, con un ulteriore controllo disponibile tramite l'`--ssl-mode`opzione. Dal lato client, RDS Proxy supporta tutte le SSL modalità.

Per il client, le SSL modalità sono le seguenti:

PREFERRED

SSLè la prima scelta, ma non è obbligatoria.

DISABLED

No SSL è consentito.

REQUIRED

Far rispettareSSL.

VERIFY_CA

Applica SSL e verifica l'autorità di certificazione (CA).

VERIFY_IDENTITY

Applica SSL e verifica la CA e il nome host della CA.

Quando si utilizza un client con `--ssl-mode` `VERIFY_CA` o `VERIFY_IDENTITY`, specificare l'opzione `--ssl-ca` puntando a una CA in formato `.pem`. Per utilizzare il `.pem` file, scarica tutte le CA root PEMs da Amazon Trust Services e inseriscile in un unico `.pem` file.

RDSIl proxy utilizza certificati wildcard, che si applicano sia a un dominio che ai suoi sottodomini. Se si utilizza il `mysql` client per connettersi con la SSL modalità`VERIFY_IDENTITY`, attualmente è necessario utilizzare il comando My SQL 8.0-compatible. `mysql`

Failover

Il failover è una funzionalità ad alta disponibilità che sostituisce un'istanza di database con un'altra quando l'istanza originale diventa non disponibile. Un failover potrebbe verificarsi a causa di un problema con un'istanza di database. Potrebbe anche essere parte delle normali procedure di manutenzione, ad esempio durante un aggiornamento del database. Il failover si applica ai cluster Aurora DB con una o più istanze reader oltre all'istanza writer.

La connessione tramite un proxy rende le applicazioni più resistenti ai failover del database. Quando l'istanza DB originale non è più disponibile, RDS Proxy si connette al database di standby senza interrompere le connessioni alle applicazioni inattive. Questo aiuta ad accelerare e semplificare il processo di failover. Ciò comporta meno interruzioni per l'applicazione rispetto a un tipico problema di riavvio o di database.

Senza RDS Proxy, un failover comporta una breve interruzione. Durante l'interruzione, non è possibile eseguire operazioni di scrittura sul database in caso di failover. Tutte le connessioni al database esistenti vengono interrotte e l'applicazione deve riaprirle. Il database diventa disponibile per le nuove connessioni e le operazioni di scrittura quando un'istanza database di sola lettura viene promossa al posto di quella non disponibile.

Durante i failover del DB, RDS Proxy continua ad accettare connessioni allo stesso indirizzo IP e indirizza automaticamente le connessioni verso la nuova istanza database principale. I client che si connettono tramite RDS Proxy non sono soggetti a quanto segue:

- Ritardi di propagazione del Domain Name System (DNS) in caso di failover.
- Memorizzazione nella cache locale. DNS
- Timeout di connessione.
- Incertezza su quale istanza database è il writer corrente.
- In attesa di una risposta di query da un precedente writer che è diventato non disponibile senza chiudere le connessioni.

Per le applicazioni che mantengono il proprio pool di connessioni, RDS utilizzare Proxy significa che la maggior parte delle connessioni rimane attiva durante i failover o altre interruzioni. Vengono annullate solo le connessioni che si trovano nel bel mezzo di una transazione o di una SQL dichiarazione. RDS Il proxy accetta immediatamente nuove connessioni. Quando il database writer non è disponibile, RDS Proxy mette in coda le richieste in arrivo.

Per le applicazioni che non mantengono i propri pool di connessioni, RDS Proxy offre velocità di connessione più elevate e connessioni più aperte. Si evita il costoso sovraccarico dovuto a frequenti riconnessioni dal database. Lo fa riutilizzando le connessioni al database gestite nel pool di connessioni RDS Proxy. Questo approccio è particolarmente importante per TLS le connessioni, in cui i costi di configurazione sono significativi.

Transazioni

Tutte le istruzioni all'interno di una singola transazione utilizzano sempre la stessa connessione al database sottostante. La connessione diventa disponibile per l'utilizzo da parte di una sessione diversa al termine della transazione. L'utilizzo della transazione come unità di granularità ha le seguenti conseguenze:

- Il riutilizzo della connessione può avvenire dopo ogni singola dichiarazione quando l'impostazione Aurora SQL `autocommit` My è attivata.
- Al contrario, quando l'impostazione `autocommit` è disattivata, la prima istruzione che viene emessa in una sessione inizia una nuova transazione. Ad esempio, supponiamo di immettere una sequenza di`SELECT`, `INSERTUPDATE`, e altre istruzioni del linguaggio di manipolazione dei dati (). DML In questo caso, il riutilizzo della connessione non avviene fino a quando non invii un'istruzione come `COMMIT` o `ROLLBACK` per terminare la transazione.
- L'immissione di un'istruzione Data Definition Language (DDL) causa la fine della transazione dopo il completamento dell'istruzione.

RDSIl proxy rileva quando una transazione termina tramite il protocollo di rete utilizzato dall'applicazione client del database. Il rilevamento delle transazioni non si basa su parole chiave come `COMMIT` o `ROLLBACK` che compaiono nel testo della SQL dichiarazione.

In alcuni casi, RDS Proxy potrebbe rilevare una richiesta al database che rende poco pratico lo spostamento della sessione su un'altra connessione. In questi casi, disattiva il multiplexing per quella connessione al resto della sessione. La stessa regola si applica se RDS Proxy non può essere certo che il multiplexing sia pratico per la sessione. Questa operazione è chiamata pinning. Per informazioni su come rilevare e ridurre al minimo il pinning, consulta Evitare il blocco di un proxy RDS.

Guida introduttiva a RDS Proxy

Utilizza le informazioni contenute nelle pagine seguenti per configurare, gestire Proxy Amazon RDS per Aurora e impostare le opzioni di sicurezza correlate. Le opzioni di sicurezza controllano chi può accedere a ciascun proxy e in che modo ogni proxy si connette alle istanze DB.

Se non conosci RDS Proxy, ti consigliamo di seguire le pagine nell'ordine in cui le presentiamo.

Argomenti

- Configurazione dei prerequisiti di rete per RDS Proxy
- Configurazione delle credenziali del database in AWS Secrets Manager RDS Proxy
- Configurazione dell'autenticazione IAM per RDS Proxy
- Creazione di un proxy per
- Visualizzazione di un proxy
- Connessione a un database tramite RDS Proxy

Configurazione dei prerequisiti di rete per RDS Proxy

L'utilizzo di RDS Proxy richiede la disponibilità di un cloud privato virtuale (VPC) comune tra l' e il proxy RDS. Questo VPC deve avere un minimo di due sottoreti che si trovano in zone di disponibilità diverse. Il tuo account può possedere queste sottoreti o condividerle con altri account. Per informazioni sulla condivisione di VPC, consulta Work with shared. VPCs

Le risorse delle applicazioni client come Amazon EC2, Lambda o Amazon ECS possono trovarsi nello stesso VPC del proxy. In alternativa, possono trovarsi in un VPC separato dal proxy. Se hai

effettuato correttamente la connessione a un cluster di database Aurora, sono già disponibili le risorse di rete necessarie.

Argomenti

- Recupero delle informazioni sulle sottoreti
- Pianificazione della capacità degli indirizzi IP

Recupero delle informazioni sulle sottoreti

Se hai appena iniziato a usare Aurora, puoi apprendere le nozioni di base sulla connessione a un database seguendo le procedure riportate in. Configurazione dell'ambiente per Amazon Aurora Puoi inoltre seguire il tutorial in Nozioni di base su Amazon Aurora.

Per creare un proxy, è necessario fornire le sottoreti e il VPC in cui opera il proxy. Il seguente esempio di Linux mostra AWS CLI i comandi che esaminano le sottoreti VPCs e le sottoreti di proprietà dell'utente. Account AWS In particolare, si passa la sottorete IDs come parametri quando si crea un proxy utilizzando la CLI.

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

Il seguente esempio di Linux mostra AWS CLI i comandi per determinare la sottorete IDs corrispondente a una specifica istanza DB.

Per un cluster Aurora, è innanzitutto possibile trovare l'ID per una delle istanze database associate. È possibile estrarre la sottorete IDs utilizzata da quell'istanza DB. A tale scopo, esamina i campi nidificati all'interno degli attributi `DBSubnetGroup` e `Subnets` nell'output di descrizione dell'istanza database. È possibile specificare alcune o tutte queste sottoreti IDs quando si configura un proxy per quel server di database.

```
$ # Find the ID of any DB instance in the cluster.
$  aws rds describe-db-clusters --db-cluster-identifier my_cluster_id --query '*[].
[DBClusterMembers]|[0]|[0][*].DBInstanceIdentifier' --output text
```

```
my_instance_id
instance_id_2
instance_id_3
```

Dopo aver trovato l'identificatore dell'istanza database, esamina il VPC associato per individuare le sottoreti. Il seguente esempio Linux mostra come.

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]|[0]|[0]|[Subnets]|[0]|[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]|[0]|[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
subnet_id_4
subnet_id_5
subnet_id_6
```

Pianificazione della capacità degli indirizzi IP

Server proxy per RDS regola automaticamente la sua capacità secondo le necessità in base alle dimensioni e al numero di istanze database registrate. Alcune operazioni potrebbero richiedere anche una capacità proxy aggiuntiva, ad esempio l'aumento delle dimensioni di un database registrato o operazioni interne di manutenzione del proxy RDS. Durante queste operazioni, il proxy può aver bisogno di più indirizzi IP per fornire la capacità aggiuntiva. Questi indirizzi aggiuntivi consentono al proxy di dimensionare senza interessare il carico di lavoro. La mancanza di indirizzi IP liberi nelle sottoreti impedisce al proxy di aumentare le dimensioni, causando eventuali latenze elevate

nell'esecuzione di query o errori di connessione del client. RDS ti avvisa tramite l'evento `RDS-EVENT-0243` quando non ci sono abbastanza indirizzi IP liberi nelle tue sottoreti. Per informazioni su questo evento, consulta Utilizzo degli eventi RDS Proxy.

Riservate i seguenti numeri minimi di indirizzi IP gratuiti nelle sottoreti per il proxy, in base alle dimensioni delle classi delle istanze DB.

DB instance class (Classe istanza database)	Numero minimo di indirizzi IP liberi
db.*.xlarge o più piccola	10
db.*.2xlarge	15
db.*.4xlarge	25
db.*.8xlarge	45
db.*.12xlarge	60
db.*.16xlarge	75
db.*.24xlarge	110

Questi numeri consigliati di indirizzi IP sono stime per un proxy con solo l'endpoint predefinito. Un proxy con endpoint aggiuntivi o repliche di lettura potrebbe richiedere più indirizzi IP liberi. Per ogni endpoint aggiuntivo, ti consigliamo di riservare altri tre indirizzi IP. Per ogni replica di lettura, ti consigliamo di riservare gli indirizzi IP aggiuntivi specificati nella tabella in base alle dimensioni della replica di lettura.

> **Note**
>
> Il proxy RDS non utilizza più di 215 indirizzi IP in un VPC.

Supponi, ad esempio, di voler stimare gli indirizzi IP richiesti per un proxy associato a un cluster database Aurora.

Per questo caso:

- Il cluster database Aurora ha 1 istanza di scrittura di dimensioni db.r5.8xlarge e 1 istanza di lettura di dimensioni db.r5.2xlarge.
- Il proxy collegato a questo cluster database ha l'endpoint predefinito e 1 endpoint personalizzato con il ruolo di sola lettura.

In questo caso, il proxy richiede circa 63 indirizzi IP liberi (45 per l'istanza di scrittura, 15 per l'istanza di lettura e 3 per l'endpoint personalizzato aggiuntivo).

Configurazione delle credenziali del database in AWS Secrets Manager RDS Proxy

Il proxy RDS in Amazon RDS viene utilizzato AWS Secrets Manager per archiviare e gestire le credenziali del database in modo sicuro. Invece di incorporare le credenziali nell'applicazione, si associa un proxy a un segreto di Secrets Manager che contiene i dettagli di autenticazione necessari. Si crea un segreto Secrets Manager separato per ogni account utente del database a cui il proxy si connette sul cluster Aurora .

Argomenti

- Creazione di segreti da utilizzare con RDS Proxy

Creazione di segreti da utilizzare con RDS Proxy

Prima di creare un proxy, è necessario creare almeno un segreto che memorizzi le credenziali del database.

Console

Per creare un segreto

1. Apri la console Secrets Manager all'indirizzo https://console.aws.amazon.com/secretsmanager/.
2. Scegli Archivia un nuovo segreto.
3. Scegli Credenziali per il database Amazon RDS.
4. Inserisci un nome utente e una password. Le credenziali inserite devono corrispondere alle credenziali di un utente del database esistente nel database RDS associato. RDS Proxy utilizza queste credenziali per autenticare e stabilire connessioni al database per conto delle applicazioni.

In caso di mancata corrispondenza, puoi aggiornare il segreto in modo che corrisponda alla password del database. Finché non aggiorni il segreto, i tentativi di connessione tramite il proxy utilizzando quel segreto hanno esito negativo, ma le connessioni che utilizzano altri segreti validi continuano a funzionare.

> **Note**
>
> Per RDS per SQL Server, RDS Proxy richiede un segreto con distinzione tra maiuscole e minuscole in Secrets Manager, indipendentemente dalle impostazioni di confronto delle istanze DB. Se l'applicazione consente nomi utente con lettere maiuscole diverse, ad esempio «Admin» e «admin», è necessario creare segreti separati per ciascuno. RDS Proxy non supporta l'autenticazione del nome utente senza distinzione tra maiuscole e minuscole tra client e proxy.
> Per ulteriori informazioni sul confronto in SQL Server, consulta la documentazione di Microsoft SQL Server.

5. Per Database, seleziona il database Amazon RDS a cui accederà il segreto.
6. Inserisci altre impostazioni per il segreto, quindi scegli Store. Per istruzioni complete, consulta Creazione di un AWS Secrets Manager segreto nella Guida AWS Secrets Manager per l'utente.

AWS CLI

Quando crei un proxy tramite AWS CLI, specifichi gli Amazon Resource Names (ARNs) dei segreti corrispondenti. per tutti gli account utente del database a cui il proxy può accedere. In AWS Management Console, scegli i segreti in base ai loro nomi descrittivi.

- Per creare un segreto di Secrets Manager da utilizzare con RDS Proxy, usa il comando create-secret:

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- Puoi anche creare una chiave personalizzata per crittografare il tuo segreto di Secrets Manager. Il comando seguente crea una chiave di esempio.

```
aws kms create-key --description "test-key" --policy '{
  "Id":"kms-policy",
  "Version":"2012-10-17",
  "Statement":
    [
      {
        "Sid":"Enable IAM User Permissions",
        "Effect":"Allow",
        "Principal":{"AWS":"arn:aws:iam::account_id:root"},
        "Action":"kms:*","Resource":"*"
      },
      {
        "Sid":"Allow access for Key Administrators",
        "Effect":"Allow",
        "Principal":
          {
            "AWS":
              ["$USER_ARN","arn:aws:iam:account_id::role/Admin"]
          },
        "Action":
          [
            "kms:Create*",
            "kms:Describe*",
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms:Delete*",
            "kms:TagResource",
            "kms:UntagResource",
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion"
          ],
        "Resource":"*"
      },
      {
        "Sid":"Allow use of the key",
        "Effect":"Allow",
        "Principal":{"AWS":"$ROLE_ARN"},
        "Action":["kms:Decrypt","kms:DescribeKey"],
```

```
        "Resource":"*"
      }
    ]
}'
```

Ad esempio, i seguenti comandi creano segreti di Secrets Manager per due utenti del database:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

Per creare questi segreti crittografati con la tua AWS KMS chiave personalizzata, usa i seguenti comandi:

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

Per vedere i segreti di proprietà del tuo AWS account, usa il comando list-secrets:

```
aws secretsmanager list-secrets
```

Quando crei un proxy utilizzando la CLI, passi gli Amazon Resource Names (ARNs) di uno o più segreti al `--auth` parametro. L'esempio seguente mostra come preparare un rapporto con solo il nome e l'ARN di ogni segreto posseduto dal tuo AWS account. Questo esempio utilizza il `--output table` parametro disponibile nella AWS CLI versione 2. Se stai usando la AWS CLI versione 1, usa `--output text` invece.

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

Per confermare che il segreto contenga le credenziali corrette nel formato corretto, utilizzate il get-secret-valuecomando. Sostituisci *your_secret_name* con il nome breve o l'ARN del segreto.

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

L'output contiene una riga con un valore codificato in JSON simile al seguente:

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

Configurazione dell'autenticazione IAM per RDS Proxy

Per configurare l'autenticazione AWS Identity and Access Management (IAM) per RDS Proxy in Amazon RDS, crea e configura una policy IAM che conceda le autorizzazioni necessarie. Il proxy RDS consente di AWS Secrets Manager gestire le credenziali del database in modo sicuro, il che consente alle applicazioni di autenticarsi tramite il proxy senza gestire direttamente le credenziali.

Questo argomento fornisce i passaggi per configurare l'autenticazione IAM per RDS Proxy, inclusa la creazione della policy IAM richiesta e il suo collegamento a un ruolo IAM.

> **Tip**
>
> Questa procedura è necessaria solo se desideri creare il tuo ruolo IAM. Altrimenti, RDS può creare automaticamente il ruolo richiesto quando configuri il proxy, quindi puoi saltare questi passaggi.

Prerequisiti

Prima di configurare l'autenticazione IAM per RDS Proxy, assicurati di disporre di quanto segue:

- AWS Secrets Manager— Almeno un segreto archiviato che contiene le credenziali del database. Per istruzioni su come creare segreti, vederethe section called “Impostazione delle credenziali del database”.
- Autorizzazioni IAM: ruolo o utente IAM con autorizzazioni per creare e gestire policy, ruoli e segreti IAM. AWS Secrets Manager

Creazione di una policy IAM per l'accesso a Secrets Manager

Per consentire a RDS Proxy di recuperare le credenziali del database da Secrets Manager, crea un ruolo IAM con una policy che conceda le autorizzazioni necessarie.

Console

Per creare un ruolo per accedere ai tuoi segreti da utilizzare con il tuo proxy

1. Accedi AWS Management Console e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Crea una politica di autorizzazioni per il ruolo. Per i passaggi generali, consulta Creare politiche IAM (console).

   Incolla questa policy nell'editor JSON e apporta le seguenti modifiche:

   - Sostituire l'ID account.
   - Sostituiscilo `us-east-2` con la regione in cui risiederà il proxy.
   - Sostituisci i nomi segreti con quelli che hai creato. Per ulteriori informazioni, consulta Specificare le chiavi KMS nelle dichiarazioni politiche IAM.
   - Sostituisci l'ID della chiave KMS con quello che hai usato per crittografare i segreti di Secrets Manager, la chiave predefinita o la tua chiave.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
```

```
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. Crea il ruolo e allega la politica delle autorizzazioni. Per i passaggi generali, consulta Creare un ruolo per delegare le autorizzazioni a un servizio. AWS

   Per il tipo di entità affidabile, scegli AWS servizio. In Caso d'uso, seleziona RDS e scegli RDS - Aggiungi ruolo al database per il caso d'uso.

4. Per le politiche di autorizzazione, scegli la politica che hai creato.
5. Per Seleziona entità attendibili, inserisci la seguente politica di fiducia per il ruolo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Per creare il ruolo utilizzando il AWS CLI, invia la seguente richiesta:

```
aws iam create-role \
  --role-name my_role_name \
  --assume-role-policy-document '{"Version":"2012-10-17","Statement":
[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Quindi, allega la policy al ruolo:

```
aws iam put-role-policy \
  --role-name my_role_name \
  --policy-name secret_reader_policy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}'
```

Con il ruolo e le autorizzazioni IAM configurati, ora puoi creare un proxy e associarlo a questo ruolo. Ciò consente al proxy di recuperare le credenziali del database in modo sicuro AWS Secrets Manager e di abilitare l'autenticazione IAM per le tue applicazioni. Per istruzioni, consultare the section called “Creazione di un proxy”.

Creazione di un proxy per

Puoi utilizzare Amazon RDS Proxy per migliorare la scalabilità, la disponibilità e la sicurezza delle tue applicazioni di database raggruppando le connessioni e gestendo i failover del database in modo più efficiente. Questo argomento illustra come creare un proxy. Prima di iniziare, assicurati che il database soddisfi i prerequisiti necessari, tra cui le autorizzazioni IAM e la configurazione VPC.

Puoi associare un proxy a un cluster database Aurora MySQL o Aurora PostgreSQL.

Console

Per creare un proxy

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Scegli Create proxy (Crea proxy).
4. Configura le seguenti impostazioni per il tuo proxy.

Impostazione	Descrizione
Famiglia di motori	Il protocollo di rete del database che il proxy riconosce quando interpreta il traffico di rete da e verso il database. **Note** Per utilizzare Aurora PostgreSQL, assicurati di conservare il database nella tua istanza. `postgres` Consultare Risoluzione dei problemi relativi al `postgres` database eliminato.
Identificatore proxy	Un nome univoco all'interno dell'ID AWS dell'account e AWS della regione corrente.
Timeout della connessione al client inattiva	Il proxy chiude una connessione client se rimane inattiva per un determinato periodo. Per impostazione predefinita, si tratta di 1.800 secondi (30 minuti). Una connessione è inattiva quando l'applicazione non invia una nuova richiesta entro il tempo specificato dopo aver completato la richiesta precedente. Il proxy mantiene aperta la connessione al database sottostante e la restituisce al pool di connessioni, rendendola disponibile per nuove connessioni client. Per rimuovere in modo proattivo le connessioni obsolete, riduci il timeout della connessione client inattiva. Per ridurre al minimo i

Impostazione	Descrizione
	costi di connessione durante i picchi di carico di lavoro, aumenta il timeout.
Database	L' del cluster Aurora DB a cui accedere tramite questo proxy. L'elenco include solo istanze DB e cluster con motori di database compatibili, versioni del motore e altre impostazi oni. Se l'elenco è vuoto, crea una nuova istanza database o un cluster compatibile con RDS Proxy. A tale scopo, segui la procedura in Creazione di un cluster database Amazon Aurora. Quindi, prova a creare nuovamente il proxy.
Numero massimo di connessio ni del pool di connessioni	Un valore compreso tra 1 e 100 per definire la percentuale del `max_connections` limite che il proxy RDS può utilizzare. Se intendi utilizzare un solo proxy con questa istanza o cluster DB, imposta questo valore su 100. Per ulteriori informazioni su come RDS Proxy utilizza questa impostazione, vederethe section called “MaxConnectionsPercent”.

Impostazione	Descrizione
Filtri di blocco delle sessioni	Impedisce a RDS Proxy di bloccare determinati stati di sessione rilevati, ignorando così le misure di sicurezza predefinite per le connessioni multiplexate. Attualmente, PostgreSQL non supporta questa impostazione e l'unica opzione disponibile è. `EXCLUDE_VARIABLE_SETS` La sua attivazione potrebbe far sì che le variabili di sessione di una connessione influiscano sulle altre, con conseguenti errori o problemi di correttezza se le query si basano su variabili di sessione impostate al di fuori della transazione corrente. Utilizzate questa opzione solo dopo aver verificato che le applicazioni possano condividere in modo sicuro le connessioni al database. I seguenti modelli sono considerati sicuri: • `SET`istruzioni in cui non viene apportata alcuna modifica al valore della variabile di sessione effettiva. In altre parole, non viene apportata alcuna modifica alla variabile di sessione. • Modifichi il valore della variabile di sessione ed esegui un'istruzione nella stessa transazione. Per ulteriori informazioni, consulta Evitare il blocco di un proxy RDS.
Timeout di prestito della connessione	Se prevedi che il proxy utilizzi tutte le connessioni al database disponibili, imposta il tempo di attesa prima che restituisca un errore di timeout. È possibile specificare fino a cinque minuti. Questa impostazione si applica solo quando il proxy ha raggiunto il numero massimo di connessioni e tutte sono in uso.

Impostazione	Descrizione
Interrogazione di inizializ zazione	(Facoltativo) Aggiungi una query di inizializzazione o modifica quella corrente. Puoi specificare una o più istruzioni SQL per l'esecuzione del proxy all'apertura di ogni nuova connessione al database. L'impostazione viene in genere utilizzata con SET le istruzioni per assicurarsi che ogni connessione abbia impostazi oni identiche. Assicurati che la query che aggiungi sia valida. Per includere più variabili in una singola SET istruzione, utilizzate i separatori di virgole. Per esempio: `SET variable1 =value1, variable2 =value2` Per più istruzioni, utilizzare il punto e virgola come separatore. **Important** Poiché è possibile accedere alla query di inizializzazione come parte della configurazione del gruppo target, questa non è protetta da metodi di autenticazione o crittografia. Chiunque abbia accesso alla visualizzazione o alla gestione della configurazione del gruppo target proxy può visualizzare la query di inizializzazione. Non è necessario aggiungere dati sensibili, come password o chiavi di crittografia a lunga durata, a questa opzione.
AWS Identity and Access Management Ruolo (IAM)	Un ruolo IAM con autorizzazione ad accedere ai segreti di Secrets Manager, che rappresentano le credenziali per gli account utente del database che il proxy può utilizzar e. In alternativa, puoi creare un nuovo ruolo IAM da. AWS Management Console
I segreti di Secrets Manager	Scegli almeno un segreto di Secrets Manager che contenga le credenziali utente del database che consentano al proxy di accedere al cluster .

Impostazione	Descrizione
Tipo di autenticazione del client	Il tipo di autenticazione utilizzato dal proxy per le connessio ni dai client. La tua scelta si applica a tutti i segreti di Secrets Manager che associ a questo proxy. Se devi specificare un tipo di autenticazione client diverso per ogni segreto, crea il proxy utilizzando invece l'API AWS CLI o l'API.
Autenticazione IAM	Se richiedere o non consentire l'autenticazione IAM per le connessioni al proxy. La tua scelta si applica a tutti i segreti di Secrets Manager che associ a questo proxy. Se devi specificare un'autenticazione IAM diversa per ogni segreto, crea il tuo proxy utilizzando invece l'API AWS CLI o.
Richiedi Transport Layer Security	Applica TLS/SSL per tutte le connessioni client. Il proxy utilizza la stessa impostazione di crittografia per la connessione al database sottostante, indipendentemente dal fatto che la connessione client sia crittografata o meno.
Sottoreti	Questo campo è precompilato con tutte le sottoreti associate al tuo VPC. È possibile rimuovere tutte le sottoreti non necessarie per il proxy, ma è necessario lasciare almeno due sottoreti.

Impostazione	Descrizione
Gruppo di sicurezza VPC	Scegli un gruppo di sicurezza VPC esistente o creane uno nuovo dal. AWS Management Console Configura le regole in entrata per consentire alle applicazioni di accedere al proxy e le regole in uscita per consentire il traffico proveniente dalle destinazioni del database. **Note** Il gruppo di sicurezza deve consentire le connessio ni dal proxy al database. Serve sia per l'ingresso dalle applicazioni al proxy che per l'uscita dal proxy al database. Ad esempio, se utilizzi lo stesso gruppo di sicurezza sia per il database che per il proxy, assicurat i che le risorse all'interno di quel gruppo di sicurezza possano comunicare tra loro. Quando utilizzi un VPC condiviso, evita di utilizzar e il gruppo di sicurezza predefinito per il VPC o uno associato a un altro account. Seleziona invece un gruppo di sicurezza che appartiene al tuo account. Se non ne esiste nessuno, creane uno. Per ulteriori informazioni, consulta Work with shared VPCs. RDS implementa un proxy in più zone di disponibilità per garantire un'elevata disponibilità. Per abilitare la comunicazione Cross-AZ, l'elenco di controllo degli accessi alla rete (ACL) per la sottorete proxy deve consentire l'uscita dalla porta del motore e l'ingresso su tutte le porte. Per ulteriori informazioni sulla rete ACLs, consulta Controllare il traffico verso le sottoreti utilizzan do la rete. ACLs Se l'ACL di rete per il proxy e la destinazi one sono identici, devi aggiungere una regola di ingresso del protocollo TCP in cui Origine è impostata sul VPC CIDR. È inoltre necessario aggiungere una regola di uscita del protocoll

Impostazione	Descrizione
	o TCP specifica per la porta del motore in cui la destinazione è impostata sul VPC CIDR.
Attiva la registrazione avanzata	Abilita questa impostazione per risolvere problemi di compatibilità o prestazioni del proxy. Se abilitato, RDS Proxy registra informazioni dettagliate sulle prestazioni per aiutarti a eseguire il debug del comportamento SQL o delle prestazioni e della scalabilità della connessione proxy. Abilita questa impostazione solo per il debug e assicurati che siano state adottate misure di sicurezza adeguate per proteggere le informazioni sensibili nei log. Per ridurre al minimo il sovraccarico, RDS Proxy disattiva automaticamente questa impostazione 24 ore dopo l'attivazione. Utilizzala temporaneamente per risolvere problemi specifici.

5. Scegli Create proxy (Crea proxy).

AWS CLI

Per creare un proxy utilizzando il AWS CLI, chiamate il create-db-proxycomando con i seguenti parametri obbligatori:

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--auth`
- `--vpc-subnet-ids`

Il valore `--engine-family` prevede la distinzione tra lettere maiuscole e minuscole.

Example

Per LinuxmacOS, oUnix:

```
aws rds create-db-proxy \
```

```
    --db-proxy-name proxy_name \
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --auth ProxyAuthenticationConfig_JSON_string \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--tags comma_separated_list]
```

Per Windows:

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --auth ProxyAuthenticationConfig_JSON_string ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--tags comma_separated_list]
```

Di seguito è riportato un esempio di valore JSON per l'opzione `--auth`. Questo esempio applica un tipo di autenticazione client diverso a ciascun segreto.

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },

  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
```

```
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },

  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

> **Tip**
>
> Se non conosci già la sottorete IDs da utilizzare per il `--vpc-subnet-ids` parametro, consulta Configurazione dei prerequisiti di rete per RDS Proxy alcuni esempi su come trovarli.

> **Note**
>
> Il gruppo di protezione deve consentire l'accesso al database a cui si connette il proxy. Lo stesso gruppo di protezione viene utilizzato per l'ingresso dalle applicazioni al proxy e per l'uscita dal proxy al database. Si supponga, ad esempio, di utilizzare lo stesso gruppo di protezione per il database e il proxy. In questo caso, assicurarsi di specificare che le risorse di tale gruppo di protezione possono comunicare con altre risorse dello stesso gruppo di protezione.
>
> Quando si utilizza un VPC condiviso, non è possibile utilizzare il gruppo di sicurezza predefinito per il VPC o uno appartenente a un altro account. Scegli un gruppo di sicurezza appartenente all'account. Se non esiste, creane uno. Per ulteriori informazioni su questa limitazione, consulta Work with shared VPCs.

Per creare le associazioni corrette per il proxy, si usa anche il register-db-proxy-targetscomando. Specificare il tipo di gruppo di destinazione `default` RDS Proxy crea automaticamente un gruppo di destinazione con questo nome quando si crea ogni proxy.

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

API RDS

Per creare un proxy RDS, chiama l'operazione API Amazon RDS Create. DBProxy Passi un parametro con la struttura dei AuthConfigdati.

RDS Proxy crea automaticamente un gruppo di destinazione denominato `default` quando si crea ogni proxy. È possibile associare un cluster Aurora al gruppo di destinazione chiamando la funzione DBProxy Register Targets.

Visualizzazione di un proxy

Dopo aver creato uno o più proxy RDS, è possibile visualizzarli e gestirli nell' AWS Management Console API AWS CLI RDS. È possibile esaminarne i dettagli di configurazione, monitorare le prestazioni e determinare quali proxy modificare o eliminare in base alle esigenze.

Per consentire alle applicazioni di database di indirizzare il traffico attraverso un proxy, è necessario specificare l'endpoint proxy nella stringa di connessione.

Console

Per visualizzare un proxy nella console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Seleziona il nome del proxy per visualizzarne i dettagli.
4. Nella pagina dei dettagli, la sezione Target groups mostra come il proxy è collegato a un' Aurora DB specifica. È possibile accedere alla pagina del gruppo di destinazione predefinito per una visione più approfondita di questa associazione, comprese le impostazioni di configurazione definite durante la creazione del proxy. Queste impostazioni includono la percentuale massima di connessione, il timeout del prestito della connessione, la famiglia di motori e i filtri di blocco della sessione.

CLI

Per visualizzare il proxy utilizzando la CLI, utilizzare il describe-db-proxiescomando. Per impostazione predefinita, la richiesta restituisce tutti i proxy di proprietà del tuo account. AWS Per visualizzare i dettagli di un singolo proxy, specificarne il nome con il parametro `--db-proxy-name`.

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

Per visualizzare altre informazioni associate al proxy, utilizza i seguenti comandi.

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

Utilizza la seguente sequenza di comandi per visualizzare ulteriori dettagli sugli elementi associati al proxy:

1. Per ottenere un elenco di proxy, esegui. describe-db-proxies
2. Per mostrare i parametri di connessione, come la percentuale massima di connessioni che il proxy può utilizzare, describe-db-proxy-target esegui -groups. `--db-proxy-name` Utilizza il nome del proxy come valore del parametro.
3. Per visualizzare i dettagli del cluster Aurora DB associato al gruppo di destinazione restituito, esegui. describe-db-proxy-targets

API RDS

Per visualizzare i proxy utilizzando l'API RDS, utilizza l'operazione Descrivi. DBProxies Restituisce i valori del tipo di dati. DBProxy

Per visualizzare i dettagli delle impostazioni di connessione per il proxy, utilizzate gli identificatori proxy di questo valore restituito con l'DBProxyTargetGroupsoperazione Descrivi. Restituisce i valori del tipo di DBProxyTargetGroupdati.

Per visualizzare l'istanza RDS o il cluster Aurora DB associato al proxy, usa l'operazione DBProxyDescrivi Targets. Restituisce valori del tipo di dati DBProxyTarget.

Connessione a un database tramite RDS Proxy

Ti connetti a un cluster o cluster Aurora DB che utilizza Aurora Serverless v2 tramite un proxy in genere nello stesso modo in cui ci si connette direttamente al database. La differenza principale è che si specifica l'endpoint del proxy anziché l'endpoint del cluster. Per impostazione predefinita, tutte le connessioni proxy hanno capacità di lettura/scrittura e utilizzano l'istanza di scrittura. Se normalmente utilizzi l'endpoint di lettura per connessioni di sola lettura, puoi creare un endpoint aggiuntivo di sola lettura per il proxy e usare l'endpoint allo stesso modo. Per ulteriori informazioni, consulta Panoramica degli endpoint proxy.

Argomenti

- Connessione a un proxy utilizzando l'autenticazione nativa
- Connessione a un proxy mediante autenticazione IAM
- Considerazioni per la connessione a un proxy con PostgreSQL

Connessione a un proxy utilizzando l'autenticazione nativa

Utilizza i seguenti passaggi per connetterti a un proxy utilizzando l'autenticazione nativa:

1. Trova l'endpoint del proxy. In AWS Management Console, puoi trovare l'endpoint nella pagina dei dettagli del proxy corrispondente. Con AWS CLI, è possibile utilizzare il describe-db-proxiescomando. L'esempio seguente mostra come.

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
        },
        {
```

```
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. Specificate l'endpoint come parametro host nella stringa di connessione per l'applicazione client. Ad esempio, specificare l'endpoint proxy come valore per l'opzione `mysql -h` o l'opzione `psql -h`.
3. Fornisci lo stesso nome utente e la stessa password del database come fai normalmente.

Connessione a un proxy mediante autenticazione IAM

Quando si utilizza l'autenticazione IAM con RDS Proxy, impostare gli utenti del database per l'autenticazione con nomi utente e password regolari. L'autenticazione IAM si applica al recupero RDS Proxy delle credenziali del nome utente e della password da Secrets Manager. La connessione da RDS Proxy al database sottostante non passa tramite IAM.

Per connetterti al proxy RDS utilizzando l'autenticazione IAM, utilizza la stessa procedura di connessione generale utilizzata per l'autenticazione IAM con un cluster . Per informazioni generali sull'utilizzo di IAM, consulta Sicurezza in Amazon Aurora.

Le principali differenze nell'utilizzo di IAM per RDS Proxy includono le seguenti:

- Non si configura ogni singolo utente del database con un plugin di autorizzazione. Gli utenti del database hanno ancora nomi utente e password regolari all'interno del database. Si impostano i segreti Secrets Manager contenenti questi nomi utente e password e si autorizza RDS Proxy a recuperare le credenziali da Secrets Manager.

  L'autenticazione IAM si applica alla connessione tra il programma client e il proxy. Il proxy esegue quindi l'autenticazione nel database utilizzando il nome utente e le credenziali della password recuperate da Secrets Manager.
- Invece dell'istanza, del cluster o dell'endpoint dell'istanza di lettura, specifica l'endpoint del proxy. Per informazioni dettagliate sull'endpoint proxy, vedere Connessione al cluster di database tramite l'autenticazione IAM.

- Nel caso di autenticazione diretta di database IAM, scegli selettivamente gli utenti del database e configurali per essere identificati con uno speciale plug-in di autenticazione. È quindi possibile connettersi a tali utenti utilizzando l'autenticazione IAM.

  Nel caso d'uso del proxy, è necessario fornire al proxy segreti che contengono nome utente e password di alcuni utenti (autenticazione nativa). Quindi ci si connette al proxy utilizzando l'autenticazione IAM. Qui, si esegue questa operazione generando un token di autenticazione con l'endpoint proxy, non l'endpoint del database. Si utilizza anche un nome utente che corrisponde a uno dei nomi utente per i segreti forniti.
- Quando ci si connette a un proxy utilizzando l'autenticazione IAM è necessario utilizzare Transport Layer Security (TLS)/Secure Sockets Layer (SSL).

È possibile concedere a un utente specifico l'accesso al proxy modificando la policy IAM. Di seguito è riportato un esempio.

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

Considerazioni per la connessione a un proxy con PostgreSQL

Se crei un nuovo utente del database PostgreSQL per la connessione a RDS Proxy, assicurati di concedere il privilegio utente sul database. `CONNECT` Senza questo, l'utente non può stabilire una connessione. Per ulteriori informazioni, consulta the section called “Aggiungere un nuovo utente del database a un database Postgre quando si SQL utilizza Proxy RDS”.

Quando un client avvia una connessione a un database PostgreSQL, invia un messaggio di avvio. include coppie di stringhe di nome parametro e valore. Per i dettagli, vedere i `StartupMessage` in Formati dei messaggi PostgreSQL nella documentazione di PostgreSQL.

Quando ci si connette tramite un proxy RDS, il messaggio di avvio può includere i seguenti parametri attualmente riconosciuti:

- `user`
- `database`

Il messaggio di avvio può includere anche i seguenti parametri di runtime aggiuntivi:

- `application_name`

- `client_encoding`
- `DateStyle`
- `TimeZone`
- `extra_float_digits`
- `search_path`

Per ulteriori informazioni sulla messaggistica PostgreSQL, vedere Frontend/Backend Protocol nella documentazione di PostgreSQL.

Per PostgreSQL, se usi JDBC, ti consigliamo quanto segue per evitare il pinning:

- Impostare il parametro `assumeMinServerVersion` di connessione JDBC su almeno `9.0` per evitare il pinning. Ciò impedisce al driver JDBC di eseguire un ulteriore round trip durante l'avvio della connessione quando è in esecuzione. `SET extra_float_digits = 3`
- Impostare il parametro di connessione JDBC `ApplicationName` su *`any/your-application-name`* per evitare il pinning. In questo modo si impedisce al driver JDBC di eseguire un ulteriore round trip durante l'avvio della connessione quando viene eseguito `SET application_name = "PostgreSQL JDBC Driver"`. Si noti che il parametro JDBC è `ApplicationName` ma il parametro `StartupMessage` PostgreSQL è `application_name`.

Per ulteriori informazioni, consulta Evitare il blocco di un proxy RDS. Per ulteriori informazioni sulla connessione tramite JDBC, vedere Connessione al database nella documentazione di PostgreSQL.

Gestione di un RDS proxy

Questa sezione fornisce informazioni su come gestire il funzionamento e la configurazione del RDS proxy. Queste procedure consentono all'applicazione di utilizzare in modo più efficiente le connessioni al database e di ottenere il massimo riutilizzo della connessione. Quanto più si può trarre vantaggio dal riutilizzo della connessione, tanto maggiore CPU è il sovraccarico di memoria che è possibile risparmiare. Questo a sua volta riduce la latenza per l'applicazione e consente al database di dedicare più risorse all'elaborazione delle richieste dell'applicazione.

Argomenti

- Modifica di un RDS Proxy
- Aggiungere un nuovo utente del database quando si utilizza RDS Proxy

- Considerazioni sulla connessione del proxy RDS
- Evitare il blocco di un proxy RDS
- Eliminazione di un proxy RDS

Modifica di un RDS Proxy

Puoi modificare specifiche impostazioni associate a un proxy dopo aver creato il proxy. Esegui questa operazione modificando il proxy stesso, il suo gruppo di destinazione associato o entrambi. Ogni proxy ha un gruppo di destinazione associato.

AWS Management Console

> **Important**
>
> I valori nei campi Client authentication type (Tipo di autenticazione client) e IAM authentication (autenticazione IAM) si applicano a tutti i segreti di Secrets Manager associati al proxy. Per specificare valori diversi per ogni segreto, modifica il proxy utilizzando invece l'API AWS CLI o.

Per modificare le impostazioni di un proxy

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Nell'elenco dei proxy, scegli il proxy di cui desideri modificare le impostazioni o passa alla relativa pagina dei dettagli.
4. Per Actions (Operazioni), scegliere Modify (Modifica).
5. Inserisci o scegli le proprietà da modificare. È possibile modificare le seguenti:
   - L'identificatore Proxy: rinominare il proxy immettendo un nuovo identificatore.
   - Timeout della connessione client per inattività: immettere un periodo di tempo per il timeout della connessione client inattiva.
   - Ruolo IAM: modificare il ruolo IAM utilizzato per recuperare i segreti da Secrets Manager.
   - Segreti di Secrets Manager: aggiungere o rimuovere segreti di Secrets Manager. Questi segreti corrispondono a nomi utente e password del database.

- Tipo di autenticazione client: (solo PostgreSQL) cambia il tipo di autenticazione per le connessioni client al proxy.
- Autenticazione IAM: richiedi o disabilita l'autenticazione IAM per le connessioni al proxy.
- Richiedi Transport Layer Security: attivare o disattivare il requisito per Transport Layer Security (TLS).
- Gruppo di sicurezza VPC: aggiungere o rimuovere gruppi di sicurezza VPC da utilizzare per il proxy.
- Abilitazione della registrazione avanzata: abilitare o disabilitare la registrazione avanzata.

6. Scegliere Modify (Modifica).

Se non sono state trovate le impostazioni elencate che si desidera modificare, attenersi alla procedura seguente per aggiornare il gruppo di destinazione per il proxy. Il gruppo di destinazione associato a un proxy controlla le impostazioni relative alle connessioni del database fisico. Ogni proxy ha un gruppo di destinazione associato denominato `default`, che viene creato automaticamente insieme al proxy. Non puoi rinominare il gruppo target predefinito.

Puoi modificare il gruppo di destinazione solo dalla pagina dei dettagli del proxy, non dall'elenco nella pagina Proxies (Proxy) .

Per modificare le impostazioni di un gruppo di destinazione proxy

1. Dalla pagina Proxy, passa alla pagina dei dettagli di un proxy.
2. Per Target groups (Gruppi di destinazione), scegli il collegamento di `default`. Attualmente, tutti i proxy hanno un singolo gruppo di destinazione denominato `default`.
3. Nella pagina dei dettagli del gruppo di destinazione di default, scegli Modify (Modifica).
4. Scegli nuove impostazioni per le proprietà che è possibile modificare:

   - Database: puoi scegliere un diverso cluster Aurora.
   - Connessioni massime del pool di connessioni: puoi modificare la percentuale delle connessioni massime disponibili che il proxy può utilizzare.
   - Filtri di pinning della sessione: (opzionale) scegliere un filtro di pinning della sessione. Ciò elude le misure di sicurezza predefinite per il multiplexing delle connessioni al database tra connessioni client. Attualmente, l'impostazione non è supportata per PostgreSQL. L'unica scelta è. `EXCLUDE_VARIABLE_SETS`

L'attivazione di questa impostazione può far sì che le variabili di sessione di una connessione influiscano sulle altre connessioni. Ciò può causare errori o problemi di correttezza se le query dipendono dai valori delle variabili di sessione impostati al di fuori della transazione corrente. Valuta l'utilizzo di questa opzione dopo aver verificato che sia sicuro per le applicazioni condividere le connessioni al database tra connessioni client.

I seguenti modelli possono essere considerati sicuri:

- Istruzioni SET in cui non viene apportata alcuna modifica al valore della variabile di sessione effettiva, ovvero non viene apportata alcuna modifica alla variabile di sessione.
- Modifichi il valore della variabile di sessione ed esegui un'istruzione nella stessa transazione.

Per ulteriori informazioni, consulta Evitare il blocco di un proxy RDS.

- Timeout del prestito di connessione: puoi regolare l'intervallo di timeout del prestito di connessione. Questa impostazione si applica quando il numero massimo di connessioni è già in uso per il proxy. In questi casi, è possibile specificare per quanto tempo il proxy attende che una connessione diventi disponibile prima di restituire un errore di timeout.
- Query di inizializzazione. (Facoltativo) Aggiungi una query di inizializzazione o modifica quella corrente. Puoi specificare una o più istruzioni SQL per l'esecuzione del proxy all'apertura di ogni nuova connessione al database. L'impostazione viene in genere utilizzata con SET le istruzioni per assicurarsi che ogni connessione abbia impostazioni identiche. Assicurati che la query che aggiungi sia valida. Per includere più variabili in una singola SET istruzione, utilizzate i separatori di virgole. Per esempio:

```
SET variable1=value1, variable2=value2
```

Per più istruzioni, utilizzare il punto e virgola come separatore.

Alcune proprietà, ad esempio l'identificatore del gruppo di destinazione e il motore del database, sono fisse.

5. Scegli Modify target group (Modifica gruppo di destinazione).

AWS CLI

Per modificare un proxy utilizzando il AWS CLI, usa i comandi modify-db-proxymodify-db-proxy-target-group deregister-db-proxy-targets, e. register-db-proxy-targets

Con il comando `modify-db-proxy`, è possibile modificare proprietà come le seguenti:

- L'insieme di segreti Secrets Manager usati dal proxy.
- Indica se TLS è necessario.
- Il timeout del client inattivo.
- Indica se registrare ulteriori informazioni dalle istruzioni SQL per il debug.
- Il ruolo IAM utilizzato per recuperare i segreti Secrets Manager.
- I gruppi di sicurezza utilizzati dal proxy.

Nell'esempio seguente viene illustrato come rinominare un proxy esistente.

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

Con il comando `modify-db-proxy-target-group`, puoi modificare le impostazioni relative alla connessione o rinominare il gruppo di destinazione. Attualmente, tutti i proxy hanno un singolo gruppo di destinazione denominato `default`. Quando lavori con questo gruppo target, specifichi il nome del proxy e `default` il nome del gruppo target. Non puoi rinominare il gruppo target predefinito.

Nell'esempio seguente viene illustrato come controllare prima l'impostazione `MaxIdleConnectionsPercent` per un proxy e quindi modificarla utilizzando il gruppo di destinazione.

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
                "ConnectionBorrowTimeout": 120,
```

```
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

Con i comandi `deregister-db-proxy-targets` e `register-db-proxy-targets`, puoi modificare i cluster di database Aurora a cui il proxy è associato tramite il relativo gruppo di destinazione. Attualmente, ogni proxy può connettersi a un cluster Aurora . Il gruppo target tiene traccia dei dettagli di connessione per tutte le , tutte le istanze DB in un cluster Aurora.

L'esempio seguente inizia con un proxy associato a un cluster Aurora MySQL denominato `cluster-56-2020-02-25-1399`. Nell'esempio viene illustrato come modificare il proxy in modo che possa connettersi a un cluster diverso denominato `provisioned-cluster`.

Quando utilizzi un cluster di database Aurora, specifichi l'opzione `--db-cluster-identifier`.

L'esempio seguente modifica un proxy Aurora MySQL. Un proxy Aurora PostgreSQL ha la porta 5432.

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
    ]
}

aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy
```

```
{
    "Targets": []
}

aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

API RDS

Per modificare un proxy utilizzando l'API RDS, si utilizzano le operazioni Modify, Modify DBProxyTargetGroup, DBProxyDeregister DBProxy Targets e Register Targets. DBProxy

Con `ModifyDBProxy`, è possibile modificare proprietà come le seguenti:

- L'insieme di segreti Secrets Manager usati dal proxy.
- Indica se TLS è necessario.
- Il timeout del client inattivo.
- Indica se registrare ulteriori informazioni dalle istruzioni SQL per il debug.
- Il ruolo IAM utilizzato per recuperare i segreti Secrets Manager.

- I gruppi di sicurezza utilizzati dal proxy.

Con`ModifyDBProxyTargetGroup`, puoi modificare le impostazioni relative alla connessione. Attualmente, tutti i proxy hanno un singolo gruppo di destinazione denominato `default`. Quando si lavora con questo gruppo target, si specifica il nome del proxy e `default` il nome del gruppo target. Non puoi rinominare il gruppo target predefinito.

Con `DeregisterDBProxyTargets` e`RegisterDBProxyTargets`, si modifica l' a cui il cluster Aurora è associato il proxy tramite il relativo gruppo di destinazione. Attualmente, ogni proxy può connettersi a un cluster Aurora. Il gruppo di destinazione tiene traccia dei dettagli di connessione per le istanze database in un cluster Aurora.

Aggiungere un nuovo utente del database quando si utilizza RDS Proxy

In alcuni casi, è possibile aggiungere un nuovo utente del database a un' di database Aurora associato a un proxy. In tal caso, aggiungi o riutilizza un segreto Secrets Manager per archiviare le credenziali per tale utente. A tale scopo, esegui i seguenti passaggi:

1. Crea un nuovo segreto Secrets Manager, utilizzando la procedura descritta in Configurazione delle credenziali del database in AWS Secrets Manager RDS Proxy.
2. Aggiorna il IAM ruolo per consentire a RDS Proxy l'accesso al nuovo segreto di Secrets Manager. A tale scopo, aggiorna la sezione delle risorse della politica del IAM ruolo.
3. Modifica il RDS Proxy per aggiungere il nuovo segreto di Secrets Manager in Secrets Manager secrets.
4. Se il nuovo utente sostituisce un utente esistente, aggiorna le credenziali memorizzate nel segreto Secrets Manager del proxy per l'utente esistente.

Aggiungere un nuovo utente del database a un database Postgre quando si SQL utilizza Proxy RDS

Quando aggiungi un nuovo utente al tuo SQL database Postgre, se hai eseguito il seguente comando:

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

All'utente `rdsproxyadmin` concedere il privilegio `CONNECT` in modo che possa monitorare le connessioni sul database di destinazione.

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

È anche possibile consentire ad altri utenti del database di destinazione di eseguire controlli dell'integrità passando `rdsproxyadmin` all'utente del database nel comando precedente.

Modifica della password per un utente del database quando si utilizza Proxy RDS

In alcuni casi, è possibile modificare la password per un utente del database in un cluster Aurora di un' associato a un proxy. In tal caso, aggiorna il segreto Secrets Manager corrispondente con la nuova password.

Considerazioni sulla connessione del proxy RDS

Configurazione delle impostazioni di connessione

Per regolare il pooling di connessioni del proxy RDS, è possibile modificare le seguenti impostazioni:

- IdleClientTimeout
- MaxConnectionsPercent
- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

È possibile specificare per quanto tempo una connessione client può rimanere inattiva prima che il proxy la chiuda. Il valore predefinito è 1.800 secondi (30 minuti).

Una connessione client è considerata inattiva quando l'applicazione non invia una nuova richiesta entro il tempo specificato dopo il completamento della richiesta precedente. La connessione al database sottostante rimane aperta e viene restituita al pool di connessioni. Pertanto, è disponibile per essere riutilizzata per nuove connessioni client. Se desideri che il proxy rimuova in modo proattivo le connessioni obsolete, riduci il timeout della connessione del client inattivo. Se il carico di lavoro stabilisce connessioni frequenti con il proxy, aumenta il timeout di connessione del client inattivo per risparmiare sui costi di creazione delle connessioni.

Questa impostazione è rappresentata dal campo di timeout della connessione del client Idle nella console RDS e dall'impostazione in and nell'`IdleClientTimeout`API. AWS CLI Per informazioni su come modificare il valore del campo Idle client connection timeout (Timeout di connessione

client inattivo) nella console RDS, consulta AWS Management Console. Per informazioni su come modificare il valore dell'`IdleClientTimeout`impostazione, consulta il comando CLI modify-db-proxyo l'operazione API Modify. DBProxy

MaxConnectionsPercent

Puoi limitare il numero di connessioni che un proxy RDS può stabilire con il database di destinazione. Puoi specificare il limite come percentuale delle connessioni massime disponibili per il tuo database. Questa impostazione è rappresentata dal campo Connection pool (numero massimo di connessioni) nella console RDS e dall'`MaxConnectionsPercent`impostazione in AWS CLI and (API).

Il valore `MaxConnectionsPercent` viene espresso come percentuale dell'impostazione `max_connections` per il cluster di database Aurora usatocluster dal gruppo di destinazione. Il proxy non crea tutte queste connessioni in anticipo. Questa impostazione consente al proxy di stabilire queste connessioni quando il carico di lavoro le richiede.

Ad esempio, per una destinazione di database registrata con il parametro `max_connections` impostato su 1000 e il parametro `MaxConnectionsPercent` impostato su 95, RDS Proxy imposta 950 connessioni come limite massimo per le connessioni simultanee al database di destinazione specificato.

Un effetto collaterale comune del raggiungimento del numero massimo di connessioni al database consentite dal carico di lavoro è un aumento della latenza complessiva delle query e un incremento del valore della metrica `DatabaseConnectionsBorrowLatency`. È possibile monitorare le connessioni al database attualmente utilizzate e il totale consentito confrontando le metriche `DatabaseConnections` e `MaxDatabaseConnectionsAllowed`.

Se imposti questo parametri, segui le best practice riportate di seguito:

- Consenti un margine sufficiente per le connessioni per la modifica dello schema del carico di lavoro. Si consiglia di impostare il parametro su un valore superiore di almeno il 30% rispetto all'utilizzo massimo monitorato. Poiché RDS Proxy ridistribuisce le quote di connessione del database su più nodi, le modifiche alla capacità interna potrebbero richiedere un margine di almeno il 30% per connessioni aggiuntive per evitare l'incremento delle latenze di prestito.
- RDS Proxy riserva un certo numero di connessioni per il monitoraggio attivo per supportare il failover rapido, l'instradamento del traffico e le operazioni interne. Il parametro `MaxDatabaseConnectionsAllowed` non include queste connessioni riservate. Rappresenta il numero di connessioni disponibili per servire il carico di lavoro e può essere inferiore al valore derivato dall'impostazione `MaxConnectionsPercent`.

`MaxConnectionsPercent`I valori minimi consigliati sono i seguenti:

- db.t3.small: 100
- db.t3.medium: 55
- db.t3.large: 35
- db.r3.large o superiore: 20

Se più istanze di destinazione sono registrate con RDS Proxy come un cluster Aurora con nodi di lettura, imposta il valore minimo in base all'istanza registrata più piccola.

Per informazioni su come modificare il valore del campo Connection pool maximum connections (Numero massimo di connessioni del pool di connessioni) nella console RDS, consulta AWS Management Console. Per informazioni su come modificare il valore dell'`MaxConnectionsPercent`impostazione, consulta il comando CLI modify-db-proxy-target-group o l'operazione API Modify. DBProxy TargetGroup

> **Important**
>
> Se il cluster database fa parte di un database globale con l'inoltro di scrittura attivato, riduci il valore `MaxConnectionsPercent` del proxy con la quota assegnata all'inoltro di scrittura. La quota di inoltro di scrittura è impostata nel parametro `aurora_fwd_writer_max_connections_pct` del cluster database. Per informazioni sull'inoltro di scrittura, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Per informazioni sui limiti di connessione al database, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL e Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

MaxIdleConnectionsPercent

Puoi controllare il numero di connessioni al database inattive che RDS Proxy può mantenere nel pool di connessione. Per impostazione predefinita, RDS Proxy considera inattiva una connessione al database nel suo pool quando non vi è stata alcuna attività sulla connessione per cinque minuti.

Il `MaxIdleConnectionsPercent` valore è espresso come percentuale dell'`max_connections`impostazione per il gruppo target di istanze DB RDS. Il valore

predefinito è 50% di `MaxConnectionsPercent` e il limite superiore è il valore di `MaxConnectionsPercent`. Ad esempio, se`MaxConnectionsPercent`, è 80, il valore predefinito di `MaxIdleConnectionsPercent` è 40.

Con un valore elevato, il proxy lascia aperta un'alta percentuale di connessioni al database inattive. Con un valore basso, il proxy chiude un'alta percentuale di connessioni al database inattive.
Se i carichi di lavoro sono imprevedibili, valuta la possibilità di impostare un valore elevato per `MaxIdleConnectionsPercent`. In tal modo Server proxy per RDS può soddisfare i picchi di attività senza aprire molte nuove connessioni al database.

Questa impostazione è rappresentata dall'`MaxIdleConnectionsPercent`impostazione di `DBProxyTargetGroup` in AWS CLI e nell'API. Per informazioni su come modificare il valore dell'`MaxIdleConnectionsPercent`impostazione, consulta il comando CLI modify-db-proxy-target-group o l'operazione API Modify. DBProxy TargetGroup

Per informazioni sui limiti di connessione al database, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL e Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

ConnectionBorrowTimeout

Puoi specificare per quanto tempo RDS Proxy attende che una connessione di database nel pool di connessione diventi disponibile per l'uso prima di restituire un errore di timeout. Il valore predefinito è 120 secondi. Questa impostazione si applica quando il numero di connessioni è pari al massimo e quindi non sono disponibili connessioni nel pool di connessioni. Si applica anche quando non è disponibile un'istanza di database appropriata per gestire la richiesta, ad esempio quando è in corso un'operazione di failover. Utilizzando questa impostazione, è possibile impostare il periodo di attesa migliore per l'applicazione senza modificare il timeout della query nel codice dell'applicazione.

Questa impostazione è rappresentata dal campo Connection borrow timeout nella console RDS o dall'`ConnectionBorrowTimeout`impostazione di `DBProxyTargetGroup` nell'API o. AWS CLI Per informazioni su come modificare il valore del campo Connection borrow timeout (Timeout del prestito della connessione) nella console RDS, consulta AWS Management Console. Per informazioni su come modificare il valore dell'`ConnectionBorrowTimeout`impostazione, consulta il comando CLI modify-db-proxy-target-group o l'operazione API Modify. DBProxy TargetGroup

Connessioni client e database

Le connessioni dall'applicazione a Server proxy per RDS sono note come connessioni client. Le connessioni da un proxy al database sono le connessioni database. Quando si utilizza Server proxy

per RDS, le connessioni client terminano sul proxy mentre le connessioni database vengono gestite all'interno di Server proxy per RDS.

Il pooling delle connessioni lato applicazione può offrire il vantaggio di ridurre la creazione di connessioni ricorrenti tra l'applicazione e il proxy RDS.

Considerate i seguenti aspetti di configurazione prima di implementare un pool di connessioni lato applicazione:

- Durata massima della connessione client: RDS Proxy impone una durata massima delle connessioni client di 24 ore. Questo valore non è configurabile. Configura il pool con una durata massima della connessione inferiore a 24 ore per evitare interruzioni impreviste della connessione del client.
- Timeout di inattività della connessione client: RDS Proxy impone un tempo di inattività massimo per le connessioni client. Configura il pool con un valore di timeout di inattività della connessione inferiore all'impostazione di timeout di inattività della connessione client per Server proxy per RDS per evitare interruzioni impreviste della connessione.

Il numero massimo di connessioni client configurate nel pool di connessioni lato applicazione non deve essere limitato all'impostazione max_connections per RDS Proxy.

Il pooling delle connessioni dei client comporta una maggiore durata della connessione del client. Se si verifica il pinning delle connessioni, il pool delle connessioni client può ridurre l'efficienza del multiplexing. Le connessioni client bloccate ma inattive nel pool di connessioni lato applicazione continuano a mantenere una connessione al database e impediscono che la connessione al database venga riutilizzata da altre connessioni client. Esamina i log del proxy per verificare se le connessioni presentano problemi di blocco.

> **Note**
>
> RDS Proxy chiude le connessioni al database all'incirca dopo 24 ore quando non sono più in uso. Il proxy esegue questa operazione indipendentemente dal valore dell'impostazione massima delle connessioni inattive.

Evitare il blocco di un proxy RDS

Il multiplexing è più efficiente quando le richieste del database non si basano su informazioni di stato provenienti da richieste precedenti. In tal caso, RDS Proxy può riutilizzare una connessione alla conclusione di ogni transazione. Esempi di tali informazioni sullo stato includono la maggior parte delle variabili e dei parametri di configurazione che puoi modificare attraverso le istruzioni `SET` o `SELECT`. Le transazioni SQL su una connessione client possono eseguire il multiplex tra le connessioni di database sottostanti per impostazione predefinita.

Le connessioni al proxy possono entrare in uno stato noto come pinning. Quando una connessione viene bloccata, ogni transazione successiva utilizza la stessa connessione al database sottostante fino al termine della sessione. Altre connessioni client, inoltre, non possono riutilizzare tale connessione al database fino al termine della sessione. La sessione termina quando viene interrotta la connessione client.

RDS Proxy collega automaticamente una connessione client a una specifica connessione DB quando rileva una modifica dello stato della sessione che non è appropriata per altre sessioni. Il pinning riduce l'efficacia del riutilizzo della connessione. Se tutte o quasi tutte le connessioni riscontrano il pinning, potresti modificare il codice dell'applicazione o il carico di lavoro per ridurre le condizioni che causano il blocco.

Ad esempio, l'applicazione modifica una variabile di sessione o un parametro di configurazione. In questo caso, le istruzioni successive possono basarsi sulla nuova variabile o sul nuovo parametro per essere effettive. Pertanto, quando RDS Proxy elabora le richieste di modifica delle variabili di sessione o delle impostazioni di configurazione, il medesimo effettua il pinning di tale sessione alla connessione DB. In questo modo, lo stato della sessione rimane attivo per tutte le transazioni successive nella stessa sessione.

Per alcuni motori di database, questa regola non si applica a tutti i parametri che puoi impostare. RDS Proxy tiene traccia di determinate istruzioni e variabili. Pertanto, RDS Proxy non blocca la sessione quando la modificate. In tal caso, RDS Proxy riutilizza la connessione solo per altre sessioni con gli stessi valori per tali impostazioni. Per gli elenchi delle istruzioni e delle variabili tracciate per Aurora MySQL, consulta Istruzioni tracciate da Server proxy per RDS per database Aurora MySQL.

Istruzioni tracciate da Server proxy per RDS per database Aurora MySQL

RDS Proxy tiene traccia delle seguenti istruzioni MySQL:

- DROP DATABASE

- DROP SCHEMA
- USE

RDS Proxy tiene traccia delle seguenti variabili MySQL:

- `AUTOCOMMIT`
- `AUTO_INCREMENT_INCREMENT`
- `CHARACTER SET (or CHAR SET)`
- `CHARACTER_SET_CLIENT`
- `CHARACTER_SET_DATABASE`
- `CHARACTER_SET_FILESYSTEM`
- `CHARACTER_SET_CONNECTION`
- `CHARACTER_SET_RESULTS`
- `CHARACTER_SET_SERVER`
- `COLLATION_CONNECTION`
- `COLLATION_DATABASE`
- `COLLATION_SERVER`
- `INTERACTIVE_TIMEOUT`
- `NAMES`
- `NET_WRITE_TIMEOUT`
- `QUERY_CACHE_TYPE`
- `SESSION_TRACK_SCHEMA`
- `SQL_MODE`
- `TIME_ZONE`
- `TRANSACTION_ISOLATION (or TX_ISOLATION)`
- `TRANSACTION_READ_ONLY (or TX_READ_ONLY)`
- `WAIT_TIMEOUT`

> **Note**
>
> RDS Proxy tiene traccia delle modifiche alle `TRANSACTION_READ_ONLY` variabili `TRANSACTION_ISOLATION` and quando le imposti nell'ambito della sessione.

Tuttavia, se le imposti nell'ambito della transazione successiva, il proxy RDS blocca le connessioni. Questo comportamento si applica sia che si utilizzi un'`SET`istruzione o un'`SET TRANSACTION`istruzione per configurare questi valori.

Riduzione dell'associazione

L'ottimizzazione delle prestazioni di RDS Proxy comporta il tentativo di massimizzare il riutilizzo della connessione a livello di transazione (multiplexing) riducendo al minimo il pinning.

È possibile ridurre l'associazione effettuando le seguenti operazioni:

- Evitare richieste di database non necessarie che potrebbero causare il pinning.
- Impostare le variabili e le impostazioni di configurazione in modo coerente su tutte le connessioni. In questo modo, le sessioni successive hanno maggiori probabilità di riutilizzare le connessioni con quelle specifiche impostazioni.

  Tuttavia, per l'impostazione di PostgreSQL una variabile porterà al pinning della sessione.
- Per un database della famiglia di motori MySQL, applica un filtro di pinning della sessione al proxy. Puoi esentare determinati tipi di operazioni dal pinning della sessione se sai che tale operazione non influisce sul corretto funzionamento dell'applicazione.
- Scopri con quale frequenza si verifica il pinning monitorando la CloudWatch metrica `DatabaseConnectionsCurrentlySessionPinned` di Amazon. Per informazioni su questa e altri parametri CloudWatch, consulta Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch.
- Se utilizzi istruzioni `SET` per eseguire l'inizializzazione identica per ogni connessione client, puoi farlo pur mantenendo il multiplexing a livello di transazione. In questo caso, sposta le istruzioni che impostano lo stato della sessione iniziale nella query di inizializzazione utilizzata da un proxy. Questa proprietà è una stringa contenente una o più istruzioni SQL, separate da punto e virgola.

  Ad esempio, puoi definire una query di inizializzazione per un proxy che imposta determinati parametri di configurazione. Quindi, RDS Proxy applica tali impostazioni ogni volta che imposta una nuova connessione per tale proxy. Puoi rimuovere le istruzioni `SET` corrispondenti dal codice dell'applicazione, in modo che non interferiscano con il multiplexing a livello di transazione.

  Per visualizzare i parametri sulla frequenza con cui si verifica il pinning in relazione a un proxy, consulta Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch.

Condizioni che causano il pinning per tutte le famiglie di motori

Il proxy effettua il pinning della sessione alla connessione corrente nelle seguenti situazioni in cui il multiplexing potrebbe causare un comportamento imprevisto:

- Qualsiasi istruzione con una dimensione del testo maggiore di 16 KB fa sì che il proxy effettui il pinning della sessione.

Condizioni che causano l'associazione per Aurora MySQL

Per PostgreSQL, anche le seguenti interazioni causano il pinning:

- Le dichiarazioni di blocco esplicito delle tabelle `LOCK TABLE`, `LOCK TABLES` o `FLUSH TABLES WITH READ LOCK` causano il pinning della sessione da parte del proxy.
- La creazione di blocchi denominati mediante `GET_LOCK` fa sì che il proxy esegua il pinning della sessione.
- L'impostazione di una variabile utente o di sistema (con alcune eccezioni) associa la sessione al proxy. Se ciò limita in modo significativo il riutilizzo della connessione, è possibile configurare `SET` le operazioni per evitare il blocco. A tale scopo, regolate la proprietà dei filtri di blocco della sessione. Per ulteriori informazioni, consulta Creazione di un proxy per e Modifica di un RDS Proxy.
- La creazione di una tabella temporanea fa sì che il proxy effettui il pinning della sessione. In questo modo, il contenuto della tabella temporanea viene conservato per tutta la sessione indipendentemente dai limiti delle transazioni.
- La chiamata alle `FOUND_ROWS` funzioni `ROW_COUNT` and a volte causa il blocco.

  Le circostanze esatte in cui queste funzioni causano il blocco potrebbero differire tra le versioni Aurora MySQL compatibili con MySQL 5.7.
- Le istruzioni preparate fanno sì che il proxy effettui il pinning della sessione. Questa regola si applica se l'istruzione preparata utilizza il testo SQL o il protocollo binario.
- RDS Proxy non blocca le connessioni quando si utilizza SET LOCAL.
- Le chiamate di stored procedure e di funzioni archiviate non causano il pinning. RDS Proxy non rileva alcuna modifica dello stato della sessione derivante da tali chiamate. Assicurati che l'applicazione non modifichi lo stato della sessione all'interno delle routine archiviate se fai affidamento su quello stato della sessione per persistere tra le transazioni. Ad esempio, RDS Proxy

non è attualmente compatibile con una stored procedure che crea una tabella temporanea che persiste in tutte le transazioni.

Se disponi di un'approfondita conoscenza del comportamento dell'applicazione, puoi scegliere di ignorare il comportamento del pinning per determinate istruzioni dell'applicazione. A tale scopo, puoi selezionare l'opzione (Filtri di pinning della sessione durante la creazione del proxy. Attualmente, è possibile disattivare l'aggiunta della sessione per l'impostazione delle variabili di sessione e delle impostazioni di configurazione.

Condizioni che causano l'associazione per Aurora PostgreSQL

Per PostgreSQL, le seguenti interazioni causano anche il pinning:

- Utilizzo dei comandi`SET`.
- Utilizzo di`PREPARE`, `DISCARD``DEALLOCATE`, o `EXECUTE` comandi per gestire le istruzioni preparate.
- Creazione di sequenze, tabelle o viste temporanee.
- Dichiarazione dei cursori.
- Eliminare lo stato della sessione.
- Ascolto su un canale di notifica.
- Caricamento di un modulo di libreria come`auto_explain`.
- Manipolazione di sequenze utilizzando funzioni come e. `nextval` `setval`
- Interazione con le serrature utilizzando funzioni come e. `pg_advisory_lock` `pg_try_advisory_lock`

> **Note**
>
> RDS Proxy non prevede blocchi consultivi a livello di transazione, in particolare`pg_advisory_xact_lock`,, e. `pg_advisory_xact_lock_shared` `pg_try_advisory_xact_lock` `pg_try_advisory_xact_lock_shared`

- Impostazione di un parametro o ripristino dei valori predefiniti di un parametro. In particolare, utilizzo dei `set_config` comandi `SET` and per assegnare valori predefiniti alle variabili di sessione.
- Le chiamate di stored procedure e di funzioni archiviate non causano il pinning. RDS Proxy non rileva alcuna modifica dello stato della sessione derivante da tali chiamate. Assicurati che l'applicazione non modifichi lo stato della sessione all'interno delle routine archiviate se fai

affidamento su quello stato della sessione per persistere tra le transazioni. Ad esempio, RDS Proxy non è attualmente compatibile con una stored procedure che crea una tabella temporanea che persiste in tutte le transazioni.

Eliminazione di un proxy RDS

È possibile eliminare un proxy quando non è più necessario. In alternativa, puoi eliminare un proxy se metti fuori servizio l'istanza DB o il cluster ad esso associato.

AWS Management Console

Per eliminare un proxy

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Scegli il proxy da eliminare dall'elenco.
4. Scegli Delete Proxy (Elimina proxy).

AWS CLI

Per eliminare un proxy DB, usa il AWS CLI comando delete-db-proxy. Per rimuovere le associazioni correlate, usa anche il deregister-db-proxy-targetscomando.

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
    [--db-cluster-identifiers cluster_id]
```

RDS API

Per eliminare un proxy DB, chiama la RDS API funzione Amazon eleteDBProxyD. Per eliminare elementi e associazioni correlati, richiami anche le funzioni D eleteDBProxy TargetGroup e D eregisterDBProxy Targets.

Utilizzo degli endpoint Amazon RDS Proxy

Gli endpoint proxy RDS offrono modi flessibili ed efficienti per gestire le connessioni al database, migliorando la scalabilità, la disponibilità e la sicurezza. Con gli endpoint proxy, puoi:

- Semplifica il monitoraggio e la risoluzione dei problemi: utilizza più endpoint per tracciare e gestire le connessioni da diverse applicazioni in modo indipendente.
- Migliora la scalabilità di lettura: sfrutta gli endpoint dei lettori con i cluster Aurora DB per distribuire il traffico di lettura in modo efficiente, ottimizzando le prestazioni per carichi di lavoro con un elevato numero di query.
- Abilita l'accesso tra VPC: collega i database VPCs utilizzando endpoint cross-VPC, che consentono a risorse come le istanze Amazon in un VPC di accedere ai database EC2 in un altro VPC.

Argomenti

- Panoramica degli endpoint proxy
- Limitazioni per gli endpoint proxy
- Utilizzo degli endpoint di lettura con cluster Aurora
- Accesso ai database Aurora RDS VPCs
- Creazione di un endpoint proxy
- Visualizzazione degli endpoint proxy
- Modifica di un endpoint proxy
- Eliminazione di un endpoint proxy

Panoramica degli endpoint proxy

L'uso degli endpoint Server proxy per RDS include gli stessi tipi di procedure degli endpoint di cluster di database Aurora e di lettura. Se non hai familiarità con gli endpoint Aurora, puoi consultare Connessioni endpoint Amazon Aurora.

Quando si utilizza RDS Proxy con un cluster Aurora, l'endpoint predefinito supporta operazioni di lettura e scrittura. Ciò significa che indirizza tutte le richieste all'istanza writer del cluster, contribuendo al suo limite. `max_connections` Per distribuire meglio il traffico, puoi creare endpoint di lettura/scrittura o di sola lettura aggiuntivi per il tuo proxy, il che consente una gestione più efficiente del carico di lavoro e una maggiore scalabilità.

Usa un endpoint di sola lettura con il tuo proxy per gestire le query di lettura, proprio come faresti con un endpoint reader per un cluster con provisioning Aurora. Questo approccio massimizza la scalabilità di lettura di Aurora distribuendo le query su una o più istanze DB di Reader. Utilizzando un endpoint di sola lettura e aggiungendo altre istanze di lettura in base alle esigenze, è possibile aumentare il numero di query e connessioni simultanee che il cluster è in grado di gestire.

> **Tip**
>
> Quando si crea un proxy per un cluster Aurora utilizzando AWS Management Console, è possibile fare in modo che RDS Proxy crei automaticamente un endpoint di lettura. Per informazioni sui vantaggi di un endpoint di lettura, consulta Utilizzo degli endpoint di lettura con cluster Aurora.

Quando crei un endpoint proxy, puoi associarlo a un cloud privato virtuale (VPC) diverso dal VPC del proxy. Ciò consente di connettersi al proxy da un altro VPC, ad esempio uno utilizzato da un'applicazione diversa all'interno dell'organizzazione.

Per informazioni sui limiti associati agli endpoint proxy, consulta Limitazioni per gli endpoint proxy.

I log del proxy RDS fanno precedere a ogni voce il nome dell'endpoint proxy associato. Può essere il nome specificato per un endpoint definito dall'utente o il nome speciale per l'endpoint di lettura/scrittura predefinito del `default` proxy.

Ogni endpoint proxy ha il proprio set di metriche. CloudWatch Monitora le metriche per tutti gli endpoint proxy, un endpoint specifico o tutti gli endpoint di lettura/scrittura o di sola lettura di un proxy. Per ulteriori informazioni, consulta Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch.

Un endpoint proxy utilizza lo stesso meccanismo di autenticazione del proxy associato. RDS Proxy imposta automaticamente i permessi e le autorizzazioni per l'endpoint definito dall'utente, coerenti con le proprietà del proxy associato.

Per informazioni su come funzionano gli endpoint proxy per i cluster database di un database globale Aurora, consulta Come funzionano gli endpoint Server proxy per RDS con i database globali.

Limitazioni per gli endpoint proxy

Gli endpoint proxy RDS presentano le seguenti limitazioni:

- L'endpoint predefinito del proxy RDS non può essere modificato.
- Il numero massimo di endpoint definiti dall'utente per un proxy è 20. Pertanto, un proxy può avere fino a 21 endpoint: l'endpoint predefinito più 20 creati.
- Quando associ degli endpoint aggiuntivi a un proxy, RDS Proxy determina automaticamente quali istanze database nel cluster utilizzare per ciascun endpoint. Non è possibile scegliere istanze specifiche come invece è possibile con gli endpoint personalizzati Aurora.

Quando si crea un proxy, RDS crea automaticamente un endpoint VPC per comunicazioni sicure tra le applicazioni e il database. L'endpoint VPC è visibile ed è possibile accedervi dalla console Amazon VPC.

L'aggiunta di un nuovo endpoint proxy fornisce un endpoint di interfaccia. AWS PrivateLink Se aggiungi uno o più endpoint al tuo proxy, dovrai sostenere costi aggiuntivi. Per ulteriori informazioni, consulta RDS Proxy Pricing.

Utilizzo degli endpoint di lettura con cluster Aurora

Puoi creare e connetterti a endpoint di sola lettura denominati endpoint di lettura quando utilizzi RDS Proxy con cluster Aurora. Questi endpoint di lettura consentono di migliorare la scalabilità di lettura delle applicazioni che richiedono un uso intensivo di query. Gli endpoint di lettura consentono inoltre di migliorare la disponibilità delle connessioni se un'istanza database di lettura nel cluster non è disponibile.

> **Note**
>
> Quando si specifica che un nuovo endpoint è di sola lettura, RDS Proxy richiede che il cluster Aurora abbia una o più istanze database di lettura. In alcuni casi, è possibile modificare la destinazione del proxy in un cluster Aurora contenente un solo writer. In questo caso, qualsiasi richiesta all'endpoint di lettura ha esito negativo e restituisce un errore. Anche le richieste avranno esito negativo se la destinazione del proxy è un'istanza RDS invece di un cluster Aurora.
>
> Se un cluster Aurora ha istanze di lettura ma tali istanze non sono disponibili, RDS Proxy attende per inviare la richiesta invece di restituire immediatamente un errore. Se nessuna istanza di lettura diventa disponibile entro il periodo di timeout di prestito della connessione, la richiesta ha esito negativo con un errore.

In che modo gli endpoint di lettura aiutano la disponibilità delle applicazioni

In alcuni casi, una o più istanze di lettura nel cluster potrebbero non essere disponibili. In questo caso, le connessioni che utilizzano un endpoint di lettura di un proxy DB possono essere ripristinate più rapidamente di quelle che utilizzano l'endpoint di lettura Aurora. RDS Proxy instrada le connessioni solo alle istanze del lettore disponibili nel cluster. Non vi è alcun ritardo dovuto alla memorizzazione nella cache DNS quando un'istanza diventa non disponibile.

Se la connessione è multiplexing, RDS Proxy indirizza le query successive a un'istanza database di lettura diversa senza alcuna interruzione dell'applicazione. Durante il passaggio automatico a una nuova istanza di lettura, RDS Proxy controlla il ritardo della replica delle istanze di lettura vecchie e nuove. RDS Proxy assicura che la nuova istanza di lettura sia aggiornata con le stesse modifiche dell'istanza di lettura precedente. In questo modo, la tua applicazione non vedrà mai dati obsoleti quando RDS Proxy passa da un'istanza database di lettura a un'altra.

Se la connessione è bloccata, la successiva query sulla connessione restituisce un errore. Tuttavia, l'applicazione può riconnettersi immediatamente allo stesso endpoint. RDS Proxy indirizza la connessione a un'istanza database del lettore diversa che si trova nello stato `available`. Se ti riconnetti manualmente, RDS Proxy non controlla il ritardo di replica tra le vecchie e le nuove istanze di lettura.

Se il tuo cluster Aurora non dispone di istanze di lettura disponibili, RDS Proxy verifica se questa condizione è temporanea o permanente. Il comportamento in ogni caso è il seguente:

- Si assuma che il cluster abbia una o più istanze database di lettura, ma nessuna di esse si trova nello stato `Available`. Ad esempio, tutte le istanze di lettura potrebbero essere riavviate o potrebbero riscontrare dei problemi. In tal caso, i tentativi di connessione a un endpoint di lettura attendono la disponibilità di un'istanza di lettura. Se nessuna istanza di lettura diventa disponibile entro il periodo di timeout di prestito della connessione, il tentativo di connessione ha esito negativo. Se invece un'istanza di lettura diventa disponibile, il tentativo di connessione ha esito positivo.
- Si assuma che il cluster non abbia istanze database di lettura. In tal caso, RDS Proxy restituisce immediatamente un errore se si prova a connettersi a un endpoint di lettura. Per risolvere il problema, aggiungi una o più istanze di lettura al cluster prima di connetterti all'endpoint di lettura.

In che modo gli endpoint di lettura aiutano la scalabilità delle query

Gli endpoint di lettura per un proxy aiutano con la scalabilità delle query di Aurora nei seguenti modi:

- Quando aggiungi istanze di lettura al cluster Aurora, RDS Proxy può instradare nuove connessioni a qualsiasi endpoint di lettura a istanze di lettura differenti. In questo modo, le query eseguite utilizzando una connessione all'endpoint di lettura non rallentano le query eseguite utilizzando un'altra connessione all'endpoint di lettura. Le query vengono eseguite su istanze database separate. Ogni istanza database ha le proprie risorse di calcolo, cache di buffer e così via.
- Laddove possibile, RDS Proxy utilizza la stessa istanza database di lettura per tutti i problemi di query utilizzando una particolare connessione all'endpoint di lettura. In questo modo, un insieme di query correlate sulle stesse tabelle può sfruttare la memorizzazione nella cache, l'ottimizzazione del piano e così via, su una particolare istanza database.
- Se un'istanza database di lettura non è disponibile, l'effetto sull'applicazione dipende dal fatto che la sessione sia multiplexing o bloccata. Se la sessione è multiplexing, RDS Proxy indirizza tutte le query successive a un'istanza database di lettura diversa senza richiedere alcuna azione da parte tua. Se invece la sessione è bloccata, l'applicazione riceve un errore e deve riconnettersi. Puoi riconnetterti immediatamente all'endpoint di lettura e RDS Proxy indirizza la connessione a un'istanza database di lettura disponibile. Per ulteriori informazioni sul multiplexing e sul blocco per le sessioni proxy, consulta Panoramica dei concetti relativi al proxy RDS.
- Maggiore è il numero di istanze database di lettura presenti nel cluster, maggiore sarà il numero di connessioni simultanee che è possibile effettuare utilizzando gli endpoint di lettura. Supponi, ad esempio, che il cluster disponga di quattro istanze database di lettura, ciascuna configurata per supportare 200 connessioni simultanee. Supponi anche che il proxy sia configurato per utilizzare il 50% del numero massimo di connessioni. Qui, il numero massimo di connessioni che è possibile effettuare attraverso gli endpoint di lettura nel proxy è 100 (50% di 200) per la lettura 1. È 100 anche per la lettura 2, e così via, per un totale di 400. Se raddoppi il numero di istanze database di lettura del cluster a otto, anche il numero massimo di connessioni attraverso gli endpoint di lettura raddoppia a 800.

Esempi di utilizzo degli endpoint di lettura

L'esempio seguente di Linux mostra come è possibile confermare di essere connessi a un cluster Aurora MySQL tramite un endpoint di lettura. L'impostazione di configurazione di `innodb_read_only` è abilitata. I tentativi di esecuzione di operazioni di scrittura come le istruzioni `CREATE DATABASE` non riescono e restituiscono un errore. Inoltre, puoi confermare di essere connesso a un'istanza database di lettura controllando il nome dell'istanza database utilizzando la variabile `aurora_server_id`.

> **Tip**
>
> Non fare affidamento solo sul controllo del nome dell'istanza database per determinare se la connessione è di lettura/scrittura o di sola lettura. Ricorda che le istanze database in un cluster Aurora possono cambiare ruolo tra scrittura e lettura quando si verificano i failover.

```
$ mysql -h endpoint-demo-reader.endpoint.proxy-demo.us-east-1.rds.amazonaws.com -u
 admin -p
...
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
mysql> create database shouldnt_work;
ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it
 cannot execute this statement

mysql> select @@aurora_server_id;
+---------------------------------------+
| @@aurora_server_id                    |
+---------------------------------------+
| proxy-reader-endpoint-demo-instance-3 |
+---------------------------------------+
```

L'esempio seguente mostra come la connessione a un endpoint di lettura proxy può continuare a funzionare anche quando l'istanza database di lettura viene eliminata. In questo esempio, il cluster Aurora dispone di due istanze di lettura, `instance-5507` e `instance-7448` . La connessione all'endpoint di lettura inizia utilizzando una delle istanze di lettura. Durante l'esempio, questa istanza di lettura viene eliminata da un comando `delete-db-instance`. RDS Proxy passa a un'istanza di lettura diversa per le query successive.

```
$ mysql -h reader-demo.endpoint.proxy-demo.us-east-1.rds.amazonaws.com
  -u my_user -p
...
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
```

```
| instance-5507      |
+--------------------+

mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+

mysql> select count(*) from information_schema.tables;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Mentre la sessione `mysql` è ancora in esecuzione, il comando seguente elimina l'istanza di lettura a cui è connesso l'endpoint di lettura.

```
aws rds delete-db-instance --db-instance-identifier instance-5507 --skip-final-snapshot
```

Le query nella sessione `mysql` continuano a funzionare senza la necessità di riconnettersi. RDS Proxy passa quindi automaticamente a un'istanza database di lettura diversa.

```
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-7448      |
+--------------------+

mysql> select count(*) from information_schema.TABLES;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Accesso ai database Aurora RDS VPCs

Per impostazione predefinita, i componenti dello stack di tecnologia di Aurora si trovano tutti nello stesso Amazon VPC. Ad esempio, supponiamo che un'applicazione in esecuzione su un' EC2 istanza Amazon si connetta a un'istanza database del cluster Aurora DB. In questo caso, il server delle applicazioni e il database devono trovarsi entrambi all'interno dello stesso VPC.

Con RDS Proxy, puoi configurare l'accesso a un'istanza in un VPC dalle risorse di un altro VPC, come le istanze. EC2 Ad esempio, l'organizzazione potrebbe avere più applicazioni che accedono alle stesse risorse del database. Ogni applicazione potrebbe trovarsi nel proprio VPC.

Per consentire l'accesso tra VPC, crea un nuovo endpoint per il proxy. Il proxy stesso risiede nello stesso VPC del cluster di database Aurora . Tuttavia, l'endpoint cross-VPC risiede nell'altro VPC, insieme alle altre risorse come le istanze. EC2 L'endpoint cross-VPC è associato a sottoreti e gruppi di sicurezza dello stesso VPC delle risorse e di altre risorse. EC2 Queste associazioni consentono di connettersi all'endpoint dalle applicazioni che altrimenti non potrebbero accedere al database a causa delle restrizioni del VPC.

Nella procedura seguente viene illustrato come creare e accedere a un endpoint tra VPC tramite RDS Proxy:

1. Creane due VPCs o scegline due VPCs che già usi per il lavoro con Aurora . Ogni VPC dovrebbe avere le proprie risorse di rete associate come un gateway Internet, tabelle di routing, sottoreti e gruppi di sicurezza. Se si dispone di un solo VPC, è possibile consultare la procedura Nozioni di base su Amazon Aurora per configurare un altro VPC per l'utilizzo corretto di Aurora. Puoi anche esaminare il tuo VPC esistente nella EC2 console Amazon per vedere i tipi di risorse da connettere tra loro.
2. Crea un proxy database associato al cluster di database Aurora a cui desideri connetterti. Segui la procedura riportata in Creazione di un proxy per .
3. Nella pagina Dettagli del proxy nella console RDS, nella sezione Endpoint proxy seleziona Crea endpoint. Segui la procedura riportata in Creazione di un endpoint proxy.
4. Seleziona se impostare l'endpoint tra VPC in lettura/scrittura o in sola lettura.
5. Invece di accettare il valore predefinito dello stesso VPC del cluster di database Aurora , seleziona un VPC diverso. Questo VPC deve trovarsi nella stessa AWS regione del VPC in cui risiede il proxy.

6. Ora invece di accettare i valori predefiniti per le sottoreti e i gruppi di sicurezza dallo stesso VPC del cluster di database Aurora , effettua nuove selezioni. Le nuove selezioni dovranno essere relative alle sottoreti e ai gruppi di sicurezza del VPC scelto.
7. Non è necessario modificare alcuna delle impostazioni per i segreti di Secrets Manager. Le stesse credenziali funzionano per tutti gli endpoint del proxy, indipendentemente dal VPC in cui si trova ciascun endpoint.
8. Attendi che il nuovo endpoint raggiunga lo stato Disponibile.
9. Prendi nota del nome completo dell'endpoint. Questo è il valore che termina in *`Region_name`*`.rds.amazonaws.com` fornito come parte della stringa di connessione per l'applicazione di database.
10. Accedi al nuovo endpoint da una risorsa nello stesso VPC dell'endpoint. Un modo semplice per testare questo processo è creare una nuova EC2 istanza in questo VPC. Quindi, accedi all' EC2 istanza ed `psql` esegui i comandi `mysql` or per connetterti utilizzando il valore dell'endpoint nella stringa di connessione.

Creazione di un endpoint proxy

Per creare un endpoint proxy, segui queste istruzioni:

Console

Per creare un endpoint proxy

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Fai clic sul nome del proxy per il quale desideri creare un nuovo endpoint.

   Sarà visualizzata la pagina dei dettagli del proxy.
4. In Endpoint proxy, seleziona Crea endpoint proxy.

   Verrà visualizzata la finestra Crea endpoint proxy.
5. Per Nome endpoint proxy, specifica un nome descrittivo a scelta.
6. Per Ruolo di destinazione, scegli se rendere l'endpoint di lettura/scrittura o di sola lettura.

   Le connessioni che utilizzano endpoint di lettura/scrittura possono eseguire qualsiasi tipo di operazione, ad esempio istruzioni Data Definition Language (DDL), istruzioni Data Manipulation

Language (DML) e query. Questi endpoint si connettono sempre all'istanza principale del cluster DB del Aurora. Puoi utilizzare gli endpoint di lettura/scrittura per le operazioni generali del database nel caso in cui utilizzi un singolo endpoint nell'applicazione. È inoltre possibile utilizzare endpoint di lettura/scrittura per operazioni amministrative, applicazioni di elaborazione delle transazioni online (OLTP) e lavori (). extract-transform-load ETL

Le connessioni che utilizzano un endpoint di sola lettura possono soltanto eseguire query. Quando sono presenti più istanze di lettura nel cluster AuroraRDS, Proxy può utilizzare un'istanza di lettura diversa per ogni connessione all'endpoint. In questo modo, un'applicazione ad uso intensivo di query può sfruttare i vantaggi del clustering di Aurora. Puoi aggiungere più capacità di query al cluster aggiungendo più istanze database di lettura. Queste connessioni di sola lettura non impongono alcun sovraccarico sull'istanza primaria del cluster. In questo modo, le tue query di reporting e analisi non rallentano le operazioni di scrittura delle tue applicazioni. OLTP

7. Per Virtual Private Cloud (VPC), scegli l'impostazione predefinita per accedere all'endpoint dalle stesse EC2 istanze o altre risorse normalmente utilizzate per accedere al proxy o al database associato. Per configurare VPC l'accesso incrociato per questo proxy, scegli un accesso VPC diverso da quello predefinito. Per ulteriori informazioni sull'VPCaccesso incrociato, vedereAccesso ai database Aurora RDS VPCs.
8. Per le sottoreti, per RDS impostazione predefinita, Proxy compila le stesse sottoreti del proxy associato. Per limitare l'accesso all'endpoint solo a una parte dell'intervallo di indirizzi in cui VPC è possibile connettersi ad esso, rimuovi una o più sottoreti.
9. Per il gruppo VPC di sicurezza, puoi scegliere un gruppo di sicurezza esistente o crearne uno nuovo. RDSPer impostazione predefinita, il proxy compila lo stesso gruppo o gli stessi gruppi di sicurezza del proxy associato. Se le regole in entrata e in uscita per il proxy sono appropriate per questo endpoint, mantieni la scelta predefinita.

   Se decidi di creare un nuovo gruppo di sicurezza, specifica un nome per il gruppo di sicurezza in questa pagina. Quindi modifica le impostazioni del gruppo di sicurezza dalla console in un secondo momento. EC2
10. Scegli Crea endpoint proxy.

AWS CLI

Per creare un endpoint proxy, usa il AWS CLI create-db-proxy-endpointcomando.

Includi i parametri obbligatori seguenti:

- `--db-proxy-name value`
- `--db-proxy-endpoint-name value`
- `--vpc-subnet-ids list_of_ids`. Separa la sottorete IDs con degli spazi. Non si specifica l'ID del file VPC stesso.

Puoi inoltre includere i seguenti parametri facoltativi:

- `--target-role { READ_WRITE | READ_ONLY }`. Questo parametro per impostazione predefinita è `READ_WRITE`. Il valore `READ_ONLY` ha un effetto solo su cluster Aurora con provisioning che contengono una o più istanze database di lettura. Quando il proxy è associato a Aurora che contiene solo un'istanza Writer DB, non è possibile specificare. `READ_ONLY` Utilizzo degli endpoint di lettura con cluster Aurora
- `--vpc-security-group-ids value`. Separa il gruppo di sicurezza IDs con degli spazi. Se si omette questo parametro, RDS Proxy utilizza il gruppo di sicurezza predefinito per. VPC RDSII proxy determina il valore in VPC base alla sottorete IDs specificata per il `--vpc-subnet-ids` parametro.

Example

Nell'esempio seguente viene creato un endpoint proxy denominato `my-endpoint`.

Per LinuxmacOS, oUnix:

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id ]
```

Per Windows:

```
aws rds create-db-proxy-endpoint ^
  --db-proxy-name my-proxy ^
  --db-proxy-endpoint-name my-endpoint ^
  --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
  --target-role READ_ONLY ^
  --vpc-security-group-ids security_group_id
```

RDS API

Per creare un endpoint proxy, utilizzare l'azione RDS API C reateDBProxy Endpoint.

Visualizzazione degli endpoint proxy

Per visualizzare gli endpoint proxy esistenti, segui queste istruzioni:

Console

Per visualizzare i dettagli di un endpoint proxy

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Nell'elenco scegli il proxy di cui si desidera visualizzare l'endpoint. Fai clic sul nome del proxy per visualizzarne la pagina dei dettagli.
4. Nella sezione Endpoint proxy, scegli l'endpoint che desideri visualizzare. Fai clic sul relativo nome per visualizzare la pagina dei dettagli.
5. Esamina i parametri di cui ti interessano i valori. Puoi controllare proprietà come le seguenti:
   - Se l'endpoint è di lettura/scrittura o di sola lettura.
   - L'indirizzo dell'endpoint utilizzato in una stringa di connessione al database.
   - Le VPC sottoreti e i gruppi di sicurezza associati all'endpoint.

AWS CLI

Per visualizzare uno o più endpoint proxy, utilizzare il comando. AWS CLI describe-db-proxy-endpoints

Puoi includere i seguenti parametri facoltativi:

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

Nell'esempio seguente viene descritto l'endpoint proxy `my-endpoint`.

Example

Per LinuxmacOS, oUnix:

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

Per Windows:

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

RDS API

Per descrivere uno o più endpoint proxy, utilizzare l'operazione RDS API D escribeDBProxy Endpoints.

Modifica di un endpoint proxy

Per modificare gli endpoint proxy, segui queste istruzioni:

Console

Per modificare uno o più endpoint proxy

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione scegli Proxies (Proxy).
3. Nell'elenco seleziona il proxy di cui desideri modificare l'endpoint. Fai clic sul nome del proxy per visualizzarne la pagina dei dettagli.
4. Nella sezione Endpoint proxy, scegli l'endpoint che desideri modificare. Puoi selezionarlo dall'elenco oppure fare clic sul relativo nome e visualizzarne la pagina dei dettagli.
5. Nella pagina dei dettagli del proxy, sotto la sezione Endpoint proxy, seleziona Modifica. Oppure, nella pagina dei dettagli dell'endpoint proxy, per Azioni, scegli Modifica.
6. Modificare i valori dei parametri desiderati.
7. Scegli Save changes (Salva modifiche).

AWS CLI

Per modificare un endpoint proxy, utilizza il AWS CLI modify-db-proxy-endpointcomando con i seguenti parametri richiesti:

- `--db-proxy-endpoint-name`

Specifica le modifiche alle proprietà dell'endpoint utilizzando uno o più dei seguenti parametri:

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`. Separa il gruppo di sicurezza IDs con degli spazi.

Nel seguente esempio l'endpoint proxy `my-endpoint` viene ridenominato in `new-endpoint-name`.

Example

In Linux, macOS, oppure Unix:

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
  --new-db-proxy-endpoint-name new-endpoint-name
```

In Windows:

```
aws rds modify-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

RDS API

Per modificare un endpoint proxy, utilizzare l'operazione RDS API M odifyDBProxy Endpoint.

Eliminazione di un endpoint proxy

Per eliminare un endpoint per il tuo proxy, segui queste istruzioni:

> **Note**
>
> Non puoi eliminare l'endpoint proxy predefinito che RDS Proxy crea automaticamente per ogni proxy.

Quando si elimina un proxy, RDS Proxy elimina automaticamente tutti gli endpoint associati.

Console

Per eliminare un endpoint proxy utilizzando il AWS Management Console

1. Nel riquadro di navigazione scegli Proxies (Proxy).
2. Nell'elenco seleziona il proxy per cui desideri eliminare l'endpoint. Fai clic sul nome del proxy per visualizzarne la pagina dei dettagli.
3. Nella sezione Endpoint proxy, scegli l'endpoint che desideri eliminare. Puoi selezionare uno o più endpoint dall'elenco oppure fare clic sul nome di un singolo endpoint e visualizzarne la pagina dei dettagli.
4. Nella pagina dei dettagli del proxy, sotto la sezione Endpoint proxy, seleziona Elimina. Oppure, nella pagina dei dettagli dell'endpoint proxy, per Azioni, scegli Elimina.

AWS CLI

Per eliminare un endpoint proxy, esegui il delete-db-proxy-endpointcomando con i seguenti parametri richiesti:

- `--db-proxy-endpoint-name`

Il comando seguente elimina l'endpoint proxy denominato `my-endpoint`.

Per LinuxmacOS, oUnix:

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

Per Windows:

```
aws rds delete-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint
```

RDS API

Per eliminare un endpoint proxy con RDSAPI, esegui l'operazione D eleteDBProxy Endpoint. Specifica il nome dell'endpoint proxy per il parametro `DBProxyEndpointName`.

Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch

Puoi monitorare RDS Proxy utilizzando Amazon CloudWatch. CloudWatch raccoglie ed elabora i dati grezzi dai proxy in metriche leggibili. near-real-time Per trovare queste metriche nella CloudWatch console, scegli Metriche, quindi scegli RDS e scegli Metriche per proxy. Per ulteriori informazioni, consulta Using Amazon CloudWatch metrics nella Amazon CloudWatch User Guide.

> **Note**
>
> RDS pubblica questi parametri per ogni EC2 istanza Amazon sottostante associata a un proxy. Un singolo proxy potrebbe essere servito da più di un'istanza. EC2 Utilizza CloudWatch le statistiche per aggregare i valori di un proxy in tutte le istanze associate. Alcuni di questi parametri potrebbero non essere visibili fino al completamento della prima connessione mediante un proxy.

Nei log RDS Proxy, ogni voce è preceduta dal nome dell'endpoint proxy associato. Questo nome può essere il nome specificato per un endpoint definito dall'utente o il nome speciale `default` per l'endpoint predefinito di un proxy che esegue le richieste. read/write

Tutte le metriche RDS Proxy sono nel gruppo `proxy`.

Ogni endpoint proxy ha le proprie metriche. CloudWatch Puoi monitorare l'utilizzo di ciascun endpoint proxy in modo indipendente. Per ulteriori informazioni sugli endpoint proxy, consulta Utilizzo degli endpoint Amazon RDS Proxy.

Puoi aggregare i valori per ogni parametro utilizzando uno dei seguenti set di dimensioni. Ad esempio, utilizzando il metodo `ProxyName`, puoi analizzare tutto il traffico per un determinato proxy. Utilizzando gli altri set di dimensioni, puoi suddividere i parametri in modi diversi. Puoi suddividere le metriche in base ai diversi endpoint o database di destinazione di ciascun proxy oppure al read/write traffico di sola lettura verso ciascun database.

- Set di dimensioni 1: `ProxyName`
- Set di dimensioni 2: `ProxyName`, `EndpointName`
- Set di dimensioni 3: `ProxyName`, `TargetGroup`, `Target`
- Set di dimensioni 4: `ProxyName`, `TargetGroup`, `TargetRole`

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`AvailabilityPercentage`	Percentuale di tempo durante il quale il gruppo target era disponibile nel ruolo indicato dalla dimensione. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 4
`ClientConnections`	Il numero corrente di connessioni client. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 2
`ClientConnectionsClosed`	Il numero di connessioni client chiuse. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`ClientConnectionsInSetup`	Il numero attuale di connessioni client è aperto ma la configurazione non è stata completata. Questa metrica	1 minuto	Dimension set 1, Dimension set 2

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
	viene segnalata ogni minuto. La statistica più utile per questo parametro è Sum.		
`ClientConnectionsNoTLS`	Numero corrente di connessioni client senza Transport Layer Security (TLS). Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 2
`ClientConnectionsReceived`	Numero di richieste di connessione client ricevute. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`ClientConnectionsSetupFailedAuth`	Numero di tentativi di connessione client non riusciti a causa di autenticazione errata o configurazione TLS errata. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`ClientConnectionsSetupSucceeded`	Il numero di connessioni client stabilito correttamente con qualsiasi meccanismo di autenticazione con o senza TLS. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`ClientConnectionsTLS`	Il numero corrente di connessioni client con TLS. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 2
`DatabaseConnectionRequests`	Numero di richieste per creare una connessione al database. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionRequestsWithTLS`	Numero di richieste per creare una connessione al database con TLS. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`DatabaseConnections`	Il numero corrente di connessioni al database. Questa metrica viene segnalata ogni minuto. La statistic a più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsBorrowLatency`	Il tempo in microsecondi necessario per il proxy monitorato per ottenere una connessione al database. La statistica più utile per questo parametro è `Average`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`DatabaseConnectionsCurrentlyBorrowed`	Il numero corrente di connessioni al database nello stato di prestito. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`DatabaseConnectionsCurrentlyInTransaction`	Il numero corrente di connessioni al database in una transazione. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlySessionPinned`	Il numero corrente di connessioni al database attualmente bloccati a causa di operazioni nelle richieste client che modificano lo stato della sessione. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsSetupFailed`	Il numero di richieste di connessione al database non riuscite. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`DatabaseConnectionsSetupSucceeded`	Il numero di connessioni al database stabilito correttamente con o senza TLS. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsWithTLS`	Il numero corrente di connessioni al database con TLS. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4
`MaxDatabaseConnectionsAllowed`	Il numero massimo di connessioni al database consentite. Questa metrica viene segnalata ogni minuto. La statistica più utile per questo parametro è `Average`.	1 minuto	Dimension set 1, Dimension set 3, Dimension set 4

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`QueryDatabaseResponseLatency`	Tempo in microsecondi impiegato dal database per rispondere alla query. La statistica più utile per questo parametro è `Average`.	1 minuto e oltre	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4
`QueryRequests`	Il numero di query ricevute. Una query che include più istruzioni viene conteggiata come una query. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`QueryRequestsNoTLS`	Numero di query ricevute da connessioni non TLS. Una query che include più istruzioni viene conteggiata come una query. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2

Parametro	Descrizione	Periodo valido	CloudWatch set di dimensioni
`QueryRequestsTLS`	Numero di query ricevute dalle connessioni TLS. Una query che include più istruzioni viene conteggiata come una query. La statistica più utile per questo parametro è `Sum`.	1 minuto e oltre	Dimension set 1, Dimension set 2
`QueryResponseLatency`	Il tempo in microsecondi tra l'ottenimento di una richiesta di query e il proxy che risponde ad essa. La statistica più utile per questo parametro è `Average`.	1 minuto e oltre	Dimension set 1, Dimension set 2

È possibile trovare i registri delle attività del proxy RDS CloudWatch in. AWS Management Console Ogni proxy ha una voce nella pagina Log groups (Gruppi di registro) .

> **Important**
>
> Questi registri sono destinati all'utilizzo umano per scopi di risoluzione dei problemi e non per l'accesso programmatico. Il formato e il contenuto dei registri sono soggetti a modifiche.
> In particolare, i log meno recenti non contengono prefissi che indicano l'endpoint per ogni richiesta. Nei log più recenti, ogni voce è preceduta dal nome dell'endpoint proxy associato. Questo nome può essere il nome specificato per un endpoint definito dall'utente o il nome speciale `default` per le richieste che utilizzano l'endpoint predefinito di un proxy.

Utilizzo degli eventi RDS Proxy

Un evento indica una modifica in un ambiente, ad esempio un AWS ambiente o un servizio o un'applicazione di un partner Software as a Service (SaaS). In alternativa, può essere una delle vostre applicazioni o servizi personalizzati. Ad esempio, Aurora genera un evento quando crei o modifichi un RDS proxy. Amazon Aurora offre eventi ad Amazon quasi EventBridge in tempo reale. Di seguito, puoi trovare un elenco di eventi RDS Proxy a cui puoi iscriverti e un esempio di evento Proxy. RDS

Per ulteriori informazioni sull'utilizzo degli eventi, consulta quanto segue:

- Per istruzioni su come visualizzare gli eventi utilizzando AWS Management Console AWS CLI, o RDSAPI, vedi Visualizzazione degli RDS eventi Amazon.
- Per informazioni su come configurare Aurora a cui inviare eventi EventBridge, consulta. Creazione di una regola che si attiva su un evento Amazon Aurora

RDSEventi proxy

La tabella seguente mostra la categoria di eventi e un elenco di eventi quando il tipo di origine è un RDS proxy.

Categoria	ID evento RDS	Messaggio	Note
modifica della configura zione	RDS- EVENT -0204	RDSproxy DB modificato *name*.	Nessuno
modifica della configura zione	RDS- EVENT -0207	RDSha modificato il punto finale del proxy DB *name*.	Nessuno
modifica della configura zione	RDS- EVENT -0213	RDSha rilevato l'aggiunt a dell'istanza DB e l'ha aggiunta automaticamente al gruppo target del proxy DB *name*.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
modifica della configura zione	RDS- EVENT -0214	RDSha rilevato l'elimina zione di un'istanza DB *name* e lo ha rimosso automaticamente dal gruppo target *name* del proxy DB *name*.	Nessuno
modifica della configura zione	RDS- EVENT -0215	RDS- eliminazione rilevata del cluster DB *name* e lo ha rimosso automaticamente dal gruppo di destinazione *name* del proxy DB *name*.	Nessuno
creazione	RDS- EVENT -0203	RDSproxy DB creato *name*.	Nessuno
creazione	RDS- EVENT -0206	RDSendpoint creato *name* per proxy DB *name*.	Nessuno
eliminazione	RDS- EVENT -0205	RDSproxy DB eliminato *name*.	Nessuno
eliminazione	RDS- EVENT -0208	RDSendpoint eliminato *name* per proxy DB *name*.	Nessuno

Categoria	ID evento RDS	Messaggio	Note
errore	RDS- EVENT -0243	RDSnon è riuscito a fornire la capacità per il proxy *name* perché non ci sono abbastanza indirizzi IP disponibili nelle tue sottoreti : *name*. Per risolvere il problema, assicurati che le sottoreti abbiano il numero minimo di indirizzi IP inutilizzati, come consigliato nella documentazione del RDS proxy.	Per determinare il numero consigliato per la classe di istanza, consulta Pianificazione della capacità degli indirizzi IP.
errore	RDSEVENT- -0275	RDSha limitato alcune connessioni al proxy DB *name*. Il numero di richieste di connessione simultane e dal client al proxy ha superato il limite.	Nessuno

Di seguito è riportato un esempio di evento RDS Proxy in JSON formato. L'evento mostra che ha RDS modificato l'endpoint denominato `my-endpoint` del RDS Proxy denominato`my-rds-proxy`. L'ID dell'evento è RDS - EVENT -0207.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
     "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
```

```
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```

RDSEsempi della riga di comando del proxy

Per vedere come le combinazioni di comandi e SQL istruzioni di connessione interagiscono con RDS Proxy, guarda gli esempi seguenti.

Esempi

- Preserving Connections to a MySQL Database Across a Failover
- Adjusting the max_connections Setting for an Aurora DB Cluster

Example Conservazione delle connessioni a un SQL database My durante un failover

Questo mio SQL esempio dimostra come le connessioni aperte continuino a funzionare durante un failover. come quando un database viene riavviato o diventa non disponibile a causa di un problema. In questo esempio viene utilizzato un proxy denominato `the-proxy` e un cluster di database Aurora con istanze DB `instance-8898` e `instance-9814`. Quando il comando `failover-db-cluster` viene eseguito dalla riga di comando di Linux, l'istanza writer a cui il proxy è connesso cambia in un'istanza database diversa. Puoi vedere che l'istanza database associata al proxy cambia mentre la connessione rimane aperta.

```
$ mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p
Enter password:
...

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
```

```
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ # Initially, instance-9814 is the writer.
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-8898 is the writer.
$ fg
mysql -h the-proxy.proxy-demo.us.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-8898      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-9814 is the writer again.
$ fg
mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)
+---------------+---------------+
| Variable_name | Value         |
+---------------+---------------+
| hostname      | ip-10-1-3-178 |
```

```
+---------------+---------------+
1 row in set (0.02 sec)
```

Example Regolazione dell'impostazione max_connections per un cluster di database Aurora

Questo esempio dimostra come è possibile regolare l'`max_connections`impostazione per un cluster Aurora SQL My DB. A tale scopo, create il vostro gruppo di parametri del cluster DB in base alle impostazioni dei parametri predefinite per i cluster compatibili con My 5.7. SQL Specifichi un valore per l'impostazione `max_connections`, sovrascrivendo la formula che imposta il valore predefinito. Associ il gruppo di parametri del cluster di database al cluster di database.

```
export REGION=us-east-1
export CLUSTER_PARAM_GROUP=rds-proxy-mysql-57-max-connections-demo
export CLUSTER_NAME=rds-proxy-mysql-57

aws rds create-db-parameter-group --region $REGION \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-parameter-group-name $CLUSTER_PARAM_GROUP \
  --description "Aurora MySQL 5.7 cluster parameter group for RDS Proxy demo."

aws rds modify-db-cluster --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP

echo "New cluster param group is assigned to cluster:"
aws rds describe-db-clusters --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --query '*[*].{DBClusterParameterGroup:DBClusterParameterGroup}'

echo "Current value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"

echo -n "Enter number for max_connections setting: "
read answer

aws rds modify-db-cluster-parameter-group --region $REGION --db-cluster-parameter-
group-name $CLUSTER_PARAM_GROUP \
  --parameters "ParameterName=max_connections,ParameterValue=$
$answer,ApplyMethod=immediate"
```

```
echo "Updated value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"
```

Risoluzione dei problemi per RDS Proxy

Di seguito, sono disponibili idee per la risoluzione di alcuni problemi comuni relativi al proxy RDS e informazioni sui CloudWatch registri di RDS Proxy.

Nei log RDS Proxy, ogni voce è preceduta dal nome dell'endpoint proxy associato. Questo nome può essere quello specificato per un endpoint definito dall'utente. In alternativa, può essere il nome speciale dell'endpoint predefinito `default` di un proxy che esegue richieste di lettura/scrittura. Per ulteriori informazioni sugli endpoint proxy, consulta Utilizzo degli endpoint Amazon RDS Proxy.

Argomenti

- Verifica della connettività a un proxy
- Problemi e soluzioni comuni
- Risoluzione dei problemi relativi al proxy RDS con RDS for MySQL
- Risoluzione dei problemi relativi al proxy RDS con RDS per PostgreSQL

Verifica della connettività a un proxy

È possibile utilizzare i seguenti comandi per verificare che tutti i componenti, ad esempio il proxy, il database e le istanze di calcolo presenti nella connessione, possano comunicare tra loro.

Esamina il proxy stesso usando il describe-db-proxiescomando. Esamina anche il gruppo target associato utilizzando il comando describe-db-proxy-target-groups. Verifica che i dettagli delle destinazioni corrispondano al cluster Aurora che intendi associare al proxy. Utilizzare comandi come i seguenti.

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

Per confermare che il proxy è in grado di connettersi al database sottostante, esamina le destinazioni specificate nei gruppi di destinazione utilizzando il describe-db-proxy-targetscomando. Utilizzare un comando come il seguente.

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

L'output del describe-db-proxy-targetscomando include un `TargetHealth` campo. È possibile esaminare i campi `State`, `Reason` e `Description` all'interno di `TargetHealth` per verificare se il proxy può comunicare con l'istanza database sottostante.

- Un valore `State` di `AVAILABLE` indica che il proxy può connettersi all'istanza database.
- Un valore `State` di `UNAVAILABLE` indica un problema di connessione temporaneo o permanente. In questo caso, esaminare i campi `Reason` e `Description`. Ad esempio, se `Reason` ha un valore pari a `PENDING_PROXY_CAPACITY`, provare a connettersi nuovamente dopo che il proxy ha terminato l'operazione di ridimensionamento. Se `Reason` ha un valore di `UNREACHABLE`, `CONNECTION_FAILED` o `AUTH_FAILURE`, utilizzare la spiegazione del campo `Description` per facilitare la diagnosi del problema.
- Il valore del campo `State` potrebbe essere `REGISTERING` per un breve periodo prima di passare a `AVAILABLE` o `UNAVAILABLE`.

Se il seguente comando Netcat (`nc`) ha esito positivo, puoi accedere all'endpoint proxy dall' EC2 istanza o da un altro sistema a cui hai effettuato l'accesso. Questo comando segnala un errore se non ti trovi nello stesso VPC del proxy e del database associato. Potresti essere in grado di accedere direttamente al database senza essere nello stesso VPC. Tuttavia, non puoi accedere al proxy a meno che non ti trovi nello stesso VPC.

```
nc -zx MySQL_proxy_endpoint 3306

nc -zx PostgreSQL_proxy_endpoint 5432
```

Puoi usare i seguenti comandi per assicurarti che l' EC2 istanza abbia le proprietà richieste. In particolare, il VPC dell' EC2 istanza deve essere lo stesso del VPC per il cluster Aurora dell'istanza DB RDS a cui si connette il proxy.

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

Esamina i segreti Secrets Manager utilizzati per il proxy.

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

Assicurati che il `SecretString` campo visualizzato da `get-secret-value` sia codificato come una stringa JSON che include i campi and. `username` `password` Nell'esempio seguente viene illustrato il formato del campo `SecretString`.

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

Problemi e soluzioni comuni

Questa sezione descrive alcuni problemi comuni e potenziali soluzioni quando si utilizza RDS Proxy.

Dopo aver eseguito il comando `aws rds describe-db-proxy-targets` CLI, se la `TargetHealth` descrizione indica`Proxy does not have any registered credentials`, verifica quanto segue:

- Per l'accesso la proxy, l'utente dispone di credenziali registrate.
- Il ruolo IAM per accedere al segreto di Secrets Manager utilizzato dal proxy è valido.

È possibile che si verifichino i seguenti eventi RDS durante la creazione di o la connessione a un proxy DB.

Categoria	ID evento RDS	Descrizione
errore	RDS-EVENT-0243	RDS non è stato in grado di allocare la capacità per il proxy perché non ci sono sufficienti indirizzi IP disponibili nelle sottoreti. Per risolvere il problema, assicurati che le sottoreti abbiano il numero minimo di indirizzi IP non utilizzati. Per determinare

Categoria	ID evento RDS	Descrizione
		il numero consigliato per la classe di istanza, consulta Pianificazione della capacità degli indirizzi IP.
errore	RDS-EVENT-0275	RDS ha limitato alcune connessioni al proxy DB. *name* Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.

È possibile che si verifichino i seguenti problemi durante la creazione di un nuovo proxy o la connessione a un proxy.

Errore	Cause o soluzioni alternative
`403: The security token included in the request is invalid`	Seleziona un ruolo IAM esistente invece di crearne uno nuovo.

Risoluzione dei problemi relativi al proxy RDS con RDS for MySQL

È possibile che si verifichino i seguenti problemi durante la connessione a un proxy MySQL.

Errore	Cause o soluzioni alternative
`ERROR 1040 (HY000): Connections rate limit`	La velocità di richieste di connessione dal client al proxy ha superato il limite.

Errore	Cause o soluzioni alternative
`exceeded (limit_value )`	
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	Il numero di richieste simultanee con autenticazione IAM dal client al proxy ha superato il limite.
`ERROR 1040 (HY000): Number simultaneous connections exceeded (limit_value )`	Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.
`ERROR 1045 (28000): Access denied for user 'DB_USER'@'%' (using password: YES)`	Il segreto Secrets Manager utilizzato dal proxy non corrisponde al nome utente e alla password di un utente di database esistente. Aggiorna le credenziali nel segreto Secrets Manager o assicurati che l'utente del database esista e disponga della stessa password del segreto.
`ERROR 1105 (HY000): Unknown error`	Si è verificato un errore sconosciuto.
`ERROR 1231 (42000): Variable ''character_set_client'' can't be set to the value of value`	Il valore impostato per il parametro `character_set_client` non è valido. Ad esempio, il valore `ucs2` non è valido perché può provocare un arresto anomalo del server MySQL.

Errore	Cause o soluzioni alternative
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	Hai abilitato l'impostazione Richiedi Transport Layer Security nel proxy ma la tua connessione includeva il parametro `ssl-mode=DISABLED` nel client MySQL. Eseguire una delle operazioni seguenti: • Disattivare l'impostazione Richiedi Transport Layer Security per il proxy. • Connettersi al database utilizzando l'impostazione minima di `ssl-mode=REQUIRED` nel client MySQL.
`ERROR 2026 (HY000): SSL connection error: Internal Server` *`Error`*	L'handshake TLS con il proxy non è riuscito. Alcuni possibili motivi includono quanto segue: • SSL è richiesto ma il server non lo supporta. • Si è verificato un errore interno del server. • Si è verificato un handshake non valido.
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	Il proxy è in attesa di acquisire una connessione al database. Alcuni possibili motivi includono quanto segue: • Il proxy non è in grado di stabilire una connessione al database perché sono state raggiunte le connessioni massime. • Il proxy non è in grado di stabilire una connessione al database perché il database non è disponibile.

Risoluzione dei problemi relativi al proxy RDS con RDS per PostgreSQL

È possibile che si verifichino i seguenti problemi durante la connessione a un proxy PostgreSQL.

Errore	Causa	Soluzione
`ERROR 28000: IAM authentication is allowed only with SSL connections.`	L'utente ha tentato di connettersi al database utilizzando l'autenticazione IAM con l'impostazione	L'utente deve connettersi al database utilizzando l'imposta zione minima di `sslmode=r equire` nel client PostgreSQ L. Per ulteriori informazioni,

Errore	Causa	Soluzione
	`sslmode=disable` nel client PostgreSQL.	consulta la documentazione Supporto SSL PostgreSQL.
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the credentials for this role and try again.`	Non c'è un Secrets Manager segreto per questo ruolo.	Aggiungere un Secrets Manager segreto per questo ruolo. Per ulteriori informazioni, consulta Configurazione dell'autenticazione IAM per RDS Proxy.
`ERROR 28000: RDS supports only IAM, MD5, or SCRAM authentication.`	Il client di database utilizzato per connettersi al proxy utilizza un meccanismo di autenticazione non attualmente supportato dal proxy.	Se non utilizzi l'autenticazione IAM, utilizza l'autenticazione tramite password MD5 o SCRAM.
`ERROR 28000: A user name is missing from the connection startup packet. Provide a user name for this connection.`	Il client di database utilizzato per connettersi al proxy non invia un nome utente quando si tenta di stabilire una connessione.	Assicurarsi di definire un nome utente quando si imposta una connessione al proxy utilizzando il client PostgreSQL di propria scelta.
`ERROR 28000: IAM is allowed only with SSL connections.`	Un client ha tentato di connettersi utilizzando l'autenticazione IAM, ma SSL non è stato abilitato.	Abilitare SSL nel client PostgreSQL.

Errore	Causa	Soluzione
`ERROR 28000: This RDS Proxy requires TLS connections.`	L'utente ha abilitato l'impostazione Richiedi Transport Layer Security ma ha tentato di connettersi con `sslmode=disable` nel client PostgreSQL.	Per risolvere questo errore, effettuare una delle seguenti operazioni: • Disattivare l'impostazione Richiedi Transport Layer Security del proxy. • Connettersi al database utilizzando l'impostazione minima di `sslmode=allow` nel client PostgreSQL.
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	Questo errore potrebbe essere dovuto ai seguenti fattori: • Il client ha fornito il nome utente IAM non corretto. • Il client ha fornito un token di autorizzazione IAM non corretto per l'utente • Il client utilizza una policy IAM che non dispone delle autorizzazioni necessarie. • Il client ha fornito un token di autorizzazione IAM scaduto per l'utente.	Per correggere questo errore, effettuare le seguenti operazioni: 1. Verificare che l'utente IAM fornito esista. 2. Verificare che il token di autorizzazione IAM appartenga all'utente IAM fornito. 3. Verificare che la policy IAM disponga di autorizzazioni adeguate per RDS. 4. Verificare la validità del token di autorizzazione IAM utilizzato.
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	La password per questo ruolo non corrisponde al segreto Secrets Manager.	Controlla il segreto per questo ruolo in Secrets Manager per vedere se la password è uguale a quella utilizzata nel client PostgreSQL.

Errore	Causa	Soluzione
`ERROR 28P01: The IAM authentication failed for the role` *`role_name`* `. Check the IAM token for this role and try again.`	Si è verificato un problema con il token IAM utilizzato per l'autenticazione IAM.	Generare un nuovo token di autenticazione e utilizzarlo in una nuova connessione.
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	Il client PostgreSQL utilizzato per connettersi al proxy utilizza un protocollo precedente a 3.0.	Utilizzare un client PostgreSQL più recente che supporti il protocollo di messaggistica 3.0. Se si utilizza la CLI `psql` di PostgreSQL, utilizzare una versione maggiore o uguale a 7.4.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	Il client PostgreSQL utilizzato per connettersi al proxy sta tentando di utilizzare la modalità di replica in streaming, che non è attualmente supportata dal proxy RDS.	Disattivare la modalità di replica in streaming nel client PostgreSQL utilizzato per la connessione.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option` *`option_name`* `.`	Tramite il messaggio di avvio, il client PostgreSQL utilizzato per connettersi al proxy richiede un'opzione che non è attualmente supportata dal proxy RDS.	Disattivare l'opzione visualizzata come non supportata dal messaggio precedente nel client PostgreSQL utilizzato per connettersi.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	Il numero di richieste simultanee con autenticazione IAM dal client al proxy ha superato il limite.	Ridurre la velocità con cui vengono stabilite le connessioni che utilizzano l'autenticazione IAM da un client PostgreSQL.

Errore	Causa	Soluzione
`ERROR 53300: The maximum number of client connections to the proxy exceeded` *`number_value`* `.`	Il numero di richieste di connessione simultanee dal client al proxy ha superato il limite.	Ridurre il numero di connessioni attive dai client PostgreSQL a questo proxy RDS.
`ERROR 53300: Rate of connection to proxy exceeded` *`number_value`* `.`	La velocità di richieste di connessione dal client al proxy ha superato il limite.	Ridurre la velocità con cui vengono stabilite le connessioni da un client PostgreSQL.
`ERROR XX000: Unknown error.`	Si è verificato un errore sconosciuto.	Rivolgiti all' AWS assistenza per esaminare il problema.

Errore	Causa	Soluzione
`ERROR 08000: Timed-out waiting to acquire database connection.`	Il proxy è scaduto in attesa di acquisire una connessione al database entro la durata specificata dall'`ConnectionBorrowTimeout` impostazione. Alcuni possibili motivi includono quanto segue: • Il proxy non può stabilire una connessione al database perché sono state raggiunte le connessioni massime. • Il proxy non può stabilire una connessione al database perché il database non è disponibile o se la connessione al database richiede più tempo di quanto configurato. `ConnectionBorrowTimeout`	Le possibili soluzioni sono le seguenti: • Evita di bloccare le connessioni proxy. Consultare Evitare il blocco di un proxy RDS. • Revisione `ConnectionBorrowTimeout` e `MaxConnectionsPercent` impostazioni. Consultare Considerazioni sulla connessione del proxy RDS. • Verifica la disponibilità degli obiettivi. Consulta `AvailabilityPercentage` in Monitoraggio dei parametri del proxy RDS con Amazon CloudWatch.

Errore	Causa	Soluzione
`ERROR XX000: Request returned an error: database_error .`	La connessione al database stabilita dal proxy ha restituito un errore.	La soluzione dipende dall'errore specifico del database. Un esempio : `Request returned an error: database "your-database-name" does not exist`. Ciò significa che il nome del database specificato non esiste nel server di database oppure che il nome utente utilizzato come nome del database (se non è specificato un nome del database) non esiste nel server.

Risoluzione dei problemi relativi al **`postgres`** database eliminato

Se elimini erroneamente il `postgres` database dall'istanza, devi ripristinare il database per ripristinare la connettività dell'istanza. Esegui i seguenti comandi all'interno dell'istanza DB:

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Utilizzo di RDS Proxy con AWS CloudFormation

Puoi usare RDS Proxy con AWS CloudFormation. Questo ti aiuta a creare gruppi di risorse correlate. Un gruppo di questo tipo può includere un proxy che può connettersi a un cluster di database Aurora appena creato. Il supporto di RDS Proxy in AWS CloudFormation coinvolge due nuovi tipi di registro: `DBProxy` e `DBProxyTargetGroup`.

L'elenco seguente mostra un modello AWS CloudFormation di esempio per RDS Proxy.

```
Resources:
```

```
DBProxy:
  Type: AWS::RDS::DBProxy
  Properties:
    DBProxyName: CanaryProxy
    EngineFamily: MYSQL
    RoleArn:
     Fn::ImportValue: SecretReaderRoleArn
    Auth:
      - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
    VpcSubnetIds:
      Fn::Split: [",", "Fn::ImportValue": SubnetIds]

ProxyTargetGroup:
  Type: AWS::RDS::DBProxyTargetGroup
  Properties:
    DBProxyName: CanaryProxy
    TargetGroupName: default
    DBInstanceIdentifiers:
      - Fn::ImportValue: DBInstanceName
  DependsOn: DBProxy
```

Per ulteriori informazioni sulle risorse di questo esempio, consulta `DBProxy` e `DBProxyTargetGroup`.

Per ulteriori informazioni sulle risorse che è possibile creare utilizzando AWS CloudFormation, consulta RDS resource type reference.

Utilizzo del Server proxy per RDS con i database globali Aurora

Un database globale Aurora è un singolo database che comprende più Regioni AWS, consentendo letture globali a bassa latenza e ripristino di emergenza da interruzioni dell'attività a livello regionale. Fornisce tolleranza ai guasti incorporata per la distribuzione perché l'istanza database non si basa su una singola Regione AWS, ma su più Regioni e zone di disponibilità diverse. Per ulteriori informazioni, consulta Utilizzo del database globale Amazon Aurora.

È possibile utilizzare Server proxy per RDS con qualsiasi cluster database in un database globale Aurora. Prima di iniziare a utilizzare queste funzionalità, assicurati di verificare le seguenti informazioni.

> **Important**
>
> Se il cluster database fa parte di un database globale con l'inoltro di scrittura attivato, riduci il valore `MaxConnectionsPercent` del proxy con la quota assegnata all'inoltro di scrittura. La quota di inoltro di scrittura è impostata nel parametro `aurora_fwd_writer_max_connections_pct` del cluster database. Per informazioni sull'inoltro di scrittura, consulta Utilizzo dell'inoltro di scrittura in un database globale Amazon Aurora.

Limitazioni di Server proxy per RDS con i database globali

Quando il cluster database Aurora ha l'inoltro di scrittura attivato, Server proxy per RDS non supporta il valore `SESSION` per la variabile `aurora_replica_read_consistency`, la cui impostazione può causare un comportamento imprevisto.

Come funzionano gli endpoint Server proxy per RDS con i database globali

Se comprendi come gli endpoint Server proxy per RDS funzionano con i database globali, puoi gestire meglio le tue applicazioni che utilizzano i database Aurora con entrambe queste funzionalità.

Per un proxy con il cluster primario di un database globale come destinazione registrata, gli endpoint proxy funzionano allo stesso modo che per qualsiasi cluster database Aurora. Gli endpoint di lettura/scrittura del proxy inviano tutte le richieste all'istanza di scrittura del cluster. Gli endpoint di sola lettura del proxy inviano tutte le richieste alle istanze di lettura. Se un'istanza di lettura non è disponibile mentre la connessione è aperta, Server proxy per RDS reindirizza le query successive sulla connessione a un'altra istanza di lettura. Per un proxy con un cluster secondario come destinazione registrata, le richieste inviate agli endpoint di sola lettura del proxy vengono inviate anche alle istanze di lettura. Poiché il cluster non ha istanze di scrittura, le richieste inviate agli endpoint di lettura/scrittura non riescono e viene restituito l'errore "`The target group doesn't have any associated read/write instances`".

Le operazioni di failover e switchover globale del database prevedono entrambe un cambio di ruolo tra il cluster database primario e uno dei cluster database secondari. Quando il cluster secondario selezionato diventa il nuovo cluster primario, una delle istanze di lettura viene promossa a istanza di scrittura. Questa istanza database è ora la nuova istanza di scrittura del cluster globale. Assicurati di reindirizzare le operazioni di scrittura dell'applicazione all'endpoint di lettura/scrittura appropriato

del proxy associato al nuovo cluster primario. Questo endpoint del proxy può essere l'endpoint predefinito o un endpoint di lettura/scrittura personalizzato.

Server proxy per RDS mette in coda tutte le richieste tramite endpoint di lettura/scrittura e le invia all'istanza di scrittura del nuovo cluster primario non appena è disponibile. Ciò avviene indipendentemente dal completamento dell'operazione di failover o switchover. Durante il failover o lo switchover, l'endpoint predefinito del proxy per il vecchio cluster primario accetta ancora le operazioni di scrittura. Tuttavia, non appena quel cluster diventa secondario, tutte le operazioni di scrittura restituiscono esito negativo. Per informazioni su come e quando eseguire specifiche operazioni di failover o switchover globali, consulta i seguenti argomenti:

- Switchover globale del database: Esecuzione di switchover per database globali Amazon Aurora
- Failover globale del database: Ripristino di un database globale Amazon Aurora da un'interruzione non pianificata

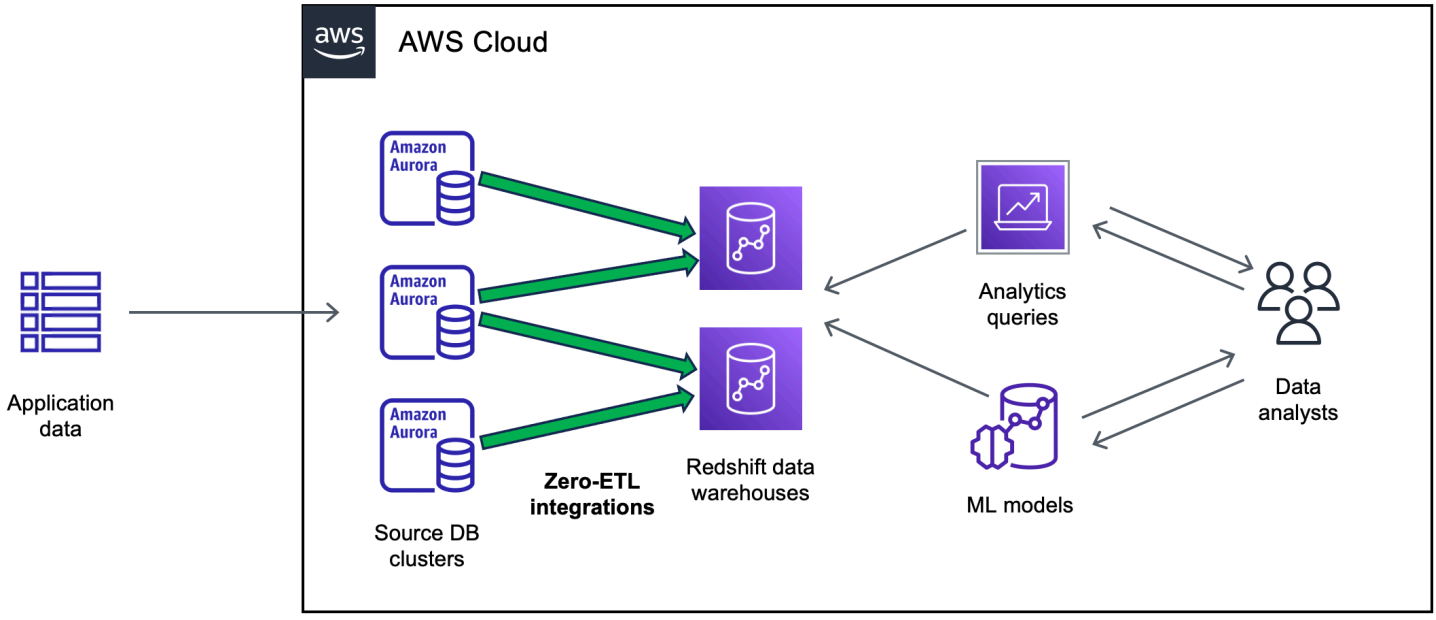
Integrazioni Zero-ETL di Aurora con Amazon Redshift

L'integrazione Zero-ETL di Aurora con Amazon Redshift consente di eseguire operazioni di analisi e machine learning (ML) quasi in tempo reale utilizzando Amazon Redshift su petabyte di dati transazionali di Aurora. È una soluzione completamente gestita per rendere disponibili i dati transazionali in Amazon Redshift dopo averli scritti su un , un cluster Aurora DB. L'estrazione, la trasformazione e il caricamento (ETL) è il processo di combinazione di dati provenienti da più fonti in un ampio data warehouse centrale.

Un'integrazione zero-ETL rende i dati del cluster Aurora DB del disponibili in Amazon Redshift quasi in tempo reale. Una volta che i dati sono in Amazon Redshift, puoi potenziare i tuoi carichi di lavoro di analisi, ML e intelligenza artificiale utilizzando le funzionalità integrate di Amazon Redshift, come apprendimento automatico, viste materializzate, condivisione dei dati, accesso federato a più data store e data lake e integrazioni con Amazon AI e altro. SageMaker QuickSight Servizi AWS

Per creare un'integrazione zero-ETL, specifichi un , un cluster Aurora DB come origine e un data warehouse Amazon Redshift come destinazione. L'integrazione replica i dati dal database di origine nel data warehouse di destinazione.

Il diagramma seguente illustra questa funzionalità.



L'integrazione monitora lo stato della pipeline dei dati ed esegue il ripristino in caso di problemi quando possibile. Puoi creare integrazioni da più database DB) in un unico spazio dei nomi Amazon Redshift, che ti consente di ricavare informazioni su più applicazioni.

Per informazioni sui prezzi per le integrazioni zero-ETL, consulta i prezzi di Amazon RDS, Aurora e i prezzi di Amazon Redshift.

Argomenti

- Vantaggi
- Concetti chiave
- Limitazioni
- Quote
- Regioni supportate
- Guida introduttiva alle integrazioni Zero-ETL di Aurora con Amazon Redshift
- Creazione di integrazioni Zero-ETL di Aurora con Amazon Redshift
- Filtraggio dei dati per le integrazioni zero-ETL di con Amazon Redshift
- Aggiungere dati a un cluster Aurora DB) e interrogarli in Amazon Redshift
- Visualizzazione e monitoraggio delle ETL integrazioni zero di Aurora con Amazon Redshift
- Modifica delle integrazioni zero di Aurora con Amazon ETL Redshift
- Eliminazione delle integrazioni zero di Aurora con Amazon ETL Redshift
- Risoluzione dei problemi delle ETL integrazioni zero di Aurora con Amazon Redshift

Vantaggi

Le integrazioni Zero-ETL di Aurora con Amazon Redshift offrono i seguenti vantaggi:

- Ti consentono di ottenere approfondimenti di tipo olistico da più origini dati.
- Eliminano la necessità di creare e gestire pipeline dei dati complesse che eseguono operazioni di estrazione, trasformazione e caricamento (ETL). Le integrazioni Zero-ETL forniscono e gestiscono le pipeline per te, eliminando le sfide legate alla loro creazione e gestione.
- Ti consentono di ridurre il carico e i costi operativi e di concentrarti sul miglioramento delle applicazioni.
- Consenti di sfruttare le funzionalità di analisi e ML di Amazon Redshift per ricavare informazioni dettagliate da dati transazionali e di altro tipo, per rispondere efficacemente a eventi critici e urgenti.

Concetti chiave

Per iniziare a utilizzare le integrazioni Zero-ETL, tieni presente i seguenti concetti:

Integrazione

Una pipeline di dati completamente gestita che replica automaticamente i dati e gli schemi transazionali da un cluster RDS a un data warehouse Amazon Redshift.

Il cluster Aurora DB da cui vengono replicati i dati. È possibile specificare un cluster DB che utilizza istanze DB o Aurora Serverless v2 istanze DB assegnate come origine.

Data warehouse di destinazione

Si tratta del data warehouse di Amazon Redshift in cui viene eseguita la replica dei dati. Esistono due tipi di data warehouse: un data warehouse con cluster con provisioning e un data warehouse serverless. Un data warehouse con cluster con provisioning è costituito da un insieme di risorse di calcolo denominate nodi, strutturate in un gruppo denominato cluster. Un data warehouse serverless è composto da un gruppo di lavoro che archivia le risorse di calcolo e da un spazio dei nomi che ospita gli oggetti e gli utenti del database. Entrambi i data warehouse utilizzano un motore Amazon Redshift e contengono uno o più database.

I cluster DB di più di origine possono scrivere sulla stessa destinazione.

Per ulteriori informazioni sui nodi principali e sui nodi di calcolo, consulta Architettura del sistema di data warehouse nella Guida per sviluppatori di database di Amazon Redshift.

Limitazioni

Le seguenti limitazioni si applicano alle integrazioni Zero-ETL di Aurora con Amazon Redshift.

Argomenti

- Limitazioni generali
- Aurora MySQL
- Limitazioni di Aurora PostgreSQL
- Limitazioni di Amazon Redshift

Limitazioni generali

- Il cluster DB del di origine deve trovarsi nella stessa regione del data warehouse Amazon Redshift di destinazione.
- Non puoi rinominare un cluster DB di o una delle sue istanze se dispone di integrazioni esistenti.
- Non è possibile creare più integrazioni tra gli stessi database di origine e di destinazione.
- Non è possibile eliminare un cluster DB di integrazioni esistenti. Devi prima eliminare tutte le integrazioni associate.
-
- Se il cluster di è all'origine di una distribuzione blu/verde, gli ambienti blu e verde non possono avere integrazioni zero-ETL esistenti durante lo switchover. Occorre eliminare l'integrazione, eseguire lo switchover e poi ricrearla.
- Un cluster DB deve contenere almeno un'istanza DB per essere la fonte di un'integrazione.
- Non è possibile creare un'integrazione per un cluster DB di origine che è un clone tra più account, come quelli condivisi utilizzando AWS Resource Access Manager ()AWS RAM.
- Se il cluster di origine è il cluster database primario in un database globale Aurora e esegue il failover su uno dei relativi cluster secondari, l'integrazione diventa inattiva. È necessario eliminare e ricreare l'integrazione.
- Non è possibile creare un'integrazione per un database di origine in cui viene creata attivamente un'altra integrazione.
- Durante la fase iniziale della creazione di un'integrazione o quando una tabella viene risincronizzata, il seeding dei dati dall'origine alla destinazione può richiedere 20-25 minuti o più, a seconda delle dimensioni del database di origine. Questo ritardo può portare a un aumento del ritardo di replica.
- Alcuni tipi di dati non sono supportati. Per ulteriori informazioni, consulta the section called “Differenze dei tipi di dati”.
- Le tabelle di sistema, le tabelle temporanee e le viste non vengono replicate su Amazon Redshift.
- `ALTER TABLE`Le operazioni di partizione provocano la risincronizzazione della tabella per ricaricare i dati da Aurora ad Amazon Redshift. La tabella non sarà disponibile per l'interrogazione durante la risincronizzazione. Per ulteriori informazioni, consulta the section called “Una o più tabelle Amazon Redshift richiedono una risincronizzazione”.

Aurora MySQL

- Il cluster DB di origine deve eseguire una versione supportata di Aurora MySQL. Per un elenco delle versioni supportate, consulta the section called “Integrazioni Zero-ETL ”.
- Le integrazioni Zero-ETL si basano sui log binari MySQL (binlog) per acquisire le modifiche continue dei dati. Non utilizzare il filtraggio dei dati basato su binlog, poiché può causare incongruenze di dati tra il database di origine e quello di destinazione.
- Le integrazioni Zero-ETL sono supportate solo per i database configurati per l'utilizzo del motore di storage InnoDB.
- I riferimenti a chiavi esterne con aggiornamenti di tabella predefiniti non sono supportati. In particolare, `ON DELETE` le `ON UPDATE` regole non sono supportate con `CASCADESET NULL`, e `SET DEFAULT` le azioni. Se si tenta di creare o aggiornare una tabella con tali riferimenti a un'altra tabella, la tabella entrerà in uno stato di errore.
- Le transazioni XA eseguite sul cluster DB di origine fanno sì che l'integrazione entri in uno stato di. `Syncing`

Limitazioni di Aurora PostgreSQL

- Il cluster DB di origine deve eseguire una versione supportata di Aurora PostgreSQL. Per un elenco delle versioni supportate, consulta the section called “Integrazioni Zero-ETL ”.
- Se si seleziona un cluster DB di origine Aurora PostgreSQL, è necessario specificare almeno un modello di filtro dei dati. Come minimo, il modello deve includere un singolo database (*`database-name`*`.*.*`) per la replica su Amazon Redshift. Per ulteriori informazioni, consulta the section called “Filtraggio dei dati per integrazioni zero-ETL”.
- Tutti i database creati all'interno del cluster Aurora PostgreSQL DB di origine devono utilizzare la codifica UTF-8.
- Se esegui transazioni di partizionamento dichiarativo sul cluster DB di origine, tutte le tabelle interessate entrano in uno stato di errore e non sono più accessibili in Amazon Redshift.
- Le transazioni in due fasi non sono supportate.
- Se si eliminano tutte le istanze DB da un cluster DB che è l'origine di un'integrazione e quindi si aggiunge nuovamente un'istanza DB, la replica si interrompe tra i cluster di origine e quelli di destinazione.
- Il cluster DB di origine non può utilizzare Aurora Limitless Database.

Limitazioni di Amazon Redshift

Per un elenco delle limitazioni di Amazon Redshift relative alle integrazioni Zero-ETL, consulta Considerazioni sull'utilizzo delle integrazioni Zero-ETL con Amazon Redshift nella Amazon Redshift Management Guide.

Quote

Sul tuo account sono disponibili le seguenti quote relative alle integrazioni Zero-ETL di Aurora con Amazon Redshift. Salvo dove diversamente specificato, ogni quota fa riferimento a una Regione specifica.

Nome	Predefinito	Descrizione
Integrazioni	100	Numero totale di integrazioni all'interno di un Account AWS.
Integrazioni per data warehouse di destinazione	50	Numero di integrazioni che inviano dati a un unico data warehouse Amazon Redshift di destinazione.
Integrazioni per cluster di origine	5	

Inoltre, Amazon Redshift pone determinati limiti al numero di tabelle consentite in ogni istanza database o nodo del cluster. Per ulteriori informazioni, consulta Quote e limiti in Amazon Redshift nella Guida alla gestione di Amazon Redshift.

Regioni supportate

Le integrazioni Zero-ETL di Aurora con Amazon Redshift sono disponibili in un sottoinsieme di. Regioni AWS Per un elenco delle regioni supportate, consultare the section called “Integrazioni Zero-ETL ”.

Guida introduttiva alle integrazioni Zero-ETL di Aurora con Amazon Redshift

Prima di creare un'integrazione zero-ETL con Amazon Redshift, configura il DB e il data warehouse Amazon Redshift con i parametri e le autorizzazioni richiesti. Durante la configurazione, dovrai completare i seguenti passaggi:

1. Creazione di un gruppo di parametri per il cluster DB personalizzato..
2. Crea cluster DB di database di origine.
3. Creazione di un data warehouse Amazon Redshift di destinazione.

Dopo aver completato questi passaggi, passa alla sezione the section called “Creazione di integrazioni Zero-ETL”.

Puoi usare il AWS SDKs per automatizzare il processo di configurazione per te. Per ulteriori informazioni, consulta the section called “Configurare un'integrazione utilizzando AWS SDKs”.

> **Tip**
>
> Puoi fare in modo che RDS completi questi passaggi di configurazione al posto tuo mentre crei l'integrazione, anziché eseguirli manualmente. Per iniziare subito a creare un'integrazione, consulta the section called “Creazione di integrazioni Zero-ETL”.

Fase 1: creazione di un gruppo di parametri del cluster DB personalizzato

Le integrazioni Aurora zero-ETL con Amazon Redshift richiedono valori specifici per i parametri del cluster DB che controllano la replica. In particolare, Aurora MySQL richiede binlog () avanzato e Aurora PostgreSQL richiede una replica logica avanzata ()*aurora_enhanced_binlog*. `aurora.enhanced_logical_replication`

Per configurare la registrazione binaria o la replica logica, è necessario innanzitutto creare un gruppo di parametri del cluster DB personalizzato e quindi associarlo al cluster DB di origine.

Aurora MySQL (famiglia aurora-mysql8.0):

- `aurora_enhanced_binlog=1`
- `binlog_backup=0`

- `binlog_format=ROW`
- `binlog_replication_globaldb=0`
- `binlog_row_image=full`
- `binlog_row_metadata=full`

Inoltre, assicurati che il parametro `binlog_transaction_compression` non sia impostato su `ON` e che il parametro `binlog_row_value_options` non sia impostato su `PARTIAL_JSON`.

Per ulteriori informazioni su Aurora MySQL Enhanced binlog, consulta. the section called "Configurazione del file di log binario avanzato"

Aurora PostgreSQL (famiglia aurora-postgresql16):

- `rds.logical_replication=1`
- `aurora.enhanced_logical_replication=1`
- `aurora.logical_replication_backup=0`
- `aurora.logical_replication_globaldb=0`

L'abilitazione della replica logica avanzata (`aurora.enhanced_logical_replication`) scriverà sempre tutti i valori delle colonne nel log write ahead (WAL) anche se non è abilitata. `REPLICA IDENTITY FULL` Ciò potrebbe aumentare gli IOPS per il cluster DB di origine.

> **Important**
>
> Se abiliti o disabiliti il parametro del cluster `aurora.enhanced_logical_replication` DB, l'istanza DB principale invalida tutti gli slot di replica logica. Ciò interrompe la replica dall'origine alla destinazione ed è necessario ricreare gli slot di replica sull'istanza DB principale. Per evitare interruzioni, mantenete costante lo stato dei parametri durante la replica.

Passaggio 2: selezionare o creare un cluster DB del di origine

Dopo aver creato un gruppo di parametri del cluster DB personalizzato, scegli o crea un cluster . Questo cluster di sarà l'origine della replica dei dati su Amazon Redshift. È possibile specificare un cluster DB che utilizza istanze DB assegnate oppure Aurora Serverless v2 Istanze DB come origine.

Per istruzioni su come creare un cluster di database di istanze DB , consulta the section called “Creazione di un cluster di database”o. the section called “Creare un Aurora Serverless v2 cluster di database”

Il database deve eseguire una versione del motore DB supportata. Per un elenco delle versioni supportate, consulta the section called “Integrazioni Zero-ETL ”.

Quando create il database, in Configurazione aggiuntiva, modificate il gruppo di parametri predefinito del cluster DB con il gruppo di parametri personalizzato creato nel passaggio precedente.

> **Note**
>
> Se si associa il gruppo di parametri al cluster DB del dopo che il cluster di è già stato creato, è necessario riavviare l'istanza DB principale del nel cluster per applicare le modifiche prima di poter creare un'integrazione zero-ETL. Per istruzioni, consulta the section called “Riavvio di un cluster o di un'istanza Aurora DB”.

Fase 3: creazione di un data warehouse Amazon Redshift di destinazione

Dopo aver creato il cluster DB del di origine, devi creare e configurare un data warehouse di destinazione in Amazon Redshift. Il data warehouse deve soddisfare i seguenti requisiti:

- Utilizzando un tipo di RA3 nodo con almeno due nodi o Redshift Serverless.
- Deve essere crittografato (se si utilizza un cluster con provisioning). Per ulteriori informazioni, consulta Crittografia dei database di Amazon Redshift.

Per istruzioni su come creare un data warehouse, consulta Creazione di un cluster per i cluster con provisioning o Creazione di un gruppo di lavoro con uno spazio dei nomi per Redshift Serverless.

Abilitazione della distinzione tra maiuscole e minuscole nel data warehouse

Affinché l'integrazione venga eseguita correttamente, il parametro di distinzione tra maiuscole e minuscole (`enable_case_sensitive_identifier`) deve essere abilitato per il data warehouse. Per impostazione predefinita, la distinzione tra maiuscole e minuscole è disabilitata su tutti i cluster con provisioning e sui gruppi di lavoro Redshift serverless.

Per abilitare la distinzione tra maiuscole e minuscole, esegui i seguenti passaggi a seconda del tipo di data warehouse:

- Cluster con provisioning: per abilitare la distinzione tra maiuscole e minuscole su un cluster con provisioning, crea un gruppo di parametri personalizzato con il parametro `enable_case_sensitive_identifier` abilitato. Poi, associa il gruppo di parametri al cluster. Per istruzioni, consulta Gestione di gruppi di parametri mediante la console o Configurazione dei valori di parametro mediante AWS CLI.

  > **Note**
  >
  > Ricordati di riavviare il cluster dopo aver associato il gruppo di parametri personalizzati.

- Gruppo di lavoro serverless: per abilitare la distinzione tra maiuscole e minuscole su un gruppo di lavoro SRedshift Serverless, è necessario utilizzare la AWS CLI. La console Amazon Redshift attualmente non supporta la modifica dei valori dei parametri Redshift Serverless. Invia la seguente richiesta di aggiornamento al gruppo di lavoro:

  ```
  aws redshift-serverless update-workgroup \
    --workgroup-name target-workgroup \
    --config-parameters
   parameterKey=enable_case_sensitive_identifier,parameterValue=true
  ```

  Non è necessario riavviare un gruppo di lavoro dopo aver modificato i valori dei parametri.

Configura l'autorizzazione per il data warehouse

Dopo aver creato un data warehouse, è necessario configurare il di origine Aurora DB cluster come fonte di integrazione autorizzata. Per istruzioni, consulta Configurazione dell'autorizzazione per il data warehouse Amazon Redshift.

Configurare un'integrazione utilizzando AWS SDKs

Invece di configurare ogni risorsa manualmente, puoi eseguire il seguente script Python per configurare automaticamente le risorse richieste. L'esempio di codice lo utilizza AWS SDK per Python (Boto3)per creare un cluster Amazon Aurora DB di origine, un', ciascuno con i valori dei parametri richiesti. Attende quindi che i database siano disponibili prima di creare un'integrazione zero-ETL tra di essi. È possibile commentare diverse funzioni a seconda delle risorse che è necessario configurare.

Per installare le dipendenze richieste, eseguire i seguenti comandi:

```
pip install boto3
pip install time
```

All'interno dello script, modificate facoltativamente i nomi dei gruppi di origine, destinazione e parametri. La funzione finale crea un'integrazione che `my-integration` prende il nome dall'impostazione delle risorse.

Esempio di codice Python

Aurora MySQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora MySQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-mysql8.0',
        Description='For Aurora MySQL binary logging'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
```

```
            {
                'ParameterName': 'aurora_enhanced_binlog',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_format',
                'ParameterValue': 'ROW',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_image',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_metadata',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-mysql',
        EngineVersion='8.0.mysql_aurora.3.05.2',
        DatabaseName='myauroradb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
```

```
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-mysql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora MySQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
```

```
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')
```

```
    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
```

```
    main()
```

Aurora PostgreSQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora PostgreSQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-postgresql16',
        Description='For Aurora PostgreSQL logical replication'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'rds.logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.enhanced_logical_replication',
```

```
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-postgresql',
        EngineVersion='16.4.aurora-postgresql',
        DatabaseName='mypostgresdb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-postgresql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
```

```
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora PostgreSQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
```

```
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']
```

```
    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Passaggi successivi

Con un di origine, un cluster Aurora DB e un data warehouse di destinazione Amazon Redshift, ora puoi creare un'integrazione zero-ETL e replicare i dati. Per istruzioni, consulta the section called “Creazione di integrazioni Zero-ETL”.

Creazione di integrazioni Zero-ETL di Aurora con Amazon Redshift

Quando crei un'integrazione Zero-ETL di Aurora, specifichi il di origine, il cluster Aurora DB e il data warehouse Amazon Redshift di destinazione. Puoi anche personalizzare le impostazioni di crittografia e aggiungere tag. Aurora crea un'integrazione tra il cluster DB del database di e la sua destinazione.

Una volta che l'integrazione è attiva, tutti i dati inseriti nel cluster DB del di origine verranno replicati nel target Amazon Redshift configurato.

Argomenti

- Prerequisiti
- Autorizzazioni richieste
- Creazione di integrazioni Zero-ETL
- Crittografia delle integrazioni con una chiave gestita dal cliente
- Passaggi successivi

Prerequisiti

Prima di creare un'integrazione zero-ETL, devi creare un cluster DB di di origine e un data warehouse Amazon Redshift di destinazione. È inoltre necessario consentire la replica nel data warehouse aggiungendo il cluster DB del come fonte di integrazione autorizzata.

Per istruzioni su come completare ciascuno di questi passaggi, consulta the section called “Guida introduttiva alle integrazioni Zero-ETL”.

Autorizzazioni richieste

Per creare un'integrazione Zero-ETL, occorrono alcune autorizzazioni IAM. Come requisito minimo, dovrai disporre delle autorizzazioni per eseguire le seguenti operazioni:

- Crea integrazioni zero-ETL per il cluster Aurora DB del RDS di origine.
- Visualizzazione ed eliminazione di tutte le integrazioni Zero-ETL.
- Creazione di integrazioni in entrata nel data warehouse di destinazione. Non hai bisogno di questa autorizzazione se lo stesso account possiede il data warehouse Amazon Redshift e questo account è il principale autorizzato di tale data warehouse. Per informazioni sull'aggiunta di principali autorizzati, consulta Configurazione dell'autorizzazione per il data warehouse Amazon Redshift.

La seguente policy di esempio mostra le autorizzazioni con privilegi minimi richieste per creare e gestire le integrazioni. Potresti non aver bisogno di queste autorizzazioni esatte se il tuo utente o ruolo dispone di autorizzazioni più ampie, come una policy gestita. `AdministratorAccess`

> **Note**
>
> Redshift Amazon Resource Names (ARNs) hanno il seguente formato. Nota l'uso di una barra (`/`) anziché dei due punti (`:`) prima dell'UUID dello spazio dei nomi serverless.
>
> - Cluster con provisioning: `arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid`
> - Serverless: `arn:aws:redshift-serverless:{region}:{account-id}:namespace/namespace-uuid`

Policy di esempio

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rds:CreateIntegration"
        ],
        "Resource": [
            "arn:aws:rds:{region}:{account-id}:cluster:source-db",
            "arn:aws:rds:{region}:{account-id}:integration:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "rds:DescribeIntegrations"
        ],
        "Resource": ["*"]
    },
    {
        "Effect": "Allow",
        "Action": [
            "rds:DeleteIntegration",
            "rds:ModifyIntegration"
        ],
        "Resource": [
            "arn:aws:rds:{region}:{account-id}:integration:*"
        ]
    },
```

```
    {
        "Effect": "Allow",
        "Action": [
            "redshift:CreateInboundIntegration"
        ],
        "Resource": [
            "arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid"
        ]
    }]
}
```

Scelta di un data warehouse di destinazione in un account diverso

Se prevedi di specificare un data warehouse Amazon Redshift di destinazione che si trova in un altro Account AWS, devi creare un ruolo che consenta agli utenti dell'account corrente di accedere alle risorse nell'account di destinazione. Per ulteriori informazioni, consulta Fornire l'accesso a un utente IAM in un altro utente Account AWS di tua proprietà.

Il ruolo deve disporre delle seguenti autorizzazioni, che consentono all'utente di visualizzare i cluster Amazon Redshift con provisioning e gli spazi dei nomi Redshift Serverless disponibili nell'account di destinazione.

Autorizzazioni richieste e policy di attendibilità

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Il ruolo deve avere la seguente policy di attendibilità, che specifica l'ID dell'account di destinazione.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::{external-account-id}:root"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Per istruzioni sulla creazione del ruolo, consulta Creazione di un ruolo utilizzando policy di attendibilità personalizzate.

Creazione di integrazioni Zero-ETL

Puoi creare un'integrazione zero-ETL utilizzando l'API AWS Management Console AWS CLI, the o RDS.

Console RDS

Creazione di un'integrazione Zero-ETL

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione a sinistra, scegli Interfacce di rete.
3. Scegli Crea un'integrazione Zero-ETL.
4. In Identificatore dell'integrazione, inserisci un nome per l'integrazione. Il nome può contenere fino a 63 caratteri alfanumerici e può includere trattini.
5. Scegli Next (Successivo).
6. Per Origine, seleziona il cluster Aurora DB del da cui provengono i dati.

> **Note**
>
> RDS avvisa l'utente se i parametri del cluster DB non sono configurati correttamente. Se ricevi questo messaggio, puoi scegliere Correggi per me o configurarli manualmente.

Per istruzioni su come correggerli manualmente, consulta the section called “Fase 1: creazione di un gruppo di parametri del cluster DB personalizzato”.
La modifica dei parametri del cluster DB richiede un riavvio.

7. (Facoltativo) Seleziona Personalizza le opzioni di filtraggio dei dati e aggiungi filtri di dati alla tua integrazione. Puoi utilizzare i filtri di dati per definire l'ambito della replica nel data warehouse di destinazione. Per ulteriori informazioni, consulta the section called “Filtraggio dei dati per integrazioni zero-ETL”.
8. Una volta configurato correttamente il cluster DB del di origine, scegli Avanti.
9. Per Destinazione, esegui queste operazioni:

   1. (Facoltativo) Per utilizzare un account diverso Account AWS per il target Amazon Redshift, scegli Specificare un account diverso. Quindi, inserisci l'ARN di un ruolo IAM con le autorizzazioni per visualizzare i data warehouse. Per istruzioni su come creare il ruolo IAM, consulta the section called “Scelta di un data warehouse di destinazione in un account diverso”.
   2. Per il data warehouse di Amazon Redshift, seleziona la destinazione per i dati replicati dal cluster DB del di origine. Puoi scegliere un cluster Amazon Redshift con provisioning o uno spazio dei nomi Redshift Serverless come destinazione.

   > **Note**
   >
   > RDS ti avvisa se le policy relative alle risorse o le impostazioni di distinzione tra maiuscole e minuscole per il data warehouse specificato non sono configurate correttamente. Se ricevi questo messaggio, puoi scegliere Correggi per me o configurarli manualmente. Per istruzioni su come correggerli manualmente, consulta Attivazione della distinzione tra maiuscole e minuscole per il data warehouse e Configurazione dell'autorizzazione per il data warehouse nella Guida alla gestione di Amazon Redshift.
   > La modifica della distinzione tra maiuscole e minuscole per un cluster Redshift con provisioning richiede un riavvio. Prima di poter creare l'integrazione, è necessario completare il riavvio e applicare correttamente il nuovo valore del parametro al cluster.
   > Se l'origine e la destinazione selezionate si trovano in Account AWS diversi, Amazon RDS non può correggere queste impostazioni per te. Devi accedere all'altro account e correggerle manualmente in Amazon Redshift.

10. Una volta configurato correttamente il data warehouse di destinazione, scegli Avanti.

11. (Facoltativo) In Tag, aggiungi uno o più tag all'integrazione. Per ulteriori informazioni, consulta the section called “Taggare RDSAurora e risorse”.
12. In Crittografia, specifica come eseguire la crittografia dell'integrazione. Per impostazione predefinita, RDS crittografa tutte le integrazioni con un. Chiave di proprietà di AWS Per scegliere invece una chiave gestita dal cliente, abilita Personalizza le impostazioni di crittografia e scegli una chiave KMS da utilizzare per la crittografia. Per ulteriori informazioni, consulta the section called “Crittografia delle risorse Amazon Aurora”.

    Aggiungi un contesto di crittografia (facoltativo). Per ulteriori informazioni, consultare Contesto della crittografia nella Guida per gli sviluppatori di AWS Key Management Service .

    > **Note**
    >
    > Amazon RDS aggiunge le seguenti coppie di contesti di crittografia oltre a quelle che aggiungi:
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`
    >
    > Ciò riduce il numero complessivo di coppie che puoi aggiungere da 8 a 6 e contribuisce al limite complessivo di caratteri del vincolo di concessione. Per ulteriori informazioni, consulta Using grant constraints nella Developer Guide.AWS Key Management Service
13. Scegli Next (Successivo).
14. Rivedi le impostazioni dell'integrazione e scegli Crea un'integrazione Zero-ETL.

    Se la creazione ha esito negativo, consulta the section called “Non riesco a creare un'integrazione zero ETL” per la procedura di risoluzione dei problemi.

Lo stato dell'integrazione è `Creating` mentre l'integrazione è in fase di creazione, mentre lo stato del data warehouse Amazon Redshift di destinazione è `Modifying`. Durante questo periodo, non puoi eseguire query sul data warehouse o apportare modifiche alla configurazione.

Quando la creazione dell'integrazione viene completata correttamente, lo stato dell'integrazione e del data warehouse Amazon Redshift di destinazione cambia in `Active`.

AWS CLI

Per creare un'integrazione zero-ETL utilizzando il AWS CLI, utilizzate il comando create-integration con le seguenti opzioni:

- `--integration-name`: specifica un nome per l'integrazione.
- `--source-arn`— Specificare l'ARN del cluster Aurora DB che sarà l'origine dell'integrazione.
- `--target-arn`: specifica l'ARN del data warehouse di Amazon Redshift che sarà la destinazione dell'integrazione.

Example

In Linux, macOS, oppure Unix:

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

In Windows:

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

API RDS

Per creare un'integrazione Zero-ETL con l'API Amazon RDS, utilizza l'operazione `CreateIntegration` con i seguenti parametri:

- `IntegrationName`: specifica un nome per l'integrazione.
- `SourceArn`— Specificare l'ARN del cluster Aurora DB che sarà l'origine dell'integrazione.
- `TargetArn`: specifica l'ARN del data warehouse di Amazon Redshift che sarà la destinazione dell'integrazione.

Crittografia delle integrazioni con una chiave gestita dal cliente

Se specifichi una chiave KMS personalizzata anziché una Chiave di proprietà di AWS quando crei un'integrazione, la policy chiave deve fornire al servizio Amazon Redshift l'accesso principale all'`CreateGrant`azione. Inoltre, deve consentire all'utente corrente di eseguire le azioni `DescribeKey` e`CreateGrant`.

La seguente politica di esempio mostra come fornire le autorizzazioni richieste nella politica chiave. Include chiavi contestuali per ridurre ulteriormente l'ambito delle autorizzazioni.

Esempio di politica chiave

```
{
    "Version": "2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allows the Redshift service principal to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
```

```
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current uer or role to retrieve information about a KMS
 key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::{account-ID}:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Per ulteriori informazioni, consulta Creazione di una policy delle chiavi nella Guida per gli sviluppatori di AWS Key Management Service .

Passaggi successivi

Dopo aver creato correttamente un'integrazione Zero-ETL, devi creare un database di destinazione all'interno del cluster o del gruppo di lavoro Amazon Redshift di destinazione. Quindi, puoi iniziare ad aggiungere dati al cluster Aurora DB del di origine e interrogarli in Amazon Redshift. Per istruzioni, consulta Creazione di database di destinazione in Amazon Redshift.

Filtraggio dei dati per le integrazioni zero-ETL di con Amazon Redshift

Le integrazioni Zero-ETL di Aurora supportano il filtraggio dei dati, che consente di controllare quali dati vengono replicati dal database Amazon Redshift di destinazione. Invece di replicare l'intero database, puoi applicare uno o più filtri per includere o escludere selettivamente tabelle specifiche. Ciò consente di ottimizzare le prestazioni di archiviazione e di interrogazione assicurando che vengano trasferiti solo i dati pertinenti. Attualmente, il filtraggio è limitato ai livelli di database e tabella. Il filtraggio a livello di colonna e riga non è supportato.

Il filtraggio dei dati può essere utile quando si desidera:

- Unisci determinate tabelle da due o più cluster di di origine diversi e non avrai bisogno di dati completi da nessuno dei due cluster di .
- Risparmia sui costi eseguendo analisi utilizzando solo un sottoinsieme di tabelle anziché un'intera flotta di database.
- Filtra le informazioni sensibili, come numeri di telefono, indirizzi o dettagli delle carte di credito, da determinate tabelle.

Puoi aggiungere filtri di dati a un'integrazione zero-ETL utilizzando AWS Management Console, the AWS Command Line Interface (AWS CLI) o l'API Amazon RDS.

Se l'integrazione ha come destinazione un cluster Amazon Redshift fornito, il cluster deve essere sulla patch 180 o successiva per utilizzare il filtraggio dei dati.

Argomenti

- Formato di un filtro dati
- Logica dei filtri
- Precedenza dei filtri

-
- Esempi di Aurora PostgreSQL
- Aggiungere filtri di dati a un'integrazione
- Rimozione dei filtri di dati da un'integrazione

Formato di un filtro dati

È possibile definire più filtri per una singola integrazione. Ogni filtro include o esclude tutte le tabelle di database esistenti e future che corrispondono a uno dei modelli nell'espressione del filtro. Le integrazioni Zero-ETL di Aurora utilizzano la sintassi del filtro Maxwell per il filtraggio dei dati.

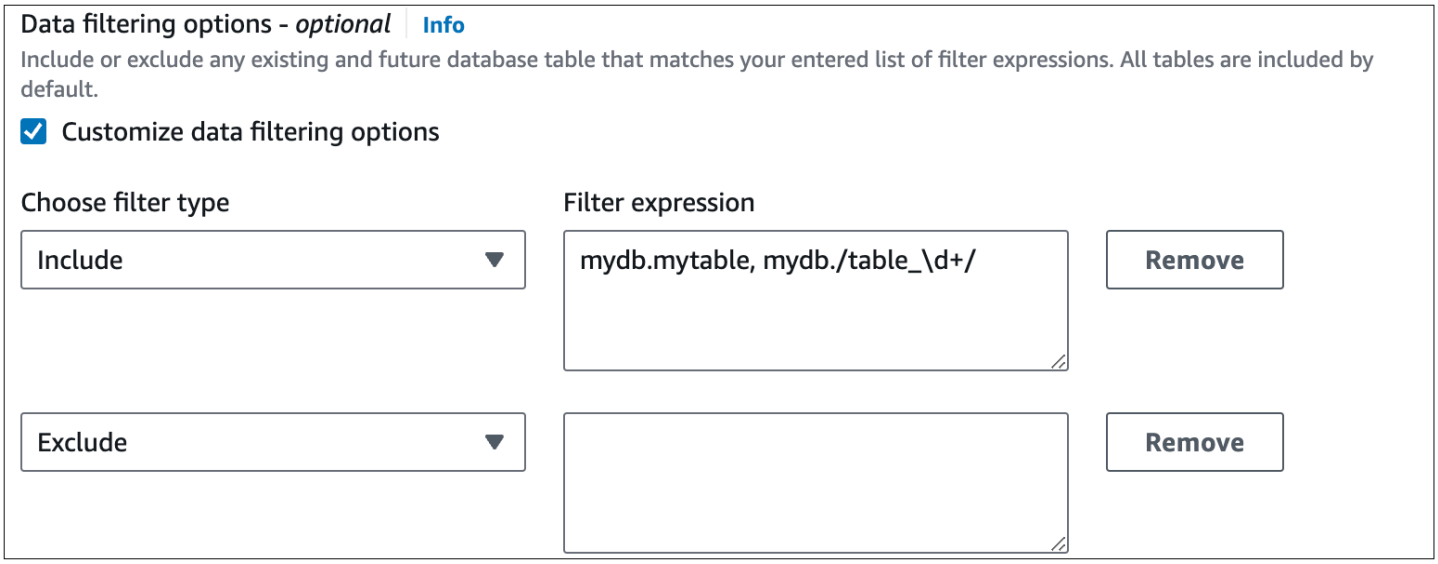
Ogni filtro ha i seguenti elementi:

Elemento	Descrizione
Tipo di filtro	Un tipo di `Include` filtro include tutte le tabelle che corrispondono a uno dei modelli nell'espressione del filtro. Un tipo di `Exclude` filtro esclude tutte le tabelle che corrispon dono a uno dei modelli.
Espressione filtro	Un elenco di modelli separati da virgole. Le espressioni devono utilizzare la sintassi del filtro Maxwell.
Pattern	Un pattern di filtro nel formato *`database`*`.`*`table`* per Aurora MySQL o per Aurora PostgreSQL. *`database`*`.`*`schema`*`.`*`table`* È possibile specificare nomi letterali o definire espressio ni regolari. **Note** Per Aurora MySQL, le espressio ni regolari sono supportate sia nel database che nel nome della tabella.

Elemento	Descrizione
	Per Aurora PostgreSQL, le espressioni regolari sono supportate solo nel nome dello schema e della tabella, non nel nome del database. Non è possibile includere filtri o denylist a livello di colonna. Una singola integrazione può avere un massimo di 99 pattern totali. Nella console, è possibile inserire modelli all'interno di una singola espressione di filtro o distribuirli tra più espressioni. Un singolo pattern non può superare i 256 caratteri di lunghezza.

> **Important**
>
> Se si seleziona un cluster DB di origine Aurora PostgreSQL, è necessario specificare almeno un modello di filtro dei dati. Come minimo, il modello deve includere un singolo database (*`database-name`*`.*.*`) per la replica su Amazon Redshift.

L'immagine seguente mostra la struttura dei filtri dati MySQL di Aurora nella console:



> **Important**
>
> Non includere informazioni di identificazione personale, riservate o sensibili nei modelli di filtro.

Filtri di dati in AWS CLI

Quando si utilizza AWS CLI per aggiungere un filtro dati, la sintassi è leggermente diversa da quella della console. È necessario assegnare un tipo di filtro (`IncludeoExclude`) a ciascun pattern singolarmente, in modo da non poter raggruppare più pattern in un unico tipo di filtro.

Ad esempio, nella console è possibile raggruppare i seguenti modelli separati da virgole in un'unica istruzione: `Include`

Aurora MySQL

```
mydb.mytable, mydb./table_\d+/
```

Aurora PostgreSQL

```
mydb.myschema.mytable, mydb.myschema./table_\d+/
```

Tuttavia, quando si utilizza il AWS CLI, lo stesso filtro dati deve avere il seguente formato:

Aurora MySQL

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

Aurora PostgreSQL

```
'include: mydb.myschema.mytable, include: mydb.myschema./table_\d+/'
```

Logica dei filtri

Se non specifichi alcun filtro di dati nella tua integrazione, Aurora presuppone un filtro predefinito `include:*.*` di, che replica tutte le tabelle nel data warehouse di destinazione. Tuttavia, se aggiungi almeno un filtro, la logica predefinita passa a`exclude:*.*`, che esclude tutte le tabelle per impostazione predefinita. Ciò consente di definire in modo esplicito quali database e tabelle includere nella replica.

Ad esempio, se si definisce il seguente filtro:

```
'include: db.table1, include: db.table2'
```

Aurora valuta il filtro come segue:

```
'exclude:*.*, include: db.table1, include: db.table2'
```

Pertanto, Aurora esegue solo `table1` `table2` repliche dal database `db` denominato al data warehouse di destinazione.

Precedenza dei filtri

Aurora valuta i filtri di dati nell'ordine specificato. In AWS Management Console, elabora le espressioni di filtro da sinistra a destra e dall'alto verso il basso. Un secondo filtro o un modello individuale che segue il primo può sostituirlo.

Ad esempio, se il primo filtro è`Include books.stephenking`, include solo la `stephenking` tabella del `books` database. Tuttavia, se si aggiunge un secondo filtro`Exclude books.*`, questo sostituisce il primo filtro. Ciò impedisce la replica di tutte le tabelle `books` dell'indice su Amazon Redshift.

Quando si specifica almeno un filtro, la logica parte dal presupposto predefinito che esclude automaticamente tutte le tabelle `exclude:*.*` dalla replica. Come procedura ottimale, definisci i filtri dal più ampio al più specifico. Inizia con una o più `Include` istruzioni per specificare i dati da replicare, quindi aggiungi `Exclude` filtri per rimuovere selettivamente determinate tabelle.

Lo stesso principio si applica ai filtri definiti utilizzando. AWS CLI Aurora valuta questi modelli di filtro nell'ordine in cui li specifichi, quindi un modello potrebbe sovrascrivere quello specificato in precedenza.

I seguenti esempi dimostrano come funziona il filtraggio dei dati per le integrazioni Aurora MySQL Zero-ETL:

- Includi tutti i database e tutte le tabelle:

```
'include: *.*'
```

- Includi tutte le tabelle all'interno del `books` database:

```
'include: books.*'
```

- Escludi tutte le tabelle denominate`mystery`:

```
'include: *.*, exclude: *.mystery'
```

- Includi due tabelle specifiche all'interno del `books` database:

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- Includi tutte le tabelle del `books` database, ad eccezione di quelle contenenti la sottostringa`mystery`:

```
'include: books.*, exclude: books./.*mystery.*/'
```

- Include tutte le tabelle del `books` database, ad eccezione di quelle che iniziano con`mystery`:

```
'include: books.*, exclude: books./mystery.*/'
```

- Include tutte le tabelle del `books` database, ad eccezione di quelle che terminano con`mystery`:

```
'include: books.*, exclude: books./.*mystery/'
```

- Include tutte le tabelle del `books` database che iniziano con`table_`, ad eccezione di quella denominata`table_stephen_king`. Ad esempio, `table_movies` o `table_books` verrebbe replicato, ma non`table_stephen_king`.

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

Esempi di Aurora PostgreSQL

I seguenti esempi dimostrano come funziona il filtraggio dei dati per le integrazioni Aurora PostgreSQL Zero-ETL:

- Includi tutte le tabelle all'interno del database: `books`

```
'include: books.*.*'
```

- Escludi tutte le tabelle denominate `mystery` nel `books` database:

```
'include: books.*.*, exclude: books.*.mystery'
```

- Includi una tabella all'interno del `books` database nello `mystery` schema e una tabella all'interno del `employee` database nello `finance` schema:

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- Includi tutte le tabelle del `books` database e `science_fiction` dello schema, ad eccezione di quelle contenenti la sottostringa`king`:

```
'include: books.science_fiction.*, exclude: books.*./.*king.*/
```

- Include tutte le tabelle del `books` database, ad eccezione di quelle con un nome di schema che inizia con`sci`:

```
'include: books.*.*, exclude: books./sci.*/.*'
```

- Include tutte le tabelle del `books` database, ad eccezione di quelle dello `mystery` schema che termina con`king`:

```
'include: books.*.*, exclude: books.mystery./.*king/'
```

- Include tutte le tabelle del `books` database che iniziano con`table_`, ad eccezione di quella denominata`table_stephen_king`. Ad esempio, `table_movies` nello `fiction` schema e `table_books` nello `mystery` schema vengono replicati, ma non `table_stephen_king` in nessuno dei due schemi:

```
'include: books.*./table_.*/, exclude: books.*.table_stephen_king'
```

Aggiungere filtri di dati a un'integrazione

Puoi configurare il filtraggio dei dati utilizzando AWS Management Console l'API AWS CLI Amazon RDS o la.

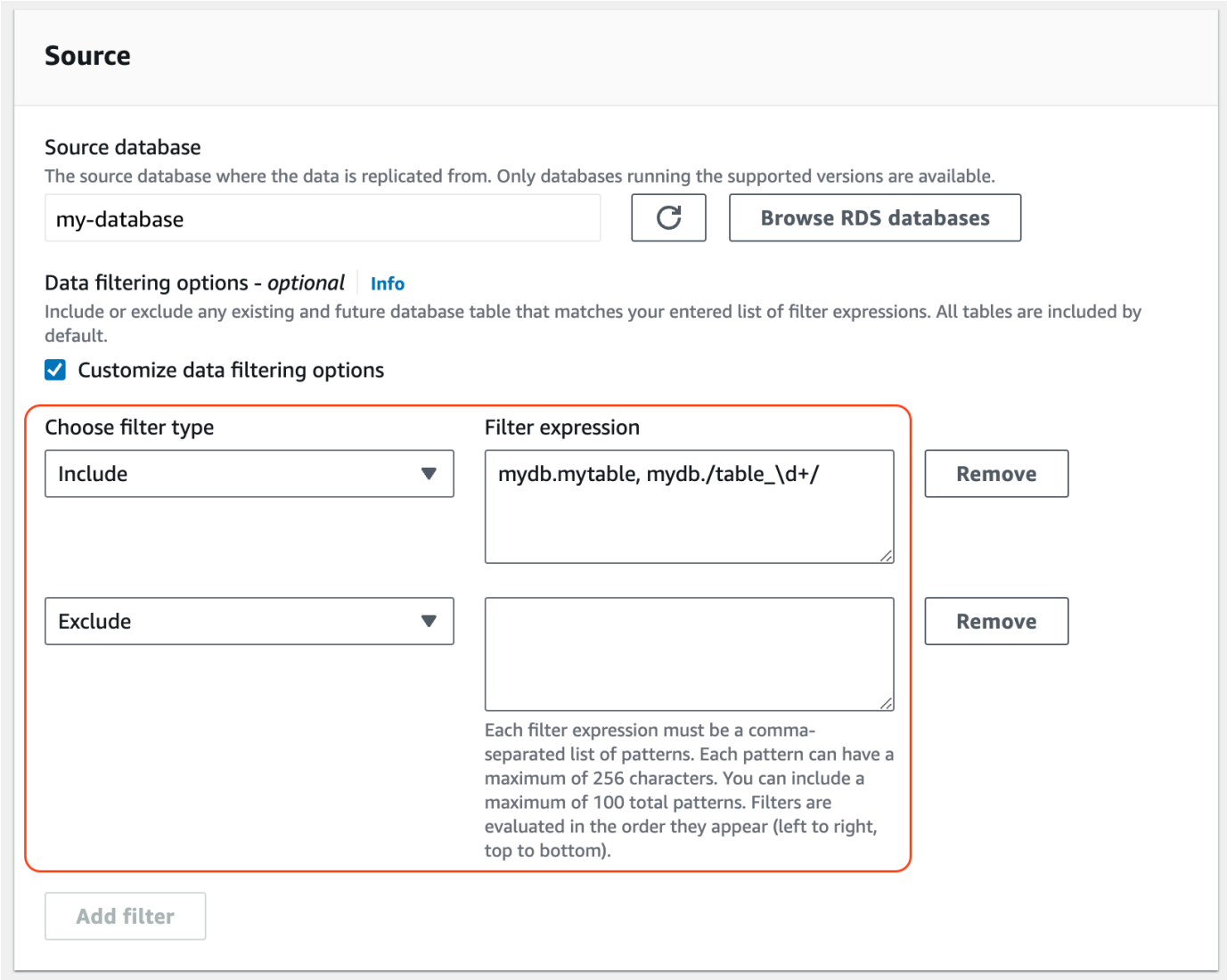
> **Important**
>
> Se aggiungi un filtro dopo aver creato un'integrazione, Aurora lo considera come se fosse sempre esistito. Rimuove tutti i dati nel data warehouse Amazon Redshift di destinazione che non soddisfano i nuovi criteri di filtraggio e risincronizza tutte le tabelle interessate.

Console RDS

Per aggiungere filtri di dati a un'integrazione zero-ETL

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Integrazioni zero-ETL. Seleziona l'integrazione a cui desideri aggiungere filtri di dati, quindi scegli Modifica.
3. In Source, aggiungi una o più `Exclude` istruzioni `Include` e.

   L'immagine seguente mostra un esempio di filtri di dati per un'integrazione MySQL:



4. Quando sei soddisfatto delle modifiche, scegli Continua e salva le modifiche.

AWS CLI

Per aggiungere filtri di dati a un'integrazione zero-ETL utilizzando AWS CLI, chiama il comando modify-integration. Oltre all'identificatore di integrazione, specificate il `--data-filter` parametro con un elenco separato da virgole di filtri Maxwell. `Include` `Exclude`

Example

L'esempio seguente aggiunge modelli di filtro a. `my-integration`

In Linux, macOS, oppure Unix:

```
aws rds modify-integration \
```

```
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

In Windows:

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

API RDS

Per modificare un'integrazione zero-ETL utilizzando l'API RDS, chiama l'operazione. ModifyIntegration Specificate l'identificatore di integrazione e fornite un elenco di modelli di filtro separati da virgole.

Rimozione dei filtri di dati da un'integrazione

Quando rimuovi un filtro dati da un'integrazione, Aurora rivaluta i filtri rimanenti come se il filtro rimosso non fosse mai esistito. Quindi replica tutti i dati precedentemente esclusi che ora soddisfano i criteri nel data warehouse Amazon Redshift di destinazione. Ciò attiva una risincronizzazione di tutte le tabelle interessate.

Aggiungere dati a un cluster Aurora DB) e interrogarli in Amazon Redshift

Per completare la creazione di un'integrazione Zero-ETL che replichi i dati da Amazon Aurora in Amazon Redshift, devi creare un database di destinazione in Amazon Redshift.

Innanzitutto, connettiti al cluster o al gruppo di lavoro Amazon Redshift e crea un database con un riferimento al tuo identificatore di integrazione. Quindi, puoi aggiungere dati al cluster Aurora DB del di origine e vederli replicati in Amazon Redshift.

Argomenti

- Creazione di un database di destinazione in Amazon Redshift
-
- Interrogazione dei dati di in Amazon Redshift
- Differenze tra i tipi di dati tra i database Aurora e Amazon Redshift

- Operazioni DDL per Aurora PostgreSQL

Creazione di un database di destinazione in Amazon Redshift

Prima di poter iniziare a replicare i dati in Amazon Redshift dopo la creazione di un'integrazione, devi creare un database di destinazione nel data warehouse di destinazione. Questo database di destinazione deve includere un riferimento all'identificatore di integrazione. Puoi utilizzare la console Amazon Redshift o Editor di query v2 per creare il database.

Per istruzioni sulla creazione di un database di destinazione, consulta Creazione di un database di destinazione in Amazon Redshift.

Dopo aver configurato l'integrazione, puoi aggiungere alcuni dati al cluster Aurora DB del che desideri replicare nel tuo data warehouse Amazon Redshift.

> **Note**
>
> Esistono differenze tra i tipi di dati in Amazon Aurora e Amazon Redshift. Per una tabella di mappature dei tipi di dati, consulta the section called “Differenze dei tipi di dati”.

Innanzitutto, connettiti al cluster DB del di origine utilizzando il client MySQL o PostgreSQL di tua scelta. Per istruzioni, consulta the section called “Connessione a un cluster database”.

Quindi, crea una tabella e inserisci una riga di dati di esempio.

> **Important**
>
> Assicurati che la tabella abbia una chiave primaria. Altrimenti, non può essere replicata nel data warehouse di destinazione.

Le utilità pg_dump e pg_restore PostgreSQL inizialmente creano tabelle senza una chiave primaria e poi le aggiungono in seguito. Se utilizzi una di queste utilità, ti consigliamo di creare prima uno schema e poi di caricare i dati in un comando separato.

MySQL

L'esempio seguente utilizza l'utilità MySQL Workbench.

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

PostgreSQL

L'esempio seguente utilizza il terminale `psql` interattivo PostgreSQL. Quando ti connetti al cluster, includi il database denominato che hai specificato durante la creazione dell'integrazione.

```
psql -h mycluster.cluster-123456789012.us-east-2.rds.amazonaws.com -p 5432 -U username
 -d named_db;

named_db=> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL,
 Author VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

named_db=> INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977,
 'Supernatural fiction');
```

Interrogazione dei dati di in Amazon Redshift

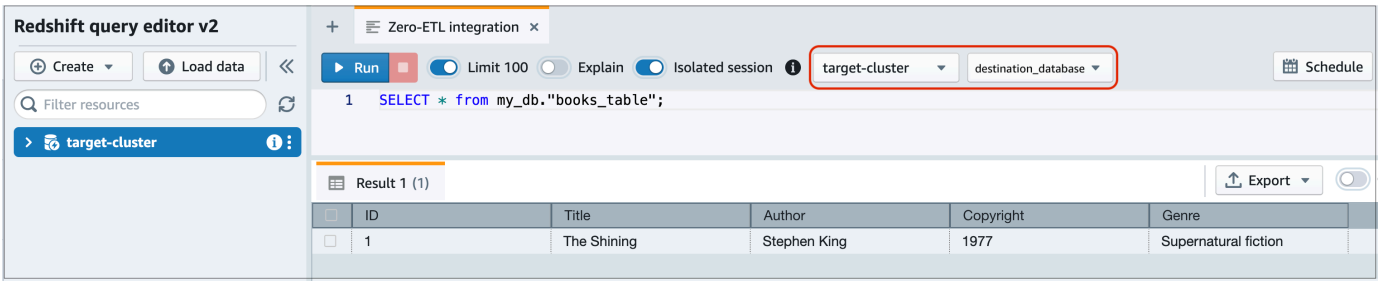
Dopo aver aggiunto i dati al cluster Aurora DB del , questi vengono replicati in Amazon Redshift e sono pronti per essere interrogati.

Esecuzione di query sui dati replicati

1. Vai alla console Amazon Redshift e scegli Editor di query v2 nel riquadro di navigazione a sinistra.
2. Connettiti al cluster o al gruppo di lavoro e scegli il database di destinazione (creato dall'integrazione) dal menu a tendina (destination_database in questo esempio). Per istruzioni sulla creazione di un database di destinazione, consulta Creazione di un database di destinazione in Amazon Redshift.

3. Usa un'istruzione SELECT per interrogare i tuoi dati. In questo esempio, è possibile eseguire il comando seguente per selezionare tutti i dati dalla tabella creata nel di origine del cluster Aurora DB:

```
SELECT * from my_db."books_table";
```



- *my_db* è il nome dello schema del database Aurora. Questa opzione è necessaria solo per i database MySQL.
- *books_table* è il nome della tabella Aurora.

È inoltre possibile interrogare i dati utilizzando un client a riga di comando. Per esempio:

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                 Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+-----------------------+----------
+--------+
  1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
 12192
```

> **Note**
>
> Per distinguere tra maiuscole e minuscole, usa le virgolette doppie (" ") per i nomi di schemi, tabelle e colonne. Per ulteriori informazioni, consulta enable_case_sensitive_identifier.

Differenze tra i tipi di dati tra i database Aurora e Amazon Redshift

La o Aurora PostgreSQL con un tipo di dati Amazon Redshift corrispondente. Amazon Aurora attualmente supporta solo questi tipi di dati per integrazioni zero-ETL.

Se una tabella nel cluster DB del di origine include un tipo di dati non supportato, la tabella non è sincronizzata e non è utilizzabile dal target Amazon Redshift. Lo streaming dall'origine alla destinazione va avanti, ma la tabella con il tipo di dati non supportato non è disponibile. Per correggere la tabella e renderla disponibile in Amazon Redshift, devi annullare manualmente la modifica iniziale e aggiornare l'integrazione eseguendo `ALTER DATABASE...INTEGRATION REFRESH`.

Argomenti

-
- Aurora PostgreSQL

Tipo di dati Aurora MySQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
INT	INTEGER	Intero a quattro byte firmato	Nessuno
SMALLINT	SMALLINT	Intero a due byte firmato	Nessuno
TINYINT	SMALLINT	Intero a due byte firmato	Nessuno
MEDIUMINT	INTEGER	Intero a quattro byte firmato	Nessuno
BIGINT	BIGINT	Intero a otto byte firmato	Nessuno
INT UNSIGNED	BIGINT	Intero a otto byte firmato	Nessuno
TINYINT UNSIGNED	SMALLINT	Intero a due byte firmato	Nessuno
MEDIUMINT UNSIGNED	INTEGER	Intero a quattro byte firmato	Nessuno

Tipo di dati Aurora MySQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
BIGINT UNSIGNED	DECIMAL(20,0)	Numerico esatto di precisione selezionabile	Nessuno
DECIMALE (p, s) = NUMERICO (p, s)	DECIMAL(p,s)	Numerico esatto di precisione selezionabile	Precisione superiore a 38 e scala superiore a 37 non supportat e
DECIMALE (p, s) SENZA SEGNO = NUMERICO (p, s) SENZA SEGNO	DECIMAL(p,s)	Numerico esatto di precisione selezionabile	Precisione superiore a 38 e scala superiore a 37 non supportat e
FLOAT4/REALE	REAL	Numero in virgola mobile a precisione singola	Nessuno
FLOAT4/REAL NON FIRMATO	REAL	Numero in virgola mobile a precisione singola	Nessuno
DOUBLE/REAL/FLOAT8	DOUBLE PRECISION	Numero in virgola mobile a precisione doppia	Nessuno
DOUBLE/REAL/FLOAT8 NON FIRMATO	DOUBLE PRECISION	Numero in virgola mobile a precisione doppia	Nessuno

Tipo di dati Aurora MySQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
BIT (n)	BARBYTE(8)	Valore binario a lunghezza variabile	Nessuno
BINARY(n)	BARBYTE (n)	Valore binario a lunghezza variabile	Nessuno
VARBINARY(n)	VARBYTE (n)	Valore binario a lunghezza variabile	Nessuno
CHAR(n)	VARCHAR(n)	Valore di stringa di lunghezza variabile	Nessuno
VARCHAR(n)	VARCHAR(n)	Valore di stringa di lunghezza variabile	Nessuno
TEXT	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
TINYTEXT	VARCHAR(255)	Valore di stringa a lunghezza variabile fino a 255 caratteri	Nessuno
MEDIUMTEXT	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno

Tipo di dati Aurora MySQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
LONGTEXT	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
ENUM	VARCHAR(1020)	Valore di stringa a lunghezza variabile fino a 1.020 caratteri	Nessuno
SET	VARCHAR(1020)	Valore di stringa a lunghezza variabile fino a 1.020 caratteri	Nessuno
DATE	DATE	Data di calendario (anno, mese, giorno)	Nessuno
DATETIME	TIMESTAMP	Data e ora (senza fuso orario)	Nessuno
TIMESTAMP(p)	TIMESTAMP	Data e ora (senza fuso orario)	Nessuno
TIME	VARCHAR(18)	Valore di stringa a lunghezza variabile fino a 18 caratteri	Nessuno

Tipo di dati Aurora MySQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
ANNO	VARCHAR(4)	Valore di stringa a lunghezza variabile fino a 4 caratteri	Nessuno
JSON	SUPER	Dati o documenti semistrutturati come valori	Nessuno

Aurora PostgreSQL

Le integrazioni zero-ETL per Aurora PostgreSQL non supportano tipi di dati personalizzati o tipi di dati creati da estensioni.

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
array	SUPER	Dati o documenti semistrutturati come valori	Nessuno
bigint	BIGINT	Intero a otto byte firmato	Nessuno
grande seriale	BIGINT	Intero a otto byte firmato	Nessuno
bit variabile (n)	BARBYTE (n)	Valore binario a lunghezza variabile fino a 16.777.216 byte	Nessuno
bit (n)	BARBYTE (n)	Valore binario a lunghezza	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
		variabile fino a 16.777.216 byte	
bit, bit variabile	BARBYTE (16777216)	Valore binario a lunghezza variabile fino a 16.777.216 byte	Nessuno
Booleano	BOOLEAN	Booleano logico (vero/falso)	Nessuno
bytea	BARBYTE (16777216)	Valore binario a lunghezza variabile fino a 16.777.216 byte	Nessuno
carattere (n)	CHAR(n)	Valore di stringa di caratteri a lunghezza fissa fino a 65.535 byte	Nessuno
carattere variabile (n)	VARCHAR(65535)	Valore della stringa di caratteri a lunghezza variabile fino a 65.535 caratteri	Nessuno
cid	BIGINT	Intero a otto byte firmato	Nessuno
sidro	ARCHAR (19)	Valore di stringa a lunghezza variabile fino a 19 caratteri	

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
data	DATE	Data di calendario (anno, mese, giorno)	Valori superiori a 294.276 d.C. Non sono supportati
double precision	DOUBLE PRECISION	Numeri a virgola mobile a doppia precisione	Valori subnormali non completamente supportati
gtsvector	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
inet	VARCHAR (19)	Valore di stringa a lunghezza variabile fino a 19 caratteri	Nessuno
integer	INTEGER	Intero a quattro byte firmato	Nessuno
int2 vector	SUPER	Dati o documenti semistrutturati come valori.	Nessuno
intervallo	INTERVAL	Durata del tempo	Sono supportati solo i tipi INTERVAL che specificano un qualificatore anno/mese o un qualificatore da giorno a secondo.

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
json	SUPER	Dati o documenti semistrutturati come valori	Nessuno
jsonb	SUPER	Dati o documenti semistrutturati come valori	Nessuno
percorso json	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
macaddr	VARCHAR (17)	Valore di stringa a lunghezza variabile fino a 17 caratteri	Nessuno
macaddr8	BARRA (23)	Valore di stringa a lunghezza variabile fino a 23 caratteri	Nessuno
money	DECIMALE (20,3)	Importo in valuta	Nessuno
nome	VARCHAR (64)	Valore di stringa a lunghezza variabile fino a 64 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
numeric(p,s)	DECIMAL(p,s)	Valore di precisione fisso definito dall'utente	• `NaN`valori non supportati • La precisione e la scala devono essere definite in modo esplicito e non devono essere superiori a 38 (precisione) e 37 (scala) • Scala negativa non supportata
oid	BIGINT	Intero a otto byte firmato	Nessuno
vettore oid	SUPER	Dati o documenti semistrutturati come valori.	Nessuno
pg_brin_bloom_summary	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_dependencies	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
pg_lsn	VARCHAR (17)	Valore di stringa a lunghezza variabile fino a 17 caratteri	Nessuno
pg_mcv_list	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_ndistinct	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_node_tree	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
pg_snapshot	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
real	REAL	Numero in virgola mobile a precisione singola	Valori subnormali non completamente supportati
recursore	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
smallint	SMALLINT	Intero a due byte firmato	Nessuno
seriale piccolo	SMALLINT	Intero a due byte firmato	Nessuno
seriale	INTEGER	Intero a quattro byte firmato	Nessuno
text	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
tid	VARCHAR (23)	Valore di stringa a lunghezza variabile fino a 23 caratteri	Nessuno
ora [(p)] senza fuso orario	VARCHAR (19)	Valore di stringa a lunghezza variabile fino a 19 caratteri	`Infinity` e `-Infinity` valori non supportati
ora [(p)] con fuso orario	VARCHAR (22)	Valore di stringa a lunghezza variabile fino a 22 caratteri	`Infinity` e `-Infinity` valori non supportati

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
timestamp [(p)] senza fuso orario	TIMESTAMP	Data e ora (senza fuso orario)	• `Infinitye -Infinity` valori non supportati • Valori maggiori di quelli `9999-12-31` non supportati • Valori B.C. non supportati
timestamp [(p)] con fuso orario	TIMESTAMPTZ	Data e ora (con fuso orario)	• `Infinitye -Infinity` valori non supportati • Valori maggiori di quelli `9999-12-31` non supportati • Valori B.C. non supportati
tsquery	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
tsvector	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno

Tipo di dati Aurora PostgreSQL	Tipo di dati di Amazon Redshift	Descrizione	Limitazioni
txid_snapshot	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno
uuid	VARCHAR (36)	Stringa di 36 caratteri a lunghezza variabile	Nessuno
xid	BIGINT	Intero a otto byte firmato	Nessuno
xid8	DECIMALE (20, 0)	Decimale a precisione fissa	Nessuno
xml	VARCHAR(65535)	Valore di stringa a lunghezza variabile fino a 65.535 caratteri	Nessuno

Operazioni DDL per Aurora PostgreSQL

Amazon Redshift è derivato da PostgreSQL, quindi condivide diverse funzionalità con Aurora PostgreSQL grazie alla loro architettura PostgreSQL comune. Le integrazioni zero-ETL sfruttano queste somiglianze per semplificare la replica dei dati da Aurora PostgreSQL ad Amazon Redshift, mappando i database per nome e utilizzando la struttura condivisa di database, schemi e tabelle.

Considera i seguenti punti quando gestisci le integrazioni Aurora PostgreSQL Zero-ETL:

- L'isolamento è gestito a livello di database.
- La replica avviene a livello di database.
- I database Aurora PostgreSQL sono mappati ai database Amazon Redshift per nome, con i dati che fluiscono verso il corrispondente database Redshift rinominato se l'originale viene rinominato.

Nonostante le loro somiglianze, Amazon Redshift e Aurora PostgreSQL presentano differenze importanti. Le seguenti sezioni descrivono le risposte del sistema Amazon Redshift per le operazioni DDL comuni.

Argomenti

- Operazioni di database
- Operazioni dello schema
- Operazioni sulle tabelle

Operazioni di database

La tabella seguente mostra le risposte di sistema per le operazioni DDL del database.

Operazione DDL	Risposta del sistema Redshift
`CREATE DATABASE`	Nessuna operazione
`DROP DATABASE`	Amazon Redshift elimina tutti i dati nel database Redshift di destinazione.
`RENAME DATABASE`	Amazon Redshift elimina tutti i dati nel database di destinazione originale e risincronizza i dati nel nuovo database di destinazione. Se il nuovo database non esiste, devi crearlo manualmente. Per istruzioni, consulta Creare un database di destinazione in Amazon Redshift.

Operazioni dello schema

La tabella seguente mostra le risposte del sistema per le operazioni DDL dello schema.

Operazione DDL	Risposta del sistema Redshift
`CREATE SCHEMA`	Nessuna operazione
`DROP SCHEMA`	Amazon Redshift elimina lo schema originale.

Operazione DDL	Risposta del sistema Redshift
`RENAME SCHEMA`	Amazon Redshift elimina lo schema originale, quindi risincronizza i dati nel nuovo schema.

Operazioni sulle tabelle

La tabella seguente mostra le risposte del sistema per le operazioni DDL della tabella.

Operazione DDL	Risposta del sistema Redshift
`CREATE TABLE`	Amazon Redshift crea la tabella. Alcune operazioni impediscono la creazione della tabella, ad esempio la creazione di una tabella senza una chiave primaria o l'esecuzi one del partizionamento dichiarativo. Per ulteriori informazioni, consulta the section called “Limitazioni di Aurora PostgreSQL” e the section called “Risoluzione dei problemi relativi alle integrazioni zero ETL”.
`DROP TABLE`	Amazon Redshift abbandona la situazione.
`TRUNCATE TABLE`	Amazon Redshift tronca la tabella.
`ALTER TABLE (RENAME... )`	Amazon Redshift rinomina la tabella o la colonna.
`ALTER TABLE (SET SCHEMA)`	Amazon Redshift elimina la tabella nello schema originale e risincronizza la tabella nel nuovo schema.
`ALTER TABLE (ADD PRIMARY KEY)`	Amazon Redshift aggiunge una chiave primaria e risincronizza la tabella.
`ALTER TABLE (ADD COLUMN)`	Amazon Redshift aggiunge una colonna alla tabella.

Operazione DDL	Risposta del sistema Redshift
`ALTER TABLE (DROP COLUMN)`	Amazon Redshift elimina la colonna se non è una colonna chiave primaria. Altrimenti, risincronizza la tabella.
`ALTER TABLE (SET LOGGED/UNLOGGED )`	Se modifichi la tabella in loggata, Amazon Redshift risincronizza la tabella. Se modifichi la tabella in unlogged, Amazon Redshift elimina la tabella.

Visualizzazione e monitoraggio delle ETL integrazioni zero di Aurora con Amazon Redshift

Puoi visualizzare i dettagli di un'ETLintegrazione zero di Aurora per visualizzarne le informazioni di configurazione e lo stato attuale. Puoi anche monitorare lo stato della tua integrazione eseguendo query sulle viste di sistema specifiche in Amazon Redshift. Inoltre, Amazon Redshift pubblica alcuni parametri relativi all'integrazione su Amazon, che puoi visualizzare all'interno della CloudWatch console Amazon Redshift.

Argomenti

- Visualizzazione delle integrazioni
- Monitoraggio delle integrazioni tramite tabelle di sistema
- Monitoraggio delle integrazioni con Amazon EventBridge

Visualizzazione delle integrazioni

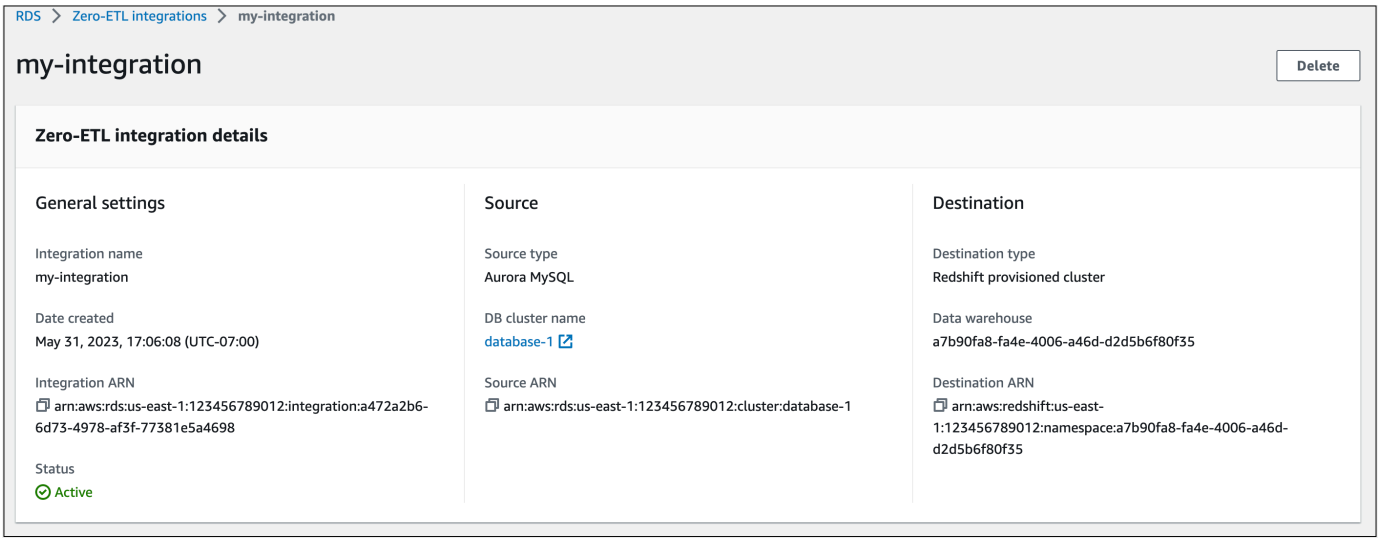
Puoi visualizzare le ETL integrazioni zero di Aurora con Amazon Redshift utilizzando il AWS Management Console, il, o il. AWS CLI RDS API

Console

Per visualizzare i dettagli di un'integrazione zero ETL

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.

2. Dal riquadro di navigazione a sinistra, scegli Zero- ETL integrazioni.
3. Seleziona un'integrazione per visualizzarne maggiori dettagli, ad esempio il di origine, il cluster DB e il data warehouse di destinazione.



Un'integrazione può avere i seguenti stati:

- `Creating`: l'integrazione è in fase di creazione.
- `Active`: l'integrazione sta inviando dati transazionali al data warehouse di destinazione.
- `Syncing`: l'integrazione ha rilevato un errore recuperabile e deve reimpostare i dati. Le tabelle interessate non sono disponibili per l'interrogazione in Amazon Redshift fino al termine della risincronizzazione.
- `Needs attention`: l'integrazione ha rilevato un evento o un errore che richiede un intervento manuale per la risoluzione. Per correggere il problema, segui le istruzioni nel messaggio di errore nella pagina dei dettagli dell'integrazione.
- `Failed`: l'integrazione ha rilevato un evento o un errore irreversibile che non può essere risolto. È necessario eliminare e ricreare l'integrazione.
- `Deleting`: l'integrazione è in fase di eliminazione.

AWS CLI

Per visualizzare tutte le ETL integrazioni zero nell'account corrente utilizzando AWS CLI, usa il comando describe-integrations e specifica l'opzione. `--integration-identifier`

Example

In Linux, macOS, oppure Unix:

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

In Windows:

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

RDS API

Per visualizzare ETL l'integrazione zero utilizzando Amazon RDSAPI, utilizza l'`DescribeIntegrations`operazione con il `IntegrationIdentifier` parametro.

Monitoraggio delle integrazioni tramite tabelle di sistema

Amazon Redshift include tabelle e viste di sistema che contengono informazioni sul funzionamento del sistema. Puoi eseguire delle query su queste tabelle e viste esattamente come faresti con qualsiasi altra tabella di database. Per ulteriori informazioni sulle tabelle e le viste di sistema in Amazon Redshift, consulta System tables and views reference nella Amazon Redshift Database Developer Guide.

Puoi interrogare le seguenti visualizzazioni e tabelle di sistema per ottenere informazioni sulle ETL integrazioni zero di Aurora con Amazon Redshift:

- SVV_ INTEGRATION — Fornisce dettagli di configurazione per le tue integrazioni.
- SVV_ _ INTEGRATION TABLE _ STATE — Descrive lo stato di ogni tabella all'interno di un'integrazione.
- SYS_ INTEGRATION _ _ TABLE STATE _ CHANGE — Visualizza i registri delle modifiche allo stato della tabella per un'integrazione.
- SYS_ INTEGRATION _ ACTIVITY — Fornisce informazioni sulle esecuzioni di integrazione completate.

Tutte le metriche Amazon relative all'integrazione provengono da CloudWatch Amazon Redshift. Per ulteriori informazioni, consulta Metriche per ETL le integrazioni zero nella Amazon Redshift

Management Guide. Al momento, Aurora non pubblica alcuna metrica di integrazione su. CloudWatch

Monitoraggio delle integrazioni con Amazon EventBridge

Amazon Redshift invia eventi relativi all'integrazione ad Amazon. EventBridge Per un elenco di eventi e l'evento corrispondenteIDs, consulta Notifiche di eventi di ETL integrazione zero con Amazon EventBridge nella Amazon Redshift Management Guide.

Modifica delle integrazioni zero di Aurora con Amazon ETL Redshift

Puoi modificare solo il nome, la descrizione e le opzioni di filtraggio dei dati per un'ETLintegrazione zero con Amazon Redshift. Non puoi modificare la AWS KMS chiave utilizzata per crittografare l'integrazione o i database di origine o di destinazione.

Se aggiungi un filtro dati a un'integrazione esistente, Aurora rivaluta il filtro come se fosse sempre esistito. Rimuove tutti i dati attualmente presenti nel data warehouse Amazon Redshift di destinazione che non soddisfano i nuovi criteri di filtraggio. Se rimuovi un filtro dati da un'integrazione, esso replica tutti i dati che in precedenza non corrispondevano ai criteri di filtro (ma che ora lo fanno) nel data warehouse di destinazione. Per ulteriori informazioni, consulta the section called “Filtraggio dei dati per integrazioni zero-ETL”.

Puoi modificare un'ETLintegrazione zero utilizzando Amazon AWS Management Console AWS CLI, the o Amazon RDSAPI.

Console RDS

Per modificare un'ETLintegrazione zero

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegli Zero- ETL integrazioni, quindi scegli l'integrazione che desideri modificare.
3. Scegli Modifica e apporta modifiche a tutte le impostazioni disponibili.
4. Quando tutte le modifiche sono come desideri, scegli Modifica.

AWS CLI

Per modificare un'ETLintegrazione zero utilizzando AWS CLI, chiamate il comando modify-integration. Oltre a `--integration-identifier`, specificate una delle seguenti opzioni:

- `--integration-name`— Specificate un nuovo nome per l'integrazione.
- `--description`— Specificare una nuova descrizione per l'integrazione.
- `--data-filter`— Specificare le opzioni di filtraggio dei dati per l'integrazione. Per ulteriori informazioni, consulta the section called “Filtraggio dei dati per integrazioni zero-ETL”.

Example

La richiesta seguente modifica un'integrazione esistente.

In Linux, macOS, oppure Unix:

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

In Windows:

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
    --integration-name my-renamed-integration
```

RDS API

Per modificare un'ETLintegrazione zero utilizzando RDSAPI, chiamate l'ModifyIntegrationoperazione. Specificate l'identificatore di integrazione e i parametri che desiderate modificare.

Eliminazione delle integrazioni zero di Aurora con Amazon ETL Redshift

Quando elimini un'ETLintegrazione zero, Aurora la rimuove dal database di origine del cluster . I tuoi dati transazionali non vengono eliminati da Amazon Aurora o Amazon Redshift, ma Aurora non invia nuovi dati ad Amazon Redshift.

Puoi eliminare un'integrazione solo quando ha lo stato di,, o. `Active` `Failed` `Syncing` `Needs attention`

È possibile eliminare ETL le integrazioni zero utilizzando il AWS Management Console AWS CLI, il o il RDSAPI.

Console

Per eliminare un'integrazione zero ETL

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Dal riquadro di navigazione a sinistra, scegli Zero- ETL integrazioni.
3. Seleziona l'ETLintegrazione zero che desideri eliminare.
4. Scegli Operazioni, Elimina dominio, quindi conferma l'eliminazione.

AWS CLI

Per eliminare un'ETLintegrazione zero, utilizzate il comando delete-integration e specificate l'`--integration-identifier`opzione.

Example

In Linux, macOS, oppure Unix:

```
aws rds delete-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

In Windows:

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

RDS API

Per eliminare un'ETLintegrazione zero utilizzando Amazon RDSAPI, utilizza l'`DeleteIntegration`operazione con il `IntegrationIdentifier` parametro.

Risoluzione dei problemi delle ETL integrazioni zero di Aurora con Amazon Redshift

Puoi verificare lo stato di un'ETLintegrazione zero interrogando la tabella di INTEGRATION sistema SVV_ in Amazon Redshift. Se la colonna `state` include il valore `ErrorState`, significa che si è verificato un problema. Per ulteriori informazioni, consulta the section called “Monitoraggio tramite tabelle di sistema”.

Utilizza le seguenti informazioni per risolvere i problemi più comuni relativi alle integrazioni zero di Amazon ETL con Amazon Redshift.

Argomenti

- Non riesco a creare un'integrazione zero ETL
- La mia integrazione è bloccata in uno stato di Syncing
- Le mie tabelle non si replicano su Amazon Redshift
- Una o più tabelle Amazon Redshift richiedono una risincronizzazione
- DDLle modifiche vengono apportate in Amazon Redshift prima del completamento della DDL transazione per Aurora Postgre SQL

Non riesco a creare un'integrazione zero ETL

Se non riesci a creare un'ETLintegrazione zero, assicurati che quanto segue sia corretto per il tuo database di origine:

- Il database di origine deve eseguire una versione del motore DB supportata. Per un elenco delle versioni supportate, consulta the section called “Integrazioni Zero-ETL ”.
- I parametri del DB sono stati configurati correttamente. Se i parametri richiesti sono impostati in modo errato o non sono associati al database, la creazione non riesce. Per informazioni, consulta the section called “Fase 1: creazione di un gruppo di parametri del cluster DB personalizzato”.

Inoltre, assicurati che quanto segue sia corretto per il data warehouse di destinazione:

- È abilitata la distinzione tra maiuscole e minuscole. Consulta Attivazione della distinzione tra maiuscole e minuscole per il data warehouse.

- Hai aggiunto il principale autorizzato e l'origine dell'integrazione corretti. Vedi Configurare l'autorizzazione per il tuo data warehouse Amazon Redshift.
- Il data warehouse è crittografato (se si tratta di un cluster fornito). Vedi Crittografia del database Amazon Redshift.

La mia integrazione è bloccata in uno stato di **`Syncing`**

La tua integrazione potrebbe mostrare costantemente uno stato pari a `Syncing` se modifichi il valore di uno dei parametri del DB richiesti.

Per risolvere questo problema, controlla i valori dei parametri nel gruppo di parametri associato al cluster DB del di origine e assicurati che corrispondano ai valori richiesti. Per ulteriori informazioni, consulta the section called “Fase 1: creazione di un gruppo di parametri del cluster DB personalizzato”.

Se modificate qualche parametro, assicuratevi di riavviare il cluster DB del per applicare le modifiche.

Le mie tabelle non si replicano su Amazon Redshift

Se non vedi una o più tabelle riflesse in Amazon Redshift, puoi eseguire il seguente comando per risincronizzarle:

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

Per ulteriori informazioni, consulta la ALTERDATABASEpagina di riferimento di Amazon RedshiftSQL.

I tuoi dati potrebbero non essere replicati perché una o più tabelle di origine non hanno una chiave primaria. La dashboard di monitoraggio in Amazon Redshift mostra lo stato di queste tabelle e lo stato dell'ETLintegrazione zero complessiva cambia a. `Failed` `Needs attention` Per risolvere questo problema, puoi identificare una chiave esistente nella tabella che può diventare una chiave primaria oppure puoi aggiungere una chiave primaria sintetica. Per soluzioni dettagliate, consulta . le seguenti risorse:

- Gestisci le tabelle senza chiavi primarie durante la creazione di ETL integrazioni SQL zero di SQL Amazon Aurora My o Amazon RDS for My con Amazon Redshift
- Gestisci le tabelle senza chiavi primarie mentre crei integrazioni SQL zero di Amazon Aurora Postgre con Amazon Redshift ETL

Una o più tabelle Amazon Redshift richiedono una risincronizzazione

L'esecuzione di determinati comandi sul database di origine potrebbe richiedere la risincronizzazione delle tabelle. In questi casi, la vista di STATE sistema SVVINTEGRATIONTABLE_ _ _ mostra un `table_state` di`ResyncRequired`, il che significa che l'integrazione deve ricaricare completamente i dati per quella tabella specifica da My SQL ad Amazon Redshift.

Quando viene avviata la risincronizzazione della tabella, lo stato diventa `Syncing`. Non è necessario eseguire alcuna azione manuale per risincronizzare una tabella. Durante la risincronizzazione dei dati delle tabelle, non puoi accedervi in Amazon Redshift.

Di seguito sono riportati alcuni esempi di operazioni che possono modificare lo stato di una tabella in `ResyncRequired` e le possibili alternative da considerare.

Operazione	Esempio	In alternativa

Aggiunta di una colonna in una posizione specifica	`ALTER TABLE table_name`
	`  ADD COLUMN column_name  INTEGER`
	`  NOT NULL first;`
		Amazon Redshift non supporta l'aggiunta di colonne in posizioni specifich e utilizzan do le parole chiave `first` o `after`. Se l'ordine delle colonne nella tabella di destinazi one non è rilevante , aggiungi la colonna alla fine della tabella utilizzando un

Operazione	Esempio	In alternativa
		comando più semplice: `ALTER TABLE table_name ADD COLUMN column_name column_type ;`

Operazione	Esempio	In alternativa

Aggiunta di una colonna timestamp con il valore predefinito to CURRENT_TIMESTAMP	`ALTER TABLE table_name`
	`  ADD COLUMN column_name  TIMESTAMP`
	`  NOT NULL DEFAULT CURRENT_TIMESTAMP;`
		Il CURRENT_TIMESTAMP valore per le righe della tabella esistenti viene calcolato da e non può essere simulato in Amazon Redshift senza la risincronizzazione completa dei dati della tabella.
		
		Se possibile, converti il valore predefinito in una costante letterale, ad esempio 2023-01-01 00:00:15, per evitare la latenza a livello di disponibilità della tabella.

Operazione	Esempio	In alternativa
Esecuzione di operazioni su più colonne all'interno di un unico comando	`ALTER TABLE table_name ADD COLUMN column_1, RENAME COLUMN column_2 TO column_3;`	Prendi in considerazione la possibilità di suddividere il comando in due operazioni distinte, `ADD` e `RENAME`, che non richiedono la risincronizzazione.

DDLle modifiche vengono apportate in Amazon Redshift prima del completamento della DDL transazione per Aurora Postgre SQL

DDLle modifiche possono apparire in Amazon Redshift prima del termine di un'DDLoperazione nelle integrazioni zero di Aurora SQL Postgre. ETL Per ulteriori informazioni, consulta Operazioni DDL per Aurora PostgreSQL.

Utilizzo Aurora Serverless v2

Aurora Serverless v2 è una configurazione on-demand con scalabilità automatica per Amazon Aurora. Aurora Serverless v2 aiuta ad automatizzare i processi di monitoraggio del carico di lavoro e di regolazione della capacità dei database. La capacità viene regolata automaticamente in base alle necessità dall'applicazione. Ti vengono addebitate solo le risorse utilizzate dai cluster database. Quindi, Aurora Serverless v2 può aiutarti a rispettare il budget ed evitare di pagare per risorse informatiche che non utilizzi.

Questo tipo di automazione è particolarmente utile per database multi-tenant, database distribuiti, sistemi di sviluppo e di test e altri ambienti con carichi di lavoro fortemente variabili e non prevedibili.

Argomenti

- Aurora Serverless v2 casi d'uso
- Vantaggi di Aurora Serverless v2
- In che modo Aurora Serverless v2 funzionamento
- Requisiti e limitazioni per Aurora Serverless v2
- Creazione di un cluster DB che utilizza Aurora Serverless v2
- Gestione di Aurora Serverless v2 Cluster database
- Prestazioni e dimensionamento per Aurora Serverless v2
- Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2
- Migrazione verso Aurora Serverless v2

Aurora Serverless v2 casi d'uso

Aurora Serverless v2 supporta molti tipi di carichi di lavoro di database. Questi variano da ambienti di sviluppo e test, a siti Web e applicazioni con carichi di lavoro imprevedibili, fino alle applicazioni business-critical più esigenti che richiedono scalabilità e disponibilità elevate.

Aurora Serverless v2 è particolarmente utile per i seguenti casi d'uso:

- Carichi di lavoro variabili. Esegui carichi di lavoro con aumenti improvvisi e non prevedibili dell'attività. Un esempio può essere quello di un sito che offre informazioni sulla viabilità, che riscontra un forte aumento dell'attività quando inizia a piovere. Un altro è un sito di e-commerce che aumenta il proprio traffico durante un'offerta o una vendita promozionale. Con Aurora

Serverless v2, il database ridimensiona automaticamente la capacità per soddisfare le esigenze di picco di carico dell'applicazione e lo ridimensiona nuovamente al termine del picco di attività. Con Aurora Serverless v2, non è più necessario predisporre la capacità di picco o media. È possibile specificare un limite massimo di capacità per gestire la peggiore situazione possibile, che viene raggiunto solo in caso di necessità.

La granularità della scalabilità in Aurora Serverless v2 ti aiuta a far corrispondere la capacità alle esigenze del tuo database. Per un cluster con provisioning, il dimensionamento richiede l'aggiunta di un'istanza database completamente nuova. Per un Aurora Serverless v1 cluster, la scalabilità verticale richiede il raddoppio del numero di unità di capacità Aurora (ACUs) per il cluster, ad esempio da 16 a 32 o da 32 a 64. Al contrario, Aurora Serverless v2 può aggiungerne mezza ACU quando è necessaria solo un po' più di capacità. Può aggiungere 0,5, 1, 1,5, 2 o metà aggiuntiva, in ACUs base alla capacità aggiuntiva necessaria per gestire un aumento del carico di lavoro. Inoltre, può rimuovere 0,5, 1, 1,5, 2 o la metà aggiuntiva ACUs quando il carico di lavoro diminuisce e tale capacità non è più necessaria.

- Applicazioni multi-tenant: con Aurora Serverless v2, non è necessario gestire individualmente la capacità del database per ogni applicazione del parco applicazioni. Aurora Serverless v2 gestisce la capacità del database individuale per te.

  È possibile creare un cluster per ciascun tenant. Ciò ti permette di utilizzare funzionalità come la clonazione, il ripristino delle snapshot e i database globali Aurora per potenziare l'elevata disponibilità e il ripristino di emergenza in base alle esigenze di ciascun tenant.

  Ogni tenant potrebbe avere periodi di attività e inattività specifici, che variano a seconda dell'ora, del periodo dell'anno, degli eventi promozionali e così via. Ogni cluster è dotato di un'ampia gamma di capacità. In questo modo, dovrai affrontare costi minimi per le istanze database dei cluster con bassa attività. Tutti i cluster sono in grado di scalare rapidamente per gestire periodi di attività elevata.
- Nuove applicazioni. Supponiamo che desideri distribuire una nuova applicazione e hai dei dubbi riguardo alle dimensioni dell'istanza da utilizzare. Usando Aurora Serverless v2, è possibile configurare un cluster con una o più istanze DB e impostare la scalabilità automatica del database in base ai requisiti di capacità dell'applicazione.
- Applicazioni a uso misto: supponiamo di disporre di un'applicazione di elaborazione delle transazioni online (OLTP), ma che si verifichino periodicamente picchi nel traffico delle query. Specificando i livelli di promozione per Aurora Serverless v2 Istanze DB in un cluster, è possibile configurare il cluster in modo che le istanze DB Reader possano scalare indipendentemente dall'istanza DB di scrittura per gestire il carico aggiuntivo. Quando il picco di utilizzo diminuisce, le

istanze database di lettura si ridimensionano adeguandosi alla capacità dell'istanza database di scrittura.

- Pianificazione della capacità. Poniamo l'esempio che tu debba regolare la capacità del database o verificare la capacità ottimale del database in base al tuo carico di lavoro. Per farlo, dovresti modificare le classi di tutte le istanze database in un cluster. Con Aurora Serverless v2, puoi evitare questo sovraccarico amministrativo. È possibile determinare la capacità minima e massima adeguata eseguendo il carico di lavoro e verificando la scalabilità effettiva delle istanze database.

  È possibile modificare le istanze DB esistenti da provisioning a Aurora Serverless v2 o da Aurora Serverless v2 a rifornito. In questi casi, non è necessario creare un nuovo cluster o una nuova istanza database.

  Con un database globale Aurora, il bisogno di capacità dei cluster secondari potrebbe essere minore rispetto a quella del cluster primario. È possibile utilizzare… Aurora Serverless v2 istanze DB nei cluster secondari. In questo modo, la capacità del cluster può aumentare se una regione secondaria viene promossa e assume il carico di lavoro dell'applicazione.
- Sviluppo e test: oltre a eseguire le applicazioni più impegnative, puoi anche utilizzare Aurora Serverless v2 per ambienti di sviluppo e test. Con Aurora Serverless v2, è possibile creare istanze DB con una capacità minima ridotta anziché utilizzare classi di istanze DB db.t* dustable. È possibile impostare una capacità massima sufficientemente elevata da consentire a tali istanze database di eseguire carichi di lavoro considerevoli senza esaurire la memoria. Quando il database non è in uso, tutte le istanze database vengono ridotte per evitare costi inutili.

  > **Tip**
  >
  > Per renderlo comodo da usare Aurora Serverless v2 in ambienti di sviluppo e test, AWS Management Console fornisce la scorciatoia Easy create quando si crea un nuovo cluster. Se scegli l'opzione Dev/Test, Aurora crea un cluster con un Aurora Serverless v2 Un'istanza DB e un intervallo di capacità tipico di un sistema di sviluppo e test.

Utilizzo Aurora Serverless v2 per i carichi di lavoro predisposti esistenti

Supponiamo che tu disponga già di un'applicazione Aurora in esecuzione su un cluster con provisioning. Puoi verificare come funzionerebbe l'applicazione con Aurora Serverless v2 aggiungendone uno o più Aurora Serverless v2 istanze DB al cluster esistente come istanze DB di lettura. È possibile verificare la frequenza con cui le istanze di lettura scalano verso l'alto e verso il

basso. È possibile utilizzare il meccanismo di failover Aurora per promuovere un Aurora Serverless v2 Istanza DB per scrivere e verificare come gestisce il carico di lavoro di lettura/scrittura. Questo passaggio viene eseguito con tempi di inattività minimi e senza modificare l'endpoint utilizzato dalle applicazioni client. Per i dettagli sulla procedura per convertire i cluster esistenti in Aurora Serverless v2, consulta Migrazione verso Aurora Serverless v2.

Vantaggi di Aurora Serverless v2

Aurora Serverless v2 è destinato a carichi di lavoro variabili o «spinosi». A causa dei carichi di lavoro non prevedibili, potresti avere difficoltà a pianificare le modifiche della capacità del database. Potresti anche avere problemi ad apportare modifiche alla capacità abbastanza velocemente utilizzando meccanismi familiari come l'aggiunta di istanze DB o la modifica delle classi di istanze DB. Aurora Serverless v2 offre i seguenti vantaggi per questi casi d'uso:

- Gestione della capacità più semplice di quella fornita: Aurora Serverless v2 riduce gli sforzi necessari per pianificare le dimensioni delle istanze DB e ridimensionare le istanze DB al variare del carico di lavoro. Inoltre, riduce lo sforzo per mantenere la capacità coerente per tutte le istanze database in un cluster.
- Scalabilità più rapida e semplice durante i periodi di elevata attività: Aurora Serverless v2 ridimensiona la capacità di calcolo e di memoria in base alle esigenze, senza interrompere le transazioni con i clienti o il carico di lavoro complessivo. La possibilità di utilizzare istanze DB Reader con Aurora Serverless v2 ti aiuta a sfruttare il ridimensionamento orizzontale oltre al ridimensionamento verticale. La possibilità di utilizzare i database globali di Aurora significa che puoi diffondere Aurora Serverless v2 leggi il carico di lavoro su più pagine. Regioni AWS Questa funzionalità offre una praticità maggiore rispetto ai meccanismi di dimensionamento dei cluster con provisioning. È anche più veloce e più granulare rispetto alle funzionalità di scalabilità di Aurora Serverless v1.
- Conveniente sotto il profilo dei costi durante i periodi di scarsa attività: Aurora Serverless v2 ti aiuta a evitare l'overprovisioning delle tue istanze DB. Aurora Serverless v2 aggiunge risorse in incrementi granulari quando le istanze DB aumentano. Paghi solo per le risorse di database che utilizzi. Aurora Serverless v2 l'utilizzo delle risorse viene misurato su base al secondo. In questo modo, quando un'istanza database scala verso il basso, la riduzione dell'utilizzo delle risorse viene registrata immediatamente.
- Maggiore parità di funzionalità con provisioned: puoi utilizzare molte funzionalità di Aurora con Aurora Serverless v2 che non sono disponibili per Aurora Serverless v1. Ad esempio, con Aurora Serverless v2 è possibile utilizzare istanze Reader DB, database globali, AWS Identity and Access

Management (IAM) autenticazione del database e Performance Insights. È inoltre possibile utilizzare molti più parametri di configurazione rispetto a Aurora Serverless v1.

In particolare, con Aurora Serverless v2 puoi sfruttare le seguenti funzionalità dei cluster predisposti:

- Istanze Reader DB: Aurora Serverless v2 può sfruttare le istanze Reader DB per scalare orizzontalmente. Quando un cluster contiene una o più istanze database di lettura, può eseguire immediatamente il failover se si verificano problemi con l'istanza database di scrittura. Questa funzionalità non è disponibile con Aurora Serverless v1.
- Cluster Multi-AZ: è possibile distribuire Aurora Serverless v2 Istanze DB di un cluster su più zone di disponibilità (). AZs La configurazione di un cluster Multi-AZ contribuisce a garantire la continuità aziendale anche nel raro caso che si verifichino problemi che coinvolgono l'intera AZ. Questa funzionalità non è disponibile con Aurora Serverless v1.
- Database globali: puoi usare Aurora Serverless v2 in combinazione con i database globali Aurora per creare ulteriori copie di sola lettura del cluster in altri paesi Regioni AWS per scopi di disaster recovery.
- RDSProxy: puoi utilizzare Amazon RDS Proxy per consentire alle tue applicazioni di raggruppare e condividere connessioni al database per migliorarne la scalabilità.

- Scalabilità più rapida, più granulare e meno dirompente rispetto a Aurora Serverless v1 – Aurora Serverless v2 può scalare verso l'alto e verso il basso più velocemente. La scalabilità può modificare la capacità di appena 0,5ACUs, anziché raddoppiare o dimezzare il numero di. ACUs Il dimensionamento avviene in genere senza alcuna interruzione dell'elaborazione. La scalabilità non implica un evento di cui devi essere a conoscenza, come nel caso di Aurora Serverless v1. La scalabilità può avvenire mentre gli SQL estratti conto sono in esecuzione e le transazioni sono aperte, senza la necessità di attendere un momento di silenzio.

In che modo Aurora Serverless v2 funzionamento

La seguente panoramica descrive come Aurora Serverless v2 funziona.

Argomenti

- Aurora Serverless v2 resoconto
- Configurazioni per i cluster Aurora DB
- Aurora Serverless v2 capacità
- Aurora Serverless v2 scalabilità

- Aurora Serverless v2 e alta disponibilità
- Aurora Serverless v2 e archiviazione
- Parametri di configurazione per i cluster Aurora

Aurora Serverless v2 resoconto

Amazon Aurora Serverless v2 è adatto per i carichi di lavoro più impegnativi e altamente variabili. Ad esempio, l'uso del database potrebbe essere pesante per un breve periodo di tempo, seguito da lunghi periodi di attività leggera o nessuna attività. Alcuni esempi sono siti web per la vendita al dettaglio, i giochi o sportivi con eventi promozionali periodici e database in grado di generare report quando necessario. Altri sono ambienti di sviluppo e test e nuove applicazioni in cui l'utilizzo potrebbe aumentare rapidamente. In casi come questi e molti altri, non sempre è possibile configurare correttamente la capacità in anticipo con il modello fornito. Ciò può anche comportare costi più elevati se si esegue il provisioning eccessivo e si dispone di capacità che poi non viene utilizzata.

Al contrario, i cluster di Aurora con provisioning sono adatti per carichi di lavoro stabili. Con i cluster con provisioning, puoi scegliere una classe di istanza database con una quantità predefinita di memoria, potenza della CPU, larghezza di banda I/O e così via. Se il carico di lavoro cambia, modifichi manualmente la classe di istanza dello scrittore e dei lettori. Il modello soggetto a provisioning funziona bene quando è possibile regolare la capacità in anticipo rispetto ai modelli di consumo previsti ed è accettabile soffrire di brevi interruzioni mentre si modifica la classe di istanza dello scrittore e dei lettori all'interno del cluster.

Aurora Serverless v2 è progettato da zero per supportare cluster DB serverless che sono immediatamente scalabili. Aurora Serverless v2 è progettato per fornire lo stesso grado di sicurezza e isolamento degli autori e dei lettori predisposti. Questi aspetti sono cruciali negli ambienti cloud serverless multitenant. Il meccanismo di dimensionamento dinamico impone un overhead molto ridotto in modo da poter rispondere rapidamente alle modifiche del carico di lavoro del database. È anche abbastanza potente da soddisfare i considerevoli aumenti della domanda di elaborazione.

Utilizzando Aurora Serverless v2, puoi creare un cluster Aurora DB senza essere vincolato a una capacità di database specifica per ogni scrittore e lettore. Puoi specificare l'intervallo minimo e massimo per la capacità. Aurora scala ciascuno Aurora Serverless v2 scrittore o lettore nel cluster entro tale intervallo di capacità. Utilizzando un cluster Multi-AZ in cui ogni scrittore o lettore può dimensionarsi dinamicamente puoi sfruttare la scalabilità dinamica e l'elevata disponibilità.

Aurora Serverless v2 ridimensiona automaticamente le risorse del database in base alle specifiche di capacità minima e massima. La scalabilità è rapida perché la maggior parte delle operazioni legate

agli eventi di dimensionamento mantiene lo scrittore o il lettore sullo stesso host. Nei rari casi in cui un Aurora Serverless v2 lo scrittore o il lettore viene spostato da un host all'altro, Aurora Serverless v2 gestisce automaticamente le connessioni. Non è necessario modificare il codice dell'applicazione client del database o le stringhe di connessione al database.

Con Aurora Serverless v2, come per i cluster forniti, la capacità di archiviazione e la capacità di elaborazione sono separate. Quando ci riferiamo a Aurora Serverless v2 capacità e scalabilità, è sempre la capacità di elaborazione ad aumentare o diminuire. Pertanto, il cluster può contenere molti terabyte di dati anche quando la CPU e la capacità di memoria si dimensionano verso il basso.

Anziché effettuare il provisioning e gestire i server di database, puoi indicare la capacità del database. Per informazioni dettagliate su Aurora Serverless v2 capacità, vedereAurora Serverless v2 capacità. La capacità effettiva di ciascuno Aurora Serverless v2 scrittore o lettore varia nel tempo, a seconda del carico di lavoro. Per i dettagli su questi meccanismi, consulta Aurora Serverless v2 scalabilità.

> **Important**
>
> Con Aurora Serverless v1, il cluster dispone di un'unica misura della capacità di elaborazione che può essere scalata tra i valori di capacità minima e massima. Con Aurora Serverless v2, il cluster può contenere lettori oltre allo scrittore. Ciascuno Aurora Serverless v2 scrittore e lettore possono scalare tra i valori di capacità minima e massima. Quindi, la capacità totale del tuo Aurora Serverless v2 il cluster dipende sia dall'intervallo di capacità definito per il cluster DB sia dal numero di scrittori e lettori presenti nel cluster. In qualsiasi momento specifico, ti viene addebitato solo il Aurora Serverless v2 capacità che viene utilizzata attivamente nel cluster Aurora DB.

Configurazioni per i cluster Aurora DB

Per ciascuno dei tuoi cluster Aurora DB, puoi scegliere qualsiasi combinazione di Aurora Serverless v2 capacità, capacità fornita o entrambe.

È possibile configurare un cluster che contenga entrambi Aurora Serverless v2 e capacità assegnata, denominata cluster a configurazione mista. Ad esempio, si supponga di aver bisogno di una capacità di lettura/scrittura superiore a quella disponibile per un Aurora Serverless v2 scrittore. In questo caso puoi configurare il cluster con uno scrittore con provisioning di dimensioni molto ampie. In tal caso, puoi ancora usare Aurora Serverless v2 per i lettori. Oppure supponi che il carico di lavoro in scrittura

per il cluster vari ma che il carico di lavoro in lettura sia costante. In questo caso, puoi configurare il tuo cluster con un Aurora Serverless v2 scrittore e uno o più lettori autorizzati.

È inoltre possibile configurare un cluster DB in cui tutta la capacità è gestita da Aurora Serverless v2. Per fare ciò, puoi creare un nuovo cluster e utilizzare Aurora Serverless v2 dall'inizio. Oppure puoi sostituire tutta la capacità fornita in un cluster esistente con Aurora Serverless v2. Ad esempio, alcuni percorsi di aggiornamento da versioni precedenti del motore richiedono di iniziare con un writer fornito e sostituirlo con un Aurora Serverless v2 scrittore. Per le procedure per creare un nuovo cluster DB con Aurora Serverless v2 o per passare a un cluster DB esistente Aurora Serverless v2, vedi Creare un Aurora Serverless v2 cluster di database ePassaggio da un cluster fornito a Aurora Serverless v2.

Se non lo usi Aurora Serverless v2 in un cluster DB, viene eseguito il provisioning di tutti gli scrittori e i lettori del cluster DB. Questo è il tipo di cluster di database più vecchio e più comune con cui la maggior parte degli utenti ha familiarità. In effetti, prima Aurora Serverless, non c'era un nome speciale per questo tipo di cluster Aurora DB. La capacità fornita è costante. Le tariffe sono relativamente semplici da prevedere. Tuttavia, è necessario prevedere in anticipo quanta capacità è necessaria. In alcuni casi le previsioni potrebbero essere imprecise o le esigenze di capacità potrebbero cambiare. In questi casi, il cluster di database può rivelarsi soggetto a provisioning in modo insufficiente (più lento di quello desiderato) o provisioning in modo eccessivo (più costoso di quello desiderato).

Aurora Serverless v2 capacità

L'unità di misura per Aurora Serverless v2 è l'unità di capacità Aurora (ACU). Aurora Serverless v2 la capacità non è legata alle classi di istanze DB utilizzate per i cluster assegnati.

Ogni ACU è una combinazione di circa 2 gigabyte (GiB) di memoria, CPU corrispondente e rete. Puoi specificare l'intervallo di capacità del database utilizzando questa unità di misura. I parametri `ServerlessDatabaseCapacity` e `ACUUtilization` consentono di determinare la capacità effettivamente utilizzata dal database e se tale capacità rientra nell'intervallo specificato.

In qualsiasi momento, ogni Aurora Serverless v2 Lo scrittore o lettore DB ha una capacità. La capacità è rappresentata come un numero a virgola mobile che rappresenta. ACUs La capacità aumenta o diminuisce ogni volta che lo scrittore o il lettore si dimensionano. Questo valore viene misurato ogni secondo. Per ogni cluster DB in cui intendi utilizzare Aurora Serverless v2, si definisce un intervallo di capacità: i valori di capacità minima e massima di ciascuno Aurora Serverless v2 lo scrittore o il lettore possono passare da uno all'altro. L'intervallo di capacità è lo stesso per ciascuno

Aurora Serverless v2 scrittore o lettore in un cluster DB. Ciascuno Aurora Serverless v2 lo scrittore o il lettore hanno le proprie capacità, che rientrano in questa gamma.

La tabella seguente mostra le Aurora Serverless v2 intervalli di capacità supportati per Aurora MySQL e Aurora PostgreSQL.

ACUsIntervallo di capacità ()	Versioni supportate da Aurora MySQL	Versioni supportate da Aurora PostgreSQL
0,5-128	3.02.0 e versioni successive	13.6 e versioni successive, 14.3 e versioni successive, 15.2 e versioni successive, 16.1 e versioni successive
0,5—256	3.06.0 e versioni successive	13.13 e versioni successive, 14.10 e versioni successive, 15.5 e versioni successive, 16.1 e versioni successive
0—256	3.08.0 e versioni successive	13.15 e versioni successive, 14.12 e versioni successive, 15.7 e versioni successive, 16.3 e versioni successive

Il più piccolo Aurora Serverless v2 la capacità che è possibile definire è 0 ACUs, per Aurora Serverless v2 versioni che supportano la funzione di pausa automatica. Puoi specificare un numero superiore se inferiore o uguale al valore di capacità massima. L'impostazione della capacità minima su un numero ridotto consente ai cluster di database con carichi leggeri di consumare risorse di calcolo minime. Allo stesso tempo, rimangono pronti ad accettare immediatamente le connessioni e a dimensionarsi quando diventano impegnati.

Si consiglia di impostare il minimo su un valore che consenta a ciascuno scrittore o lettore del database DB di mantenere il set di lavoro dell'applicazione nel buffer pool. In questo modo, il contenuto del buffer pool non viene scartato durante i periodi di inattività. Per tutte le considerazioni nella scelta del valore massimo della capacità, consulta Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster. Per tutte le considerazioni nella scelta del valore massimo

della capacità, consulta Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster.

A seconda di come si configurano i lettori in un'implementazione Multi-AZ, le loro capacità possono essere legate alla capacità dello scrittore o indipendentemente. Per i dettagli su come eseguire queste operazioni, consulta Aurora Serverless v2 scalabilità.

Monitoraggio Aurora Serverless v2 prevede la misurazione dei valori di capacità dello scrittore e dei lettori nel cluster DB nel tempo. Se il database non viene si dimensiona verso il basso fino alla capacità minima, puoi intraprendere azioni come la regolazione del minimo e l'ottimizzazione dell'applicazione database. Se il database raggiunge costantemente la sua capacità massima, puoi intraprendere operazioni come l'aumento di tale vincolo. Puoi inoltre ottimizzare l'applicazione di database e distribuire il carico di query su più lettori.

Le spese per Aurora Serverless v2 la capacità è misurata in termini di ore ACU. Per informazioni su come Aurora Serverless v2 i costi sono calcolati, consulta la pagina dei prezzi di Aurora.

Supponiamo che il numero totale di scrittori e lettori del cluster sia. *n* In tal caso, il cluster consuma approssimativamente *n* x *minimum ACUs* quando non si eseguono operazioni di database. Aurora stessa potrebbe eseguire operazioni di monitoraggio o manutenzione che causano una piccola quantità di carico. Nel momento in cui il database è in esecuzione a piena capacità, quel cluster non consuma più di *n* x *maximum ACUs*.

Per ulteriori informazioni sulla scelta dei valori minimi e massimi appropriati di ACU, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora. I valori ACU minimi e massimi specificati influiscono anche sul funzionamento di alcuni parametri di configurazione di Aurora. Aurora Serverless v2. Per informazioni dettagliate sull'interazione tra l'intervallo di capacità e i parametri di configurazione, vedereUso di gruppi di parametri per Aurora Serverless v2.

Aurora Serverless v2 scalabilità

Per ogni Aurora Serverless v2 scrittore o lettore, Aurora monitora continuamente l'utilizzo di risorse come CPU, memoria e rete. Queste misurazioni sono chiamate collettivamente carico. Il carico include le operazioni di database eseguite dall'applicazione. Include anche l'elaborazione in background per il server di database e le attività amministrative di Aurora. Quando la capacità è limitata da uno di questi fattori, Aurora Serverless v2 aumenta. Aurora Serverless v2 si espande inoltre quando rileva problemi di prestazioni che può risolvere in tal modo. È possibile monitorare l'utilizzo delle risorse e il modo in cui influisce Aurora Serverless v2 scalabilità utilizzando le

procedure di e. CloudWatch Parametri Amazon importanti per Aurora Serverless v2 Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni

Il carico può variare tra lo scrittore e i lettori del cluster di database. Lo scrittore gestisce tutte le istruzioni DDL (Data Definition Language), come `CREATE TABLE`,`ALTER TABLE` e `DROP TABLE`. Lo scrittore gestisce anche tutte le istruzioni DML (Data Manipulation Language), come ad esempio `INSERT` e `UPDATE`. I lettori possono elaborare istruzioni di sola lettura, come ad esempio le query `SELECT`.

Il ridimensionamento è l'operazione che aumenta o diminuisce Aurora Serverless v2 capacità del database. Con Aurora Serverless v2, ogni scrittore e lettore ha il proprio valore di capacità attuale, misurato in ACUs. Aurora Serverless v2 ridimensiona uno scrittore o un lettore fino a una capacità superiore quando la sua capacità attuale è troppo bassa per gestire il carico. Ridimensiona lo scrittore o il lettore a una capacità inferiore quando la sua capacità corrente è superiore a quella necessaria.

A differenza Aurora Serverless v1, scalabile raddoppiando la capacità ogni volta che il cluster DB raggiunge una soglia, Aurora Serverless v2 può aumentare la capacità in modo incrementale. Quando la richiesta di carico di lavoro inizia a raggiungere l'attuale capacità di database di uno scrittore o di un lettore, Aurora Serverless v2 aumenta il numero di ACUs per quello scrittore o lettore. Aurora Serverless v2 ridimensiona la capacità negli incrementi richiesti per fornire le migliori prestazioni per le risorse consumate. Il ridimensionamento avviene con incrementi fino a 0,5. ACUs Maggiore è la capacità attuale, maggiore è l'incremento nel dimensionamento e quindi più velocemente può essere garantito il dimensionamento.

Perché Aurora Serverless v2 il ridimensionamento è così frequente, granulare e senza interruzioni che non causa eventi discreti come AWS Management Console Aurora Serverless v1 fa. Puoi invece misurare i CloudWatch parametri di Amazon come `ServerlessDatabaseCapacity` e `ACUUtilization` e tenere traccia dei loro valori minimi, massimi e medi nel tempo. Per maggiori informazioni sui parametri di Aurora, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora. Per suggerimenti sul monitoraggio Aurora Serverless v2, consulta CloudWatch Parametri Amazon importanti per Aurora Serverless v2.

Puoi scegliere di creare un dimensionamento del lettore in modo contemporaneo allo scrittore associato o indipendentemente da questo. Tale obiettivo si raggiunge specificando il livello di promozione per quel lettore.

- I lettori associati ai livelli di promozione 0 e 1 si dimensionano contemporaneamente allo scrittore. Questo comportamento di dimensionamento rende i lettori con livelli prioritari 0 e 1 ideali per la

disponibilità. Questo perché sono sempre dimensionati alla capacità giusta per assumere il carico di lavoro dallo scrittore in caso di failover.

- I lettori nei livelli di promozione da 2 a 15 si dimensionano indipendentemente dallo scrittore. Ogni lettore si mantiene all'interno dei valori ACU minimo e massimo specificati per il cluster. Quando un lettore si dimensiona indipendentemente dallo scrittore del database associato, può diventare inattivo e dimensionarsi verso il basso mentre lo scrittore continua a elaborare un volume elevato di transazioni. Se nessun altro lettore è disponibile in livelli di promozione inferiori rimane ancora disponibile come target di failover. Tuttavia, se viene promosso al ruolo di scrittore, potrebbe essere necessario un dimensionamento verso l'alto per gestire l'intero carico di lavoro dello scrittore.

Per i dettagli sui livelli di promozione, consulta Scelta del livello di promozione per un Aurora Serverless v2 reader.

Le nozioni di punti di scala e i periodi di timeout associati da Aurora Serverless v1 non si applicano in Aurora Serverless v2. Aurora Serverless v2 il ridimensionamento può avvenire mentre le connessioni al database sono aperte, mentre le transazioni SQL sono in corso, mentre le tabelle sono bloccate e mentre le tabelle temporanee sono in uso. Aurora Serverless v2 non aspetta che arrivi un punto di quiete per iniziare la scalabilità. Il dimensionamento non interrompe le operazioni del database in corso.

Se il carico di lavoro richiede una capacità di lettura superiore a quella disponibile con un singolo scrittore e un singolo lettore, puoi aggiungerne più di uno Aurora Serverless v2 lettori al cluster. Ciascuno Aurora Serverless v2 reader può scalare entro l'intervallo dei valori di capacità minima e massima specificati per il cluster DB. Puoi utilizzare l'endpoint di lettura del cluster per gestire le sessioni di sola lettura attraverso i lettori e ridurre il carico sullo scrittore.

Se Aurora Serverless v2 esegue il ridimensionamento e la velocità con cui viene eseguito il ridimensionamento una volta avviato dipende anche dalle impostazioni ACU minime e massime per il cluster. Inoltre, dipendono dal fatto che un lettore sia configurato per dimensionarsi contemporaneamente allo scrittore o indipendentemente da esso. Per informazioni dettagliate sui fattori che influiscono Aurora Serverless v2 ridimensionamento, vederePrestazioni e dimensionamento per Aurora Serverless v2.

Ridimensionamento a zero

Nelle versioni recenti di Aurora MySQL e Aurora PostgreSQL, Aurora Serverless v2 scrittori e lettori possono scalare fino a zero. ACUs Ci riferiamo a questa funzionalità come pausa e ripresa

automatiche o pausa automatica. È possibile scegliere se consentire questo comportamento specificando un valore zero o diverso da zero per la capacità minima. Puoi anche scegliere quanto tempo aspettare prima di un Aurora Serverless v2 pause di istanza. Per informazioni sulle versioni dotate di questa funzionalità, vedereAurora Serverless v2 capacità. Per informazioni su come utilizzarlo in modo efficace, vedereRidimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2.

Nelle versioni precedenti di Aurora MySQL e Aurora PostgreSQL, inattiva Aurora Serverless v2 gli autori e i lettori possono ridimensionarsi fino al valore ACU minimo specificato per il cluster, ma non fino a zero. ACUs In tal caso, zero non ACUs è disponibile come scelta quando si imposta l'intervallo di capacità. Questo comportamento è diverso da Aurora Serverless v1, che può interrompersi dopo un periodo di inattività, ma che poi impiega del tempo prima che riprenda quando si apre una nuova connessione.

Quando il tuo database cluster con Aurora Serverless v2 la capacità non è necessaria per un certo periodo di tempo, puoi anche arrestare e avviare l'intero cluster, come con i cluster DB di cui è stato effettuato il provisioning. Questa tecnica è più appropriata per i sistemi di sviluppo e test, dove potrebbero non essere necessari per molte ore consecutive e la velocità di ripristino del cluster non è fondamentale. La funzionalità stop/start del cluster è disponibile per tutti Aurora Serverless v2 versioni. Per ulteriori informazioni su questa funzionalità, vedereAvvio e arresto di un cluster di database Amazon Aurora.

Aurora Serverless v2 e alta disponibilità

Il modo per garantire un'elevata disponibilità di un cluster Aurora DB consiste nel renderlo un cluster di database Multi-AZ. Un cluster di database Aurora Multi-AZ garantisce capacità di calcolo disponibile in ogni momento in più di una zona di disponibilità (AZ). Tale configurazione mantiene il database attivo e funzionante anche in caso di rilevanti malfunzionamenti. Aurora esegue un failover automatico in caso di problemi che riguardano lo scrittore o persino l'intera AZ. Con Aurora Serverless v2, puoi scegliere la scalabilità verso l'alto e verso il basso della capacità di elaborazione in standby insieme alla capacità dello scrittore. In questo modo, la capacità di calcolo nella seconda AZ è pronta a rilevare il carico di lavoro corrente in qualsiasi momento. Allo stesso tempo, la capacità di elaborazione complessiva AZs può essere ridotta quando il database è inattivo. Per informazioni dettagliate sul funzionamento di Aurora Regioni AWS e sulle zone di disponibilità, consulta. Architetture ad alta disponibilità per istanze database Aurora

Il Aurora Serverless v2 La funzionalità Multi-AZ utilizza lettori oltre allo scrittore. Il supporto per i lettori è una novità per Aurora Serverless v2 rispetto a Aurora Serverless v1. Puoi aggiungere fino a 15 Aurora Serverless v2 lettori distribuiti su 3 AZs su un cluster Aurora DB.

Per le applicazioni aziendali critiche che devono rimanere disponibili anche in caso di problemi che interessano l'intero cluster o l'intera AWS regione, puoi configurare un database globale Aurora. È possibile utilizzare… Aurora Serverless v2 capacità nei cluster secondari in modo che siano pronti a subentrare durante il disaster recovery. Possono anche dimensionarsi verso il basso quando il database non è impegnato. Per informazioni dettagliate sui database globali di Aurora, consulta Utilizzo del database globale Amazon Aurora.

Aurora Serverless v2 funziona in modo analogo a quanto previsto per il failover e altre funzionalità di alta disponibilità. Per ulteriori informazioni, consulta Elevata disponibilità di Amazon Aurora.

Supponiamo di voler garantire la massima disponibilità per i Aurora Serverless v2 grappolo. Puoi un lettore in aggiunta allo scrittore. Se assegni il lettore al livello di promozione 0 o 1, qualsiasi dimensionamento avvenga per lo scrittore esso viene replicato anche per il lettore. In questo modo, un lettore con capacità identica è sempre pronto a prendere il posto dello scrittore in caso di failover.

Supponi di voler eseguire report trimestrali per la tua azienda nello stesso momento in cui il cluster continua a elaborare le transazioni. Se aggiungi un Aurora Serverless v2 è possibile collegare il lettore al cluster e assegnarlo a un livello di promozione compreso tra 2 e 15, è possibile connettersi direttamente a quel lettore per eseguire i report. A seconda di quanto le query di reporting sono esigenti in termini di memoria e di utilizzo della CPU, il lettore può dimensionarsi verso l'alto per adattarsi al carico di lavoro. Può successivamente dimensionarsi di nuovo verso il basso una volta terminata la generazione dei report.

Aurora Serverless v2 e archiviazione

Lo storage per ogni cluster Aurora DB è composto da sei copie di tutti i dati, distribuite su tre. AZs Questa replica dei dati integrata si applica indipendentemente dal fatto che il cluster di database includa dei lettori in aggiunta allo scrittore. In questo modo i dati sono al sicuro anche da problemi che influiscono sulla capacità di calcolo del cluster.

Aurora Serverless v2 lo storage ha le stesse caratteristiche di affidabilità e durabilità descritte inArchiviazione Amazon Aurora. Questo perché lo storage per i cluster Aurora DB funziona allo stesso modo indipendentemente dalla capacità di elaborazione utilizzata Aurora Serverless v2 o fornito.

Parametri di configurazione per i cluster Aurora

È possibile modificare tutti gli stessi parametri di configurazione del cluster e del database per i cluster con Aurora Serverless v2 capacità come per i cluster DB assegnati. Tuttavia, alcuni parametri relativi alla capacità vengono gestiti in modo diverso per Aurora Serverless v2. In un cluster a configurazione mista, i valori dei parametri specificati per tali parametri relativi alla capacità si applicano ancora a tutti gli scrittori e i lettori forniti.

Quasi tutti i parametri funzionano allo stesso modo per Aurora Serverless v2 scrittori e lettori per quanto riguarda quelli predisposti. Le eccezioni riguardano alcuni parametri che Aurora regola automaticamente durante il dimensionamento e alcuni parametri che Aurora mantiene a valori fissi dipendenti dall'impostazione della capacità massima.

Ad esempio, la quantità di memoria riservata alla cache del buffer aumenta man mano che uno scrittore o un lettore si dimensionano verso l'alto e diminuisce man mano che si dimensionano verso il basso. In questo modo, la memoria può essere rilasciata quando il database non è impegnato. Al contrario, Aurora imposta automaticamente il numero massimo di connessioni su un valore appropriato in base all'impostazione della capacità massima. In questo modo, le connessioni attive non vengono interrotte se il carico diminuisce e Aurora Serverless v2 si ridimensiona. Per informazioni su come Aurora Serverless v2 gestisce parametri specifici, vedereUso di gruppi di parametri per Aurora Serverless v2.

Requisiti e limitazioni per Aurora Serverless v2

Quando crei un cluster in cui intendi utilizzare Aurora Serverless v2 Per le istanze DB, prestate attenzione ai seguenti requisiti e limitazioni.

Argomenti

- Disponibilità di regioni e versioni
- Cluster che utilizzano Aurora Serverless v2 deve avere un intervallo di capacità specificato
- Alcune funzionalità fornite non sono supportate in Aurora Serverless v2
- Medio Aurora Serverless v2 gli aspetti sono diversi da Aurora Serverless v1

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni

con Aurora e Aurora Serverless v2, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v2.

L'esempio seguente mostra i AWS CLI comandi per confermare i valori esatti del motore DB che è possibile utilizzare con Aurora Serverless v2 per uno specifico Regione AWS. Il `--db-instance-class` parametro per Aurora Serverless v2 è sempre`db.serverless`. Il parametro `--engine` può essere `aurora-mysql` o `aurora-postgresql`. Sostituisci i valori `--region` e `--engine` appropriati per verificare i valori `--engine-version` che puoi utilizzare. Se il comando non produce alcun output, Aurora Serverless v2 non è disponibile per quella combinazione Regione AWS di motore DB.

```
aws rds describe-orderable-db-instance-options --engine aurora-mysql --db-instance-
class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text

aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text
```

Cluster che utilizzano Aurora Serverless v2 deve avere un intervallo di capacità specificato

Un cluster Aurora deve disporre di un attributo `ServerlessV2ScalingConfiguration` prima di poter aggiungere eventuali istanze database che utilizzano la classe dell'istanza database `db.serverless`. Questo attributo specifica l'intervallo di capacità. Aurora Serverless v2 la capacità varia da un minimo di 0 unità di capacità Aurora (ACU) a un massimo di 256 ACUs, con incrementi di 0,5 ACU. Il valore minimo consentito dipende dalla versione di Aurora. Ogni ACU fornisce l'equivalente di circa 2 gibibyte (GiB) di RAM, CPU corrispondente e rete. Per informazioni dettagliate su come Aurora Serverless v2 utilizza le impostazioni dell'intervallo di capacità, vedereIn che modo Aurora Serverless v2 funzionamento.

Per gli intervalli di capacità consentiti per le varie versioni del motore DB, vedereAurora Serverless v2 capacità.

È possibile specificare i valori ACU minimo e massimo in fase di AWS Management Console creazione di un cluster e associati Aurora Serverless v2 Istanza DB. È anche possibile specificare l'opzione `--serverless-v2-scaling-configuration` nella AWS CLI. In alternativa, puoi

specificare il parametro `ServerlessV2ScalingConfiguration` con l'API Amazon RDS. Puoi specificare questo attributo quando crei un cluster o modifichi un cluster esistente. Per le procedure di impostazione dell'intervallo di capacità, consulta Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster. Per una discussione dettagliata su come scegliere i valori di capacità minima e massima e su come tali impostazioni influiscono su alcuni parametri del database, vedi Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora.

Alcune funzionalità fornite non sono supportate in Aurora Serverless v2

Le seguenti funzionalità delle istanze DB fornite da Aurora attualmente non sono disponibili per Amazon Aurora Serverless v2:

- Flussi di attività di database (DAS)
- Gestione della cache del cluster in Aurora PostgreSQL. Il parametro `apg_ccm_enabled` di configurazione non si applica a Aurora Serverless v2 Istanze DB.

Alcune funzionalità di Aurora funzionano con Aurora Serverless v2, ma potrebbe causare problemi se l'intervallo di capacità è inferiore a quello necessario per i requisiti di memoria per tali funzionalità con il carico di lavoro specifico. In tal caso, è possibile che il database non funzioni come al solito o che si verifichino out-of-memory errori. Per consigli sull'impostazione dell'intervallo di capacità appropriato, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora. Per informazioni sulla risoluzione dei problemi nel caso in cui il database riscontri out-of-memory errori dovuti a un intervallo di capacità non configurato correttamente, vedere. Evitare gli errori out-of-memory

Il dimensionamento automatico in Aurora non è supportato. Questo tipo di scalabilità aggiunge nuovi lettori per gestire carichi di lavoro aggiuntivi ad alta intensità di lettura, in base all'utilizzo della CPU. Tuttavia, la scalabilità basata sull'utilizzo della CPU non è significativa per Aurora Serverless v2. In alternativa, puoi creare Aurora Serverless v2 leggi in anticipo le istanze DB e lasciale ridimensionate a bassa capacità. È un modo più rapido e meno dirompente per scalare la capacità di lettura di un cluster rispetto all'aggiunta dinamica di nuove istanze database.

Medio Aurora Serverless v2 gli aspetti sono diversi da Aurora Serverless v1

Se sei un Aurora Serverless v1 utente e questa è la prima volta che lo usi Aurora Serverless v2, consulta le differenze tra Aurora Serverless v2 e Aurora Serverless v1 requisiti per comprendere in che modo i requisiti sono diversi tra Aurora Serverless v1 e Aurora Serverless v2.

Creazione di un cluster DB che utilizza Aurora Serverless v2

Per creare un cluster Aurora in cui puoi aggiungere Aurora Serverless v2 Le istanze DB, seguite la stessa procedura descritta in. Creazione di un cluster database Amazon Aurora Con Aurora Serverless v2, i cluster sono intercambiabili con i cluster forniti. È possibile disporre di cluster utilizzati da alcune istanze DB Aurora Serverless v2 e alcune istanze DB vengono fornite.

Argomenti

- Impostazioni per Aurora Serverless v2 Cluster database
- Creare un Aurora Serverless v2 cluster di database
- Creare un Aurora Serverless v2 istanza Writer DB

Impostazioni per Aurora Serverless v2 Cluster database

Assicurati che le impostazioni iniziali del cluster soddisfino i requisiti elencati inRequisiti e limitazioni per Aurora Serverless v2. Specificate le seguenti impostazioni per assicurarvi di poterle aggiungere Aurora Serverless v2 Istanze DB al cluster:

Regione AWS

Crea il cluster in un luogo Regione AWS Aurora Serverless v2 Le istanze DB sono disponibili. Per i dettagli sulle regioni disponibili, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v2.

DB engine version (Versione motore del database

Scegli una versione del motore compatibile con Aurora Serverless v2. Per informazioni sul Aurora Serverless v2 requisiti di versione, vedereRequisiti e limitazioni per Aurora Serverless v2.

DB instance class (Classe istanza database)

Se crei un cluster utilizzando AWS Management Console, scegli contemporaneamente la classe di istanza DB per l'istanza DB writer. Scegli Serverless come classe di istanza database. Quando sceglie questa classe di istanza database, devi specificare anche l'intervallo di capacità per l'istanza database di scrittura. Lo stesso intervallo di capacità si applica a tutte le altre Aurora Serverless v2 Istanze DB che aggiungi a quel cluster.

Se non vedi l'opzione Serverless per la classe di istanze DB, assicurati di aver scelto una versione del motore DB supportata per. Regioni supportate e motori Aurora DB per Aurora Serverless v2

Quando utilizzi Amazon AWS CLI o Amazon RDSAPI, il parametro che specifichi per la classe dell'istanza DB è`db.serverless`.

Intervallo di capacità

Inserisci i valori minimo e massimo dell'unità di capacità Aurora (ACU) che si applicano a tutte le istanze DB del cluster. Questa opzione è disponibile sia nella pagina Crea cluster che nella pagina Aggiungi lettore della console quando scegli Serverless come classe dell'istanza database.

Per gli intervalli di capacità consentiti per le varie versioni del motore DB, consulta. Aurora Serverless v2 capacità

Se non vedi i ACU campi minimo e massimo, assicurati di aver scelto la classe di istanza DB Serverless per l'istanza Writer DB.

Se inizialmente crei il cluster con un'istanza DB fornita, non specifichi il minimo e il massimo. ACUs In tal caso è possibile modificare il cluster in seguito per aggiungere tale impostazione. Puoi anche aggiungere un Aurora Serverless v2 istanza Reader DB al cluster. Puoi specificare l'intervallo di capacità come parte del processo.

Finché non specifichi l'intervallo di capacità per il cluster, non puoi aggiungerne Aurora Serverless v2 istanze DB al cluster utilizzando AWS CLI o RDSAPI. Se provi ad aggiungere un Aurora Serverless v2 Istanza DB, viene visualizzato un errore. Nelle procedure AWS CLI o nelle RDS API procedure, l'intervallo di capacità è rappresentato dall'`ServerlessV2ScalingConfiguration`attributo.

Per i cluster contenenti più di un'istanza DB Reader, la priorità di failover di ciascuna Aurora Serverless v2 L'istanza Reader DB svolge un ruolo importante nella scalabilità verso l'alto e verso il basso di tale istanza DB. Non è possibile specificare la priorità nella fase iniziale di creazione del cluster. Tieni presente questa proprietà quando al cluster aggiungi una seconda istanza database di lettura o altre istanze di questo tipo. Per ulteriori informazioni, consulta Scelta del livello di promozione per un Aurora Serverless v2 reader.

Creare un Aurora Serverless v2 cluster di database

È possibile utilizzare AWS Management Console AWS CLI, o RDS API per creare un Aurora Serverless v2 Cluster DB.

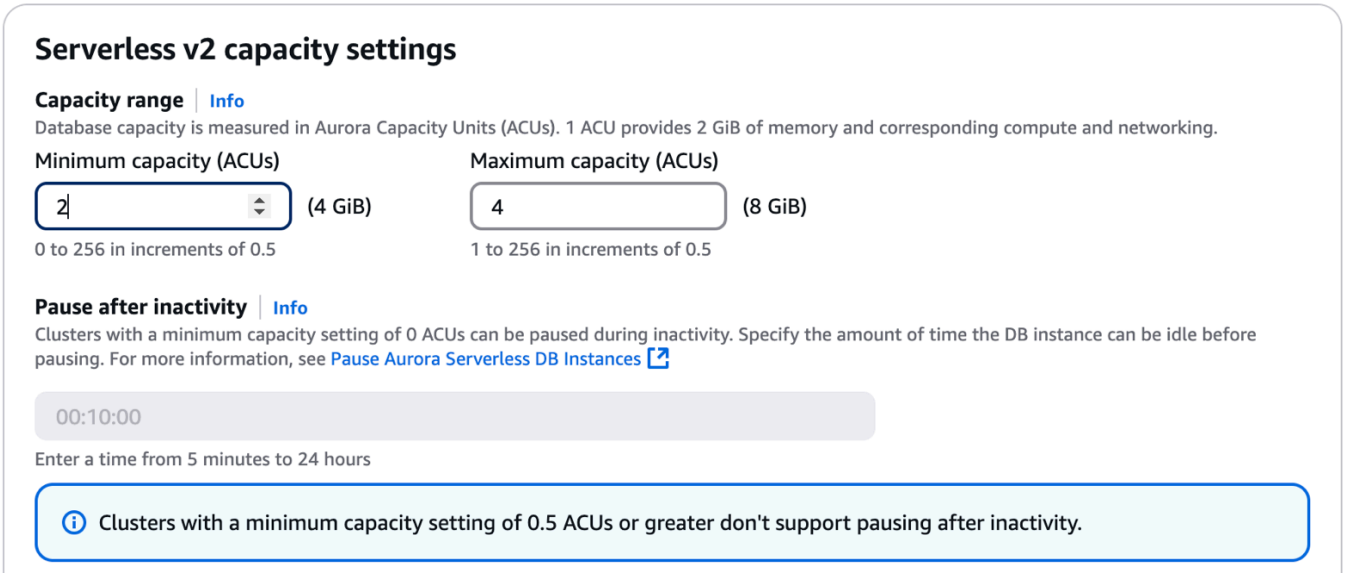
Console

Per creare un cluster con un Aurora Serverless v2 scrittore

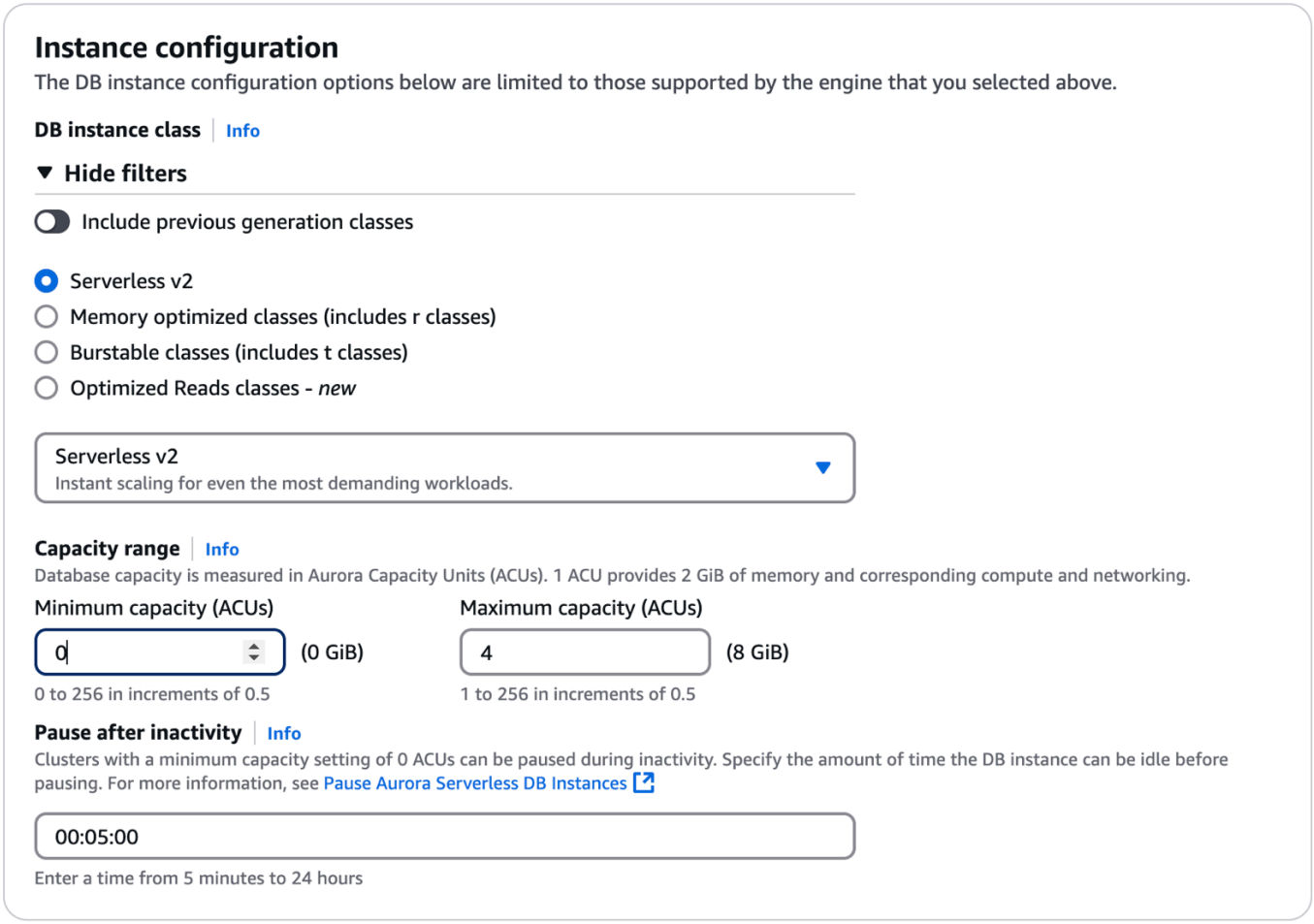
1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere Create database (Crea database). Nella pagina visualizzata, scegliere le opzioni seguenti:
   - Per Tipo di motore, scegli Aurora (My SQL Compatible) o Aurora (Postgre Compatible). SQL
   - Per Versione, scegli una delle versioni supportate per. Regioni supportate e motori Aurora DB per Aurora Serverless v2
4. Per la classe di istanze DB, seleziona Serverless v2.
5. Per l'intervallo di capacità, puoi accettare l'intervallo predefinito. In alternativa, puoi scegliere altri valori per le unità di capacità minima e massima. È possibile scegliere tra ACUs un minimo di 0 e un ACUs massimo di 256, con incrementi di 0,5ACU.

   Per ulteriori informazioni sull' Aurora Serverless v2 unità di capacità, vedere Aurora Serverless v2 capacità ePrestazioni e dimensionamento per Aurora Serverless v2.

   A seconda del motore e della versione scelti, il limite superiore può essere 128ACUs, il limite inferiore può essere 0,5 ACUs o entrambi. Per informazioni dettagliate sul limite per ogni combinazione di motore e versione Aurora, consulta. Aurora Serverless v2 capacità



La scelta di una capacità minima pari a 0 ACUs abilita Aurora Serverless v2 funzionalità di pausa e ripresa automatiche. In tal caso, puoi scegliere ulteriormente per quanto tempo Aurora Serverless v2 Le istanze DB attendono senza connessioni al database prima di interrompersi automaticamente. Per informazioni sulla funzionalità di pausa e ripresa automatica, consulta. Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2



6. Scegliete qualsiasi altra impostazione del cluster DB, come descritto inImpostazioni per cluster di database Aurora.
7. Scegli Crea database per creare il tuo cluster Aurora DB con un Aurora Serverless v2 Istanza DB come istanza writer, nota anche come istanza DB principale.

CLI

Per creare un cluster DB compatibile con Aurora Serverless v2 Istanze DB che utilizzano il AWS CLI, segui la CLI procedura descritta inCreazione di un cluster database Amazon Aurora. Includi i seguenti parametri nel comando `create-db-cluster`:

- --region *AWS_Region_where_Aurora Serverless v2_instances_are_available*
- --versione-engine *serverless_v2_compatible_engine_version*
- --serverless-v2-scaling-configuration =, = MinCapacity *minimum_capacity* MaxCapacity *maximum_capacity*

L'esempio seguente mostra la creazione di un Aurora Serverless v2 Cluster DB.

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.04.1 \
    --serverless-v2-scaling-configuration MinCapacity=1,MaxCapacity=4 \
    --master-username myuser \
    --manage-master-user-password
```

> **Note**
>
> Quando si crea un Aurora Serverless v2 Cluster DB che utilizza la AWS CLI, la modalità motore appare nell'output come `provisioned` anzichéserverless. La modalità `serverless` motore si riferisce a Aurora Serverless v1.

Questo esempio specifica l'`--manage-master-user-password`opzione per generare la password amministrativa e gestirla in Secrets Manager. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager. In alternativa, puoi utilizzare l'opzione `--master-password` per specificare e gestire personalmente la password.

Per informazioni su Aurora Serverless v2 requisiti di versione, vedereRequisiti e limitazioni per Aurora Serverless v2. Per informazioni sui numeri consentiti per l'intervallo di capacità e su cosa rappresentano questi numeri, consulta Aurora Serverless v2 capacità e Prestazioni e dimensionamento per Aurora Serverless v2.

Per verificare se per un cluster esistente sono state specificate le impostazioni di capacità, controlla l'output del comando `describe-db-clusters` per l'attributo `ServerlessV2ScalingConfiguration`. Questo attributo è simile a quello riportato di seguito.

```
"ServerlessV2ScalingConfiguration": {
```

```
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Se non specifichi il minimo e il massimo ACUs quando crei il cluster, puoi usare il `modify-db-cluster` comando successivo per aggiungere quell'impostazione. Finché non lo fai, non puoi aggiungerne Aurora Serverless v2 Istanze DB al cluster. Se tenti di aggiungere un'istanza database `db.serverless`, verrà restituito un errore.

API

Per creare un cluster DB compatibile con Aurora Serverless v2 Istanze DB che utilizzano il RDSAPI, segui la API procedura descritta inCreazione di un cluster database Amazon Aurora. Scegli le impostazioni descritte di seguito. Assicurati che l'operazione `CreateDBCluster` includa i parametri seguenti:

```
EngineVersion serverless_v2_compatible_engine_version
ServerlessV2ScalingConfiguration with MinCapacity=minimum_capacity and
 MaxCapacity=maximum_capacity
```

Per informazioni su Aurora Serverless v2 requisiti di versione, vedereRequisiti e limitazioni per Aurora Serverless v2. Per informazioni sui numeri consentiti per l'intervallo di capacità e su cosa rappresentano questi numeri, consulta Aurora Serverless v2 capacità e Prestazioni e dimensionamento per Aurora Serverless v2.

Per verificare se per un cluster esistente sono state specificate le impostazioni di capacità, controlla l'output dell'operazione `DescribeDBClusters` per l'attributo `ServerlessV2ScalingConfiguration`. Questo attributo è simile a quello riportato di seguito.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Se non specifichi il minimo e il massimo ACUs quando crei il cluster, puoi utilizzare l'`ModifyDBCluster`operazione successiva per aggiungere quell'impostazione. Finché non lo fai, non puoi aggiungerne Aurora Serverless v2 Istanze DB al cluster. Se tenti di aggiungere un'istanza database `db.serverless`, verrà restituito un errore.

Creare un Aurora Serverless v2 istanza Writer DB

Sebbene tu specifichi il Aurora Serverless v2 intervallo di capacità: quando si crea un cluster Aurora, è possibile scegliere se utilizzare Aurora Serverless v2 o fornito per ogni istanza DB del cluster. Per iniziare a usare Aurora Serverless v2 immediatamente nel tuo cluster DB, aggiungi un'istanza Writer DB che utilizza la classe di `db.serverless` istanza. Nella console, in genere si effettua questa scelta come parte del flusso di lavoro per creare il cluster DB. Pertanto, questa procedura si applica principalmente quando si esegue la configurazione tramiteCLI.

Console

Quando si crea un cluster DB utilizzando il AWS Management Console, si specificano contemporaneamente le proprietà dell'istanza Writer DB. Per utilizzare l'istanza Writer DB Aurora Serverless v2, scegli la classe di istanza DB Serverless.

Quindi si imposta l'intervallo di capacità per il cluster specificando i valori minimo e massimo dell'unità di capacità Aurora ACU (). Questi valori minimo e massimo si applicano a ciascuno Aurora Serverless v2 Istanza DB nel cluster. Per questa procedura e l'importanza delle impostazioni di capacità minima e massima, vedereCreare un Aurora Serverless v2 cluster di database.

Se non crei un Aurora Serverless v2 L'istanza DB quando si crea il cluster per la prima volta, è possibile aggiungerne una o più Aurora Serverless v2 Istanze DB successive. A tale scopo, seguire la procedura descritta in Aggiungere un Aurora Serverless v2 reader e Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2. L'intervallo di capacità viene specificato nel momento in cui si aggiunge il primo Aurora Serverless v2 Istanza DB al cluster. È possibile modificare l'intervallo di capacità in un secondo momento seguendo la procedura descritta in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

CLI

Quando si crea un Aurora Serverless v2 Cluster DB utilizzando AWS CLI, si aggiunge esplicitamente l'istanza Writer DB utilizzando il create-db-instancecomando. Includere il seguente parametro:

- `--db-instance-class db.serverless`

L'esempio seguente mostra la creazione di un Aurora Serverless v2 istanza Writer DB.

```
aws rds create-db-instance \
    --db-cluster-identifier my-serverless-v2-cluster \
    --db-instance-identifier my-serverless-v2-instance \
    --db-instance-class db.serverless \
    --engine aurora-mysql
```

Gestione di Aurora Serverless v2 Cluster database

Con Aurora Serverless v2, i cluster sono intercambiabili con i cluster forniti. Il Aurora Serverless v2 le proprietà si applicano a una o più istanze DB all'interno di un cluster DB. Pertanto, le procedure per la creazione e la modifica di cluster, per la creazione e il ripristino di snapshot e così via sono sostanzialmente le stesse procedure usate per gli altri tipi di cluster Aurora. Per le procedure generali per la gestione di cluster Aurora e istanze database, consulta Gestione di un cluster DB Amazon Aurora.

Nei seguenti argomenti, è possibile apprendere le considerazioni sulla gestione dei cluster che contengono Aurora Serverless v2 Istanze DB.

Argomenti

- Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster
- Verifica dell'intervallo di capacità per Aurora Serverless v2
- Aggiungere un Aurora Serverless v2 reader
- Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2
- Conversione di un Aurora Serverless v2 scrittore o lettore a provisioned
- Pausa Aurora Serverless v2 Istanze DB
- Scelta del livello di promozione per un Aurora Serverless v2 reader
- Utilizzo di TLS/SSL con Aurora Serverless v2
- Visualizzazione Aurora Serverless v2 scrittori e lettori
- Registrazione per Aurora Serverless v2

Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster

Per modificare i parametri di configurazione o altre impostazioni per i cluster contenenti Aurora Serverless v2 Le istanze DB, o le istanze DB stesse, seguono le stesse procedure generali dei cluster predisposti. Per informazioni dettagliate, consultare Modifica di un cluster database Amazon Aurora.

L'impostazione più importante che è unica per Aurora Serverless v2 è l'intervallo di capacità. Dopo aver impostato i valori minimi e massimi dell'unità di capacità Aurora (ACU) per un cluster Aurora, non è necessario regolare attivamente la capacità del Aurora Serverless v2 Istanze DB nel cluster. Aurora esegue automaticamente questa operazione. Questa impostazione è gestita a livello di cluster. Gli stessi valori ACU minimo e massimo si applicano a ciascuna Aurora Serverless v2 Istanza DB nel cluster.

Puoi impostare i seguenti valori specifici:

- ACUsMinimo: Aurora Serverless v2 L'istanza DB può ridurre la capacità fino a questo numero di ACUs.
- ACUsMassimo: Aurora Serverless v2 L'istanza DB può aumentare la capacità fino a questo numero di ACUs.

Le versioni più recenti del motore DB consentono una capacità massima di 256 ACUs, una capacità minima di 0 ACUs o entrambe. Per gli intervalli di capacità per le varie versioni del motore DB, vedereAurora Serverless v2 capacità.

Per la funzionalità di pausa e ripresa automatica abilitata impostando la capacità minima su 0 ACUs, vedere. Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2

Per ulteriori informazioni su come assicurarsi che Aurora Serverless v2 I cluster DB possono scalare fino a 256 ACUs, vediAggiornamento del Aurora Serverless v2 Cluster DB per consentire la scalabilità a 256 ACUs.

> **Note**
>
> Quando si modifica l'intervallo di capacità per un Aurora Serverless v2 Il cluster DB, la modifica avviene immediatamente, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione programmata.

In Aurora Serverless v2, non è possibile impostare direttamente la capacità corrente senza modificare l'intervallo di capacità, poiché la capacità si regola dinamicamente in base al carico di lavoro all'interno dell'intervallo specificato. Tuttavia, è possibile influenzare la capacità indirettamente nei seguenti modi:

- Regola la capacità minima: riduci temporaneamente la capacità minima per ridurre la capacità di base quando i carichi di lavoro sono leggeri.
- Sospendi temporaneamente la scalabilità: per fissare la capacità a un valore specifico, imposta la capacità minima e massima sullo stesso livello.
- Scala in modo proattivo per un utilizzo ottimale: se prevedi un aumento esponenziale dei carichi di lavoro, aumenta preventivamente la capacità minima per mantenere una linea di base più elevata durante i periodi di elevata attività.

Per informazioni dettagliate sugli effetti dell'intervallo di capacità e su come monitorarlo e ottimizzarlo, consulta CloudWatch Parametri Amazon importanti per Aurora Serverless v2 e Prestazioni e dimensionamento per Aurora Serverless v2. L'obiettivo è far sì che la capacità massima per il cluster sia sufficientemente elevata da essere in grado di gestire i picchi del carico di lavoro e che la capacità minima sia sufficientemente bassa da ridurre al minimo i costi quando il cluster non è occupato.

Supponiamo che in base al monitoraggio effettuato si determini che l'intervallo ACU per il cluster debba essere più alto, più basso, meno limitativo o più limitativo. Puoi impostare la capacità di un cluster Aurora su un intervallo specifico di ACUs con AWS Management Console, the o l' AWS CLI API Amazon RDS. Questo intervallo di capacità si applica a tutti Aurora Serverless v2 Istanza DB nel cluster.

Ad esempio, supponiamo che il cluster abbia un intervallo di capacità compreso tra 1 e 16 e ne ACUs contenga due Aurora Serverless v2 Istanze DB. Quindi il cluster nel suo insieme consuma tra 2 ACUs (quando è inattivo) e 32 ACUs (quando è completamente utilizzato). Se si modifica l'intervallo di capacità da 8 a 40,5 ACUs, ora l'intero cluster ne consuma 16 ACUs quando è inattivo e fino a 81 quando è completamente utilizzato. ACUs

Aurora imposta automaticamente determinati parametri per Aurora Serverless v2 istanze DB con valori che dipendono dal valore ACU massimo nell'intervallo di capacità. Per l'elenco completo dei parametri, consulta Numero massimo connessioni per Aurora Serverless v2. Per i parametri statici che si basano su questo tipo di calcolo, il valore viene nuovamente valutato al riavvio dell'istanza database. Pertanto, è possibile aggiornare il valore di tali parametri riavviando l'istanza database dopo aver modificato l'intervallo di capacità. Per verificare se è necessario

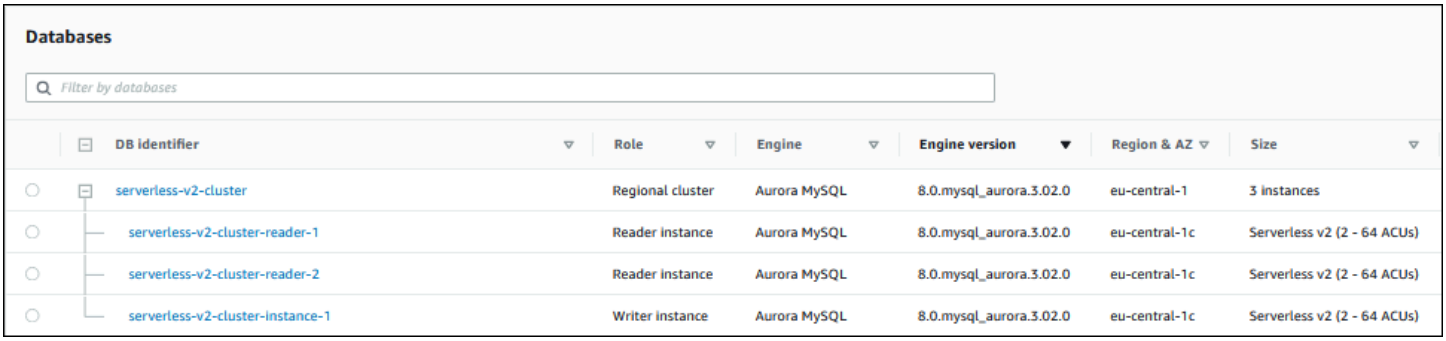
riavviare l'istanza database per acquisire le modifiche apportate ai parametri, controlla l'attributo `ParameterApplyStatus` dell'istanza database. Il valore `pending-reboot` indica che il riavvio applicherà le modifiche ad alcuni valori di parametro.

Console

È possibile impostare l'intervallo di capacità di un cluster che contiene Aurora Serverless v2 Istanze DB con. AWS Management Console

Quando si utilizza la console, si imposta l'intervallo di capacità per il cluster nel momento in cui si aggiunge il primo Aurora Serverless v2 Istanza DB a quel cluster. Questa operazione può essere eseguita quando scegli la classe di istanza database Serverless v2 per l'istanza database di scrittura durante la creazione del cluster. Oppure puoi farlo quando scegli la classe di istanza DB Serverless quando aggiungi un Aurora Serverless v2 istanza DB reader al cluster. In alternativa, puoi eseguire questa operazione quando converti un'istanza database con provisioning esistente nel cluster in un'istanza database di classe Serverless. Per le versioni complete di tali procedure, vedere Creare un Aurora Serverless v2 istanza Writer DBAggiungere un Aurora Serverless v2 reader, eConversione di uno scrittore o lettore predisposto in Aurora Serverless v2.

Qualsiasi intervallo di capacità impostato a livello di cluster si applica a tutti Aurora Serverless v2 Istanze DB nel cluster. L'immagine seguente mostra un cluster con più Aurora Serverless v2 istanze Reader DB. Ciascuna ha un intervallo di capacità identico compreso tra 2 ACUs e 64.



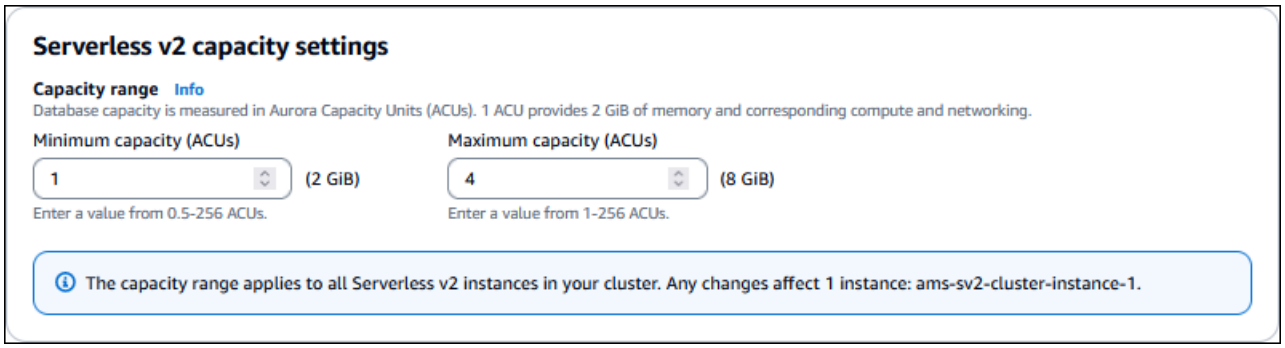
Per modificare l'intervallo di capacità di un Aurora Serverless v2 cluster

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Databases (Database).
3. Scegli il cluster che contiene il tuo Aurora Serverless v2 Istanze DB dall'elenco. Il cluster deve già contenerne almeno una Aurora Serverless v2 Istanza DB. In caso contrario, Aurora non visualizza la sezione Capacity range (Intervallo capacità).
4. Per Operazioni, scegli Modifica.

5. Nella sezione Capacity range (Intervallo capacità), scegliere quanto segue:

   a. Inserisci un valore per Minimum ACUs. La console mostra l'intervallo di valori consentito. È possibile scegliere una capacità minima da 0 a 256 ACUs. È possibile scegliere una capacità massima da 1 a 256 ACUs. È possibile regolare i valori di capacità con incrementi di 0,5 ACU.

   b. Immettete un valore per Massimo. ACUs Questo valore deve essere maggiore o uguale a Minimo ACUs. La console mostra l'intervallo di valori consentito. La figura seguente mostra questa scelta.



6. Scegli Continue (Continua). Viene visualizzata la pagina Riepilogo delle modifiche .
7. Scegliere Apply immediately (Applica immediatamente).

   La modifica della capacità viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.
8. Seleziona Modifica cluster per accettare il riepilogo delle modifiche. Puoi inoltre scegliere Indietro per modificare le modifiche o Annulla per ignorarle.

AWS CLI

Per impostare la capacità di un cluster in cui intendi utilizzarlo Aurora Serverless v2 Istanze DB che utilizzano il AWS CLI, esegui il modify-db-cluster AWS CLI comando. Specifica l'opzione `--serverless-v2-scaling-configuration`. Il cluster potrebbe già contenerne uno o più Aurora Serverless v2 istanze DB oppure puoi aggiungere le istanze DB in un secondo momento. I valori validi per i campi `MinCapacity` e `MaxCapacity` includono quelli riportati di seguito:

- `0`,`0.5`,`1`, `1.52`, e così via, con passaggi da 0,5 a un massimo di 256. Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzione di pausa automatica.

In questo esempio, viene impostato l'intervallo ACU di un cluster database Aurora denominato `sample-cluster` sul valore minimo `48.5` e sul valore massimo 64.

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
  --serverless-v2-scaling-configuration MinCapacity=48.5,MaxCapacity=64
```

La modifica della capacità viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.

Dopo averlo fatto, puoi aggiungere Aurora Serverless v2 Le istanze DB al cluster e ogni nuova istanza DB può essere scalata tra 48,5 e 64. ACUs La nuova gamma di capacità si applica anche a qualsiasi Aurora Serverless v2 Istanze DB già presenti nel cluster. Se necessario, le istanze database vengono aumentate o ridotte in modo che i relativi valori siano compresi nel nuovo intervallo di capacità.

Per ulteriori esempi di impostazione dell'intervallo di capacità utilizzando la CLI, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora.

Per modificare la configurazione di scalabilità di un Aurora Serverless Cluster DB che utilizza AWS CLI, esegui il modify-db-cluster AWS CLI comando. Specifica l'opzione `--serverless-v2-scaling-configuration` per configurare la capacità minima e la capacità massima. I valori di capacità validi includono quanto segue:

- Aurora MySQL:`0`,,,, e così via `0.5` `1` `1.52`, con incrementi da 0,5 fino a un massimo di. ACUs `256`
- Aurora PostgreSQL:`0`,,,, e così via `0.5` `1` `1.52`, con incrementi da 0,5 fino a un massimo di. ACUs `256`
- Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzione di pausa automatica.

Nell'esempio seguente, si modifica la configurazione di ridimensionamento di un Aurora Serverless v2 Istanza DB denominata `sample-instance` che fa parte di un cluster denominato`sample-cluster`.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

In Windows:

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster ^
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

API RDS

È possibile impostare la capacità di un'istanza Aurora DB con l'operazione Modify DBCluster API. Specifica il parametro `ServerlessV2ScalingConfiguration`. I valori validi per i campi `MinCapacity` e `MaxCapacity` includono quelli riportati di seguito:

- `0`,`0.5`,`1`, `1.52`, e così via, con passaggi da 0,5 a un massimo di 256. Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzione di pausa automatica.

È possibile modificare la configurazione di scalabilità di un cluster contenente Aurora Serverless v2 Istanze DB con l'operazione Modify DBCluster API. Specifica il parametro `ServerlessV2ScalingConfiguration` per configurare la capacità minima e la capacità massima. I valori di capacità validi includono quanto segue:

- Aurora MySQL:`0`,,,, e così via `0.5` `1` `1.52`, con incrementi da 0,5 fino a un massimo di. ACUs `256`
- Aurora PostgreSQL:`0`,,,, e così via `0.5` `1` `1.52`, con incrementi da 0,5 fino a un massimo di. ACUs `256`
- Il valore ACU minimo pari a 0 è disponibile solo per le versioni Aurora che supportano la funzione di pausa automatica.

La modifica della capacità viene implementata subito, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione pianificata.

Aggiornamento del Aurora Serverless v2 Cluster DB per consentire la scalabilità a 256 ACUs

In alcuni casi, quando si tenta di ridimensionare il Aurora Serverless v2 Il cluster DB fino a capacità superiori a 128 ACUs, viene visualizzato un messaggio di errore. Il messaggio di errore indica quali istanze sono incompatibili con il nuovo intervallo di scalabilità.

L'impossibilità di scalare oltre 128 ACUs può verificarsi per uno dei due motivi seguenti:

- Versione precedente del motore DB: aggiorna la versione del motore DB a una che supporti 256 ACUs. Per ulteriori informazioni, consulta Aurora Serverless v2 capacità.

- Piattaforma precedente: aggiorna la piattaforma sottostante per la tua Aurora Serverless v2 Cluster DB. Questa operazione può essere eseguita in uno dei seguenti modi:
  - Arresta e riavvia il cluster DB. Ciò porrà fine alla piattaforma sottostante esistente e ne fornirà una nuova in grado di supportare 256 ACUs.

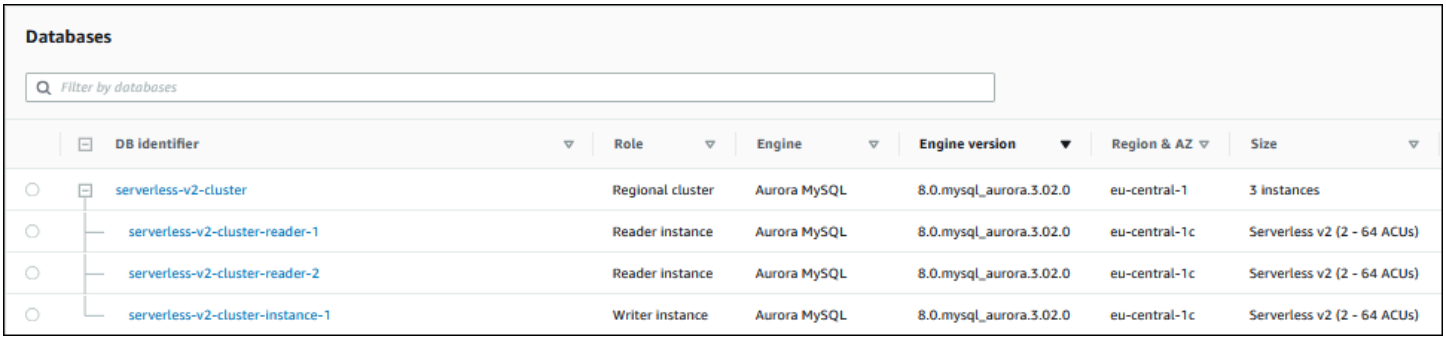
    Tuttavia, l'arresto e l'avvio di un cluster di database comportano alcuni tempi di inattività, in genere diversi minuti. Per ulteriori informazioni, consulta Avvio e arresto di un cluster di database Amazon Aurora.
  - Sostituisci le istanze precedenti ed esegui il failover su una delle nuove istanze.
    1. Aggiungi istanze DB Reader. Le nuove istanze di lettura verranno eseguite su hardware aggiornato in grado di supportare 256 istanze. ACUs Per ulteriori informazioni, consulta Aggiungere un Aurora Serverless v2 reader.
    2. Effettua il failover su una delle nuove istanze del lettore. Per ulteriori informazioni, consulta Failing su un cluster Amazon Aurora DB.
    3. Dopo il failover, potete eliminare le istanze precedenti, inclusa la precedente istanza di Writer.
  - Usa una distribuzione blu/verde. Per ulteriori informazioni, consulta Panoramica delle implementazioni di Amazon Aurora Blue/Green.
    1. Crea una distribuzione blu/verde. Per ulteriori informazioni, consulta Creazione di una distribuzione blu/verde in Amazon .
    2. Conferma di poter impostare la capacità massima per l'ambiente di staging (verde) su 256. ACUs
    3. Passa all'ambiente verde. Per ulteriori informazioni, consulta Cambiare una distribuzione blu/verde in Amazon .
    4. Eliminare il cluster DB originale.

    > **Note**
    >
    > Se conservi le credenziali del database in AWS Secrets Manager, non puoi utilizzare distribuzioni blu/verdi.
    > Aurora Global Database non supporta implementazioni blu/verdi.

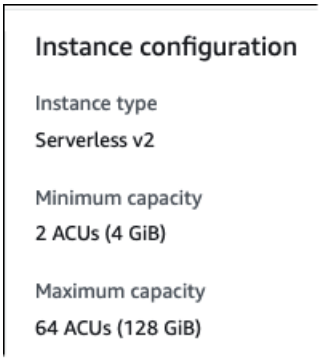
Verifica dell'intervallo di capacità per Aurora Serverless v2

La procedura per verificare l'intervallo di capacità del Aurora Serverless v2 il cluster richiede innanzitutto l'impostazione di un intervallo di capacità. Se questa verifica non è ancora stata eseguita, segui la procedura descritta in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

Qualunque sia l'intervallo di capacità impostato a livello di cluster, si applica a tutti Aurora Serverless v2 Istanze DB nel cluster. L'immagine seguente mostra un cluster con più Aurora Serverless v2 Istanze DB. Ogni istanza ha un intervallo di capacità identico.



Puoi anche visualizzare la pagina dei dettagli per qualsiasi Aurora Serverless v2 Istanza DB nel cluster. L'intervallo di capacità delle istanze database è visuaizzato nella scheda Configurazione.



È inoltre possibile visualizzare l'intervallo di capacità corrente per il cluster nella pagina Modifica del cluster. A questo punto, è possibile modificare l'intervallo di capacità. Per informazioni su come impostare o modificare l'intervallo di capacità, consulta Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

Verifica dell'intervallo di capacità corrente per un cluster Aurora

È possibile verificare l'intervallo di capacità configurato per Aurora Serverless v2 Istanze DB in un cluster esaminando l'`ServerlessV2ScalingConfiguration`attributo del cluster. L' AWS CLI

esempio seguente mostra un cluster con una capacità minima di 0,5 unità di capacità Aurora (ACUs) e una capacità massima di 16. ACUs

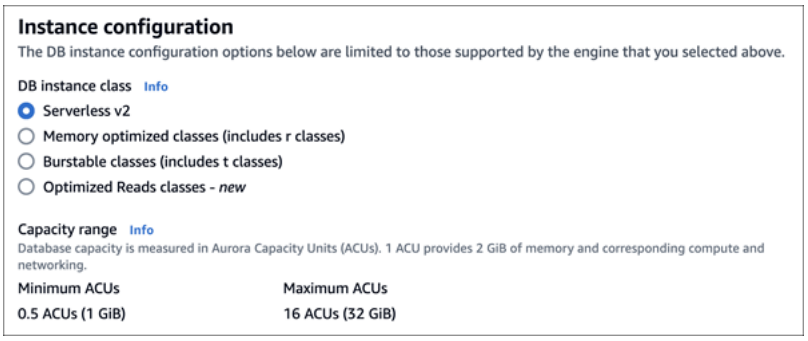
```
$ aws rds describe-db-clusters --db-cluster-identifier serverless-v2-64-acu-cluster \
  --query 'DBClusters[*].[ServerlessV2ScalingConfiguration]'
[
    [
        {
            "MinCapacity": 0.5,
            "MaxCapacity": 16.0
        }
    ]
]
```

Aggiungere un Aurora Serverless v2 reader

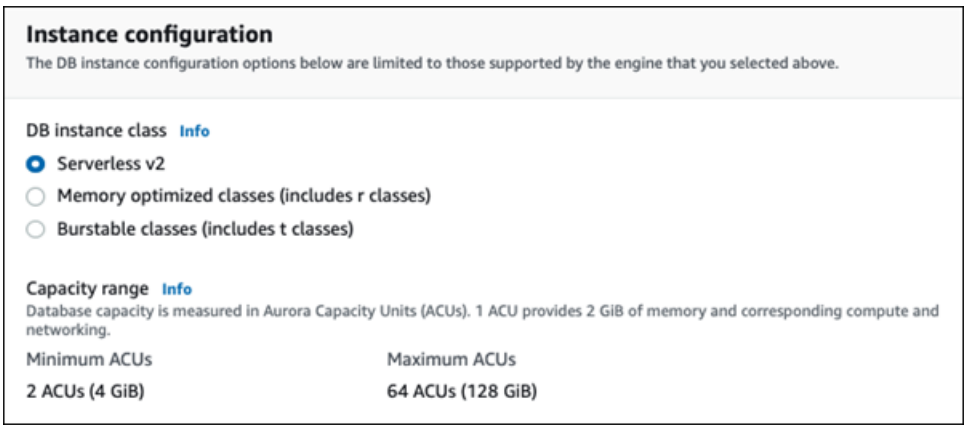
Per aggiungere un Aurora Serverless v2 Un'istanza Reader DB al tuo cluster, segui la stessa procedura generale descritta in[Aggiunta di repliche di Aurora a un cluster di database](). Seleziona Serverless v2 come classe di istanza per la nuova istanza database.

Se l'istanza Reader DB è la prima Aurora Serverless v2 Istanza DB nel cluster, scegli anche l'intervallo di capacità. Questa impostazione si applica a questa istanza DB Reader e a qualsiasi altra Aurora Serverless v2 Istanze DB che aggiungi al cluster. Ciascuno Aurora Serverless v2 L'istanza DB può essere scalata tra i valori ACU minimo e massimo.

Se ce ne sono altri Aurora Serverless v2 le istanze esistono già nel cluster, la finestra di dialogo Aggiungi lettore mostra l'intervallo di capacità corrente per il cluster. In tal caso, non è possibile modificare la capacità. L'immagine seguente mostra il rapporto sulla capacità attuale del cluster.



Se ne hai già aggiunto Aurora Serverless v2 Istanze DB al cluster, aggiungendone un'altra Aurora Serverless v2 L'istanza Reader DB mostra l'intervallo di capacità corrente. L'immagine seguente mostra i controlli di sola lettura.



Se vuoi modificare l'intervallo di capacità per il cluster, segui la procedura descritta in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

Per i cluster contenenti più di un'istanza Reader DB, la priorità di failover di ciascuna Aurora Serverless v2 L'istanza Reader DB svolge un ruolo importante nella scalabilità verso l'alto e verso il basso di tale istanza DB. Non è possibile specificare la priorità nella fase iniziale di creazione del cluster. Tieni presente questa proprietà quando al cluster aggiungi una seconda istanza database di lettura o altre istanze di questo tipo. Per ulteriori informazioni, consulta Scelta del livello di promozione per un Aurora Serverless v2 reader.

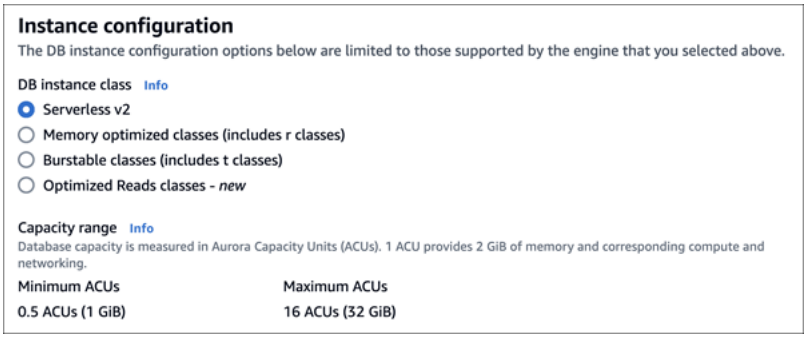
Per altri modi in cui è possibile visualizzare l'intervallo di capacità corrente per un cluster, consulta Verifica dell'intervallo di capacità per Aurora Serverless v2.

Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2

È possibile convertire un'istanza DB predisposta per utilizzarla Aurora Serverless v2. A tale scopo, segui la procedura descritta inModifica di un'istanza database in un cluster database. Il cluster deve soddisfare i requisiti riportati in Requisiti e limitazioni per Aurora Serverless v2. Ad esempio, Aurora Serverless v2 Le istanze DB richiedono che il cluster esegua determinate versioni minime del motore.

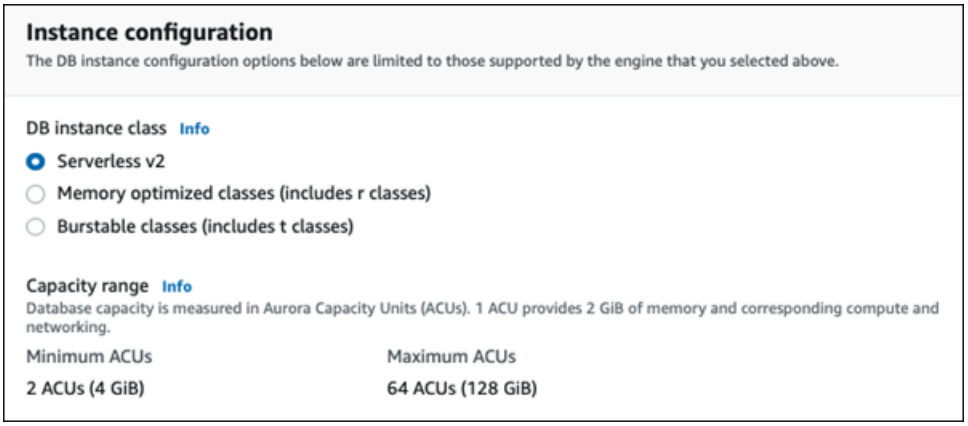
Supponiamo che stiate convertendo un cluster con provisioning in esecuzione per sfruttare i vantaggi di Aurora Serverless v2. In tal caso, è possibile ridurre al minimo i tempi di inattività convertendo un'istanza DB in Aurora Serverless v2 come prima fase del processo di passaggio al digitale. Per la procedura completa, consulta Passaggio da un cluster fornito a Aurora Serverless v2.

Se l'istanza DB che converti è la prima Aurora Serverless v2 Istanza DB nel cluster, si sceglie l'intervallo di capacità per il cluster come parte dell'operazione di modifica. Questo intervallo di capacità si applica a ciascuno Aurora Serverless v2 Istanza DB aggiunta al cluster. L'immagine seguente mostra la pagina in cui si specificano le unità di capacità Aurora minima e massima ()ACUs.



Per ulteriori informazioni sull'importanza dell'intervallo di capacità, consulta Aurora Serverless v2 capacità.

Se il cluster ne contiene già una o più Aurora Serverless v2 Istanze DB, viene visualizzato l'intervallo di capacità esistente durante l'operazione di modifica. L'immagine seguente mostra un esempio del pannello informativo.



Se desideri modificare l'intervallo di capacità per il cluster dopo averne aggiunto altro Aurora Serverless v2 Istanze DB, segui la procedura riportata inImpostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

Conversione di un Aurora Serverless v2 scrittore o lettore a provisioned

È possibile convertire un Aurora Serverless v2 Un'istanza DB in un'istanza DB fornita. A tale scopo, segui la procedura descritta in Modifica di un'istanza database in un cluster database. Scegli una classe di istanza database diversa da Serverless.

È possibile convertire un Aurora Serverless v2 Istanza DB da fornire se necessita di una capacità maggiore di quella disponibile con le unità di capacità Aurora massime ACUs () di un Aurora Serverless v2 Istanza DB. Ad esempio, le classi di istanze DB db.r5 e db.r6g più grandi hanno una capacità di memoria maggiore rispetto a Aurora Serverless v2 L'istanza DB può essere scalata fino a.

> **Tip**
>
> Alcune delle classi di istanze DB più vecchie, come db.r3 e db.t2, non sono disponibili per le versioni di Aurora utilizzate con Aurora Serverless v2. Per vedere quali classi di istanze DB è possibile utilizzare durante la conversione di un Aurora Serverless v2 Un'istanza DB in un'istanza predisposta, vedi. Motori DB supportati per classi di istanza database

Se state convertendo l'istanza Writer DB del vostro cluster da Aurora Serverless v2 per provisioned, è possibile seguire la procedura in Passaggio da un cluster fornito a Aurora Serverless v2 senso inverso. Cambia una delle istanze DB del lettore nel cluster da Aurora Serverless v2 a cui è stato effettuato il provisioning. Esegui quindi un failover per convertire l'istanza database con provisioning in istanza di scrittura.

Qualsiasi intervallo di capacità precedentemente specificato per il cluster rimane invariato, anche se tutti Aurora Serverless v2 Le istanze DB vengono rimosse dal cluster. Se vuoi modificare l'intervallo di capacità, puoi modificare il cluster, come spiegato in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

Pausa Aurora Serverless v2 Istanze DB

È possibile configurare i cluster Aurora per metterli in pausa e riprenderli automaticamente. Aurora Serverless v2 istanze DB. Per ulteriori informazioni, consulta Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2.

Scelta del livello di promozione per un Aurora Serverless v2 reader

Per i cluster contenenti più Aurora Serverless v2 Istanze DB o una combinazione di istanze fornite e Aurora Serverless v2 Per ciascuna istanza DB, presta attenzione all'impostazione del livello di promozione Aurora Serverless v2 Istanza DB. Questa impostazione controlla più comportamenti per Aurora Serverless v2 Istanze DB rispetto alle istanze DB predisposte.

In AWS Management Console, si specifica questa impostazione utilizzando l'opzione di priorità di failover in Configurazione aggiuntiva per le pagine Crea database, Modifica istanza e Aggiungi lettore. Questa proprietà viene visualizzata per le istanze database esistenti nella colonna facoltativa Livello di priorità nella pagina Database. È inoltre possibile visualizzare questa proprietà nella pagina dei dettagli di un cluster di database o un'istanza database.

Per le istanze database con provisioning, la scelta del livello 0-15 determina solo l'ordine in base al quale Aurora sceglie l'istanza database di lettura da promuovere a istanza di scrittura durante un'operazione di failover. In Aurora Serverless v2 Reader DB Instances, il numero di livello determina anche se l'istanza DB è scalabile fino a soddisfare la capacità dell'istanza DB Writer o se è scalabile indipendentemente in base al proprio carico di lavoro. Aurora Serverless v2 Le istanze Reader DB di livello 0 o 1 vengono mantenute a una capacità minima almeno pari a quella dell'istanza DB writer. In questo modo, tali istanze sono pronte a prendere il controllo dall'istanza database di scrittura in caso di failover. Se l'istanza Writer DB è un'istanza DB fornita, Aurora stima l'equivalente Aurora Serverless v2 capacità. Utilizza tale stima come capacità minima per Aurora Serverless v2 istanza Reader DB.

Aurora Serverless v2 le istanze Reader DB di livello 2—15 non hanno lo stesso vincolo di capacità minima. Quando sono inattive, possono essere ridotte fino al valore minimo dell'unità di capacità Aurora (ACU) specificato nell'intervallo di capacità del cluster.

Il seguente AWS CLI esempio di Linux mostra come esaminare i livelli di promozione di tutte le istanze DB del cluster. Il campo finale include il valore `True` per l'istanza database di scrittura e il valore `False` per tutte le istanze database di lettura.

```
$ aws rds describe-db-clusters --db-cluster-identifier promotion-tier-demo \
  --query 'DBClusters[*].DBClusterMembers[*].
[PromotionTier,DBInstanceIdentifier,IsClusterWriter]' \
  --output text

1   instance-192  True
1   tier-01-4840  False
10  tier-10-7425  False
15  tier-15-6694  False
```

Il seguente AWS CLI esempio di Linux mostra come modificare il livello di promozione di una specifica istanza DB nel cluster. I comandi modificano innanzitutto l'istanza database in base a un nuovo livello di promozione. Attendono quindi che l'istanza database ridiventi disponibile e ne confermano il nuovo livello di promozione.

```
$ aws rds modify-db-instance --db-instance-identifier instance-192 --promotion-tier 0
$ aws rds wait db-instance-available --db-instance-identifier instance-192
$ aws rds describe-db-instances --db-instance-identifier instance-192 \
  --query '*[].[PromotionTier]' --output text
0
```

Per ulteriori informazioni su come specificare i livelli di promozione per i diversi casi d'uso, consulta Aurora Serverless v2 scalabilità.

Utilizzo di TLS/SSL con Aurora Serverless v2

Aurora Serverless v2 può utilizzare il protocollo Transport Layer Security/Secure Sockets Layer (TLS/SSL () per crittografare le comunicazioni tra i client e il Aurora Serverless v2 istanze DB. Supporta TLS/SSL versions 1.0, 1.1, and 1.2. For general information about using TLS/SSL Aurora, vedi. Connessioni TLS ai cluster Aurora MySQL DB

Per maggiori informazioni sulla connessione al database Aurora MySQL con il client MySQL, consulta Connessione a un'istanza database che esegue il motore del database MySQL.

Aurora Serverless v2 supporta tutte le modalità TLS/SSL disponibili per il client MySQL (`mysql`) e il client PostgreSQL (), incluse quelle elencate nella tabella seguente. `psql`

Descrizione della modalità TLS/SSL	mysql	psql
Connettiti senza utilizzare TLS/SSL.	DISABLED	disattiva
Prova prima la connessione utilizzando TLS/SSL, ma se necessario, torna all'uso senza SSL.	PREFERRED	prefer (impostazione predefinita)
Applica l'uso di TLS/SSL.	REQUIRED	require
Applicare TLS/SSL e verificare l'autorità di certificazione (CA).	VERIFY_CA	verify-ca
Applica TLS/SSL, verifica la CA e verifica il nome host della CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v2 utilizza certificati wildcard. Se specifichi l'opzione "verifica CA" o "verifica CA e nome host CA" quando utilizzi TLS/SSL, scarica innanzitutto il trust store CA 1 radice di Amazon da Amazon Trust Services. Dopo aver fatto ciò, puoi identificare questo file in formato PEMA nel

comando client. Per eseguire questa operazione utilizzando il client PostgreSQL, procedi come segue.

In Linux, macOS, oppure Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Per ulteriori informazioni sull'utilizzo del database Aurora PostgreSQL utilizzando il client Postgres, consulta Connessione a un'istanza database che esegue il modulo di motore del database PostgreSQL.

Per informazioni generali sulla connessione ai cluster database Aurora, consulta Connessione a un cluster database Amazon Aurora.

Suite di crittografia supportate per connessioni a Aurora Serverless v2 Cluster database

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera abilitare per proteggere le connessioni TLS/SSL client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. In questo modo si impedisce l'uso di sistemi di crittografia non sicuri o obsoleti.

Aurora Serverless v2 I cluster DB basati su Aurora MySQL supportano le stesse suite di crittografia dei cluster DB forniti da Aurora MySQL. Per informazioni su queste suite di crittografia, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL.

Aurora Serverless v2 I cluster DB basati su Aurora PostgreSQL supportano le stesse suite di crittografia dei cluster DB forniti da Aurora PostgreSQL. Per informazioni su queste suite di crittografia, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL.

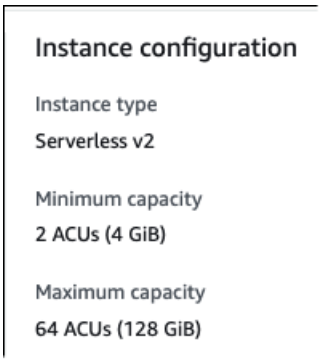
Visualizzazione Aurora Serverless v2 scrittori e lettori

È possibile visualizzare i dettagli di Aurora Serverless v2 Le istanze DB vengono eseguite nello stesso modo in cui lo si fa per le istanze DB di cui è stato eseguito il provisioning. A tale scopo, segui la procedura generale descritta in Visualizzazione di un cluster di database Amazon Aurora. Un cluster potrebbe contenere tutto Aurora Serverless v2 Istanze DB, tutte le istanze DB di cui è stato eseguito il provisioning o alcune di ciascuna.

Dopo averne creato uno o più Aurora Serverless v2 Istanze DB, è possibile visualizzare quali istanze DB sono di tipo Serverless e quali sono di tipo Instance. È inoltre possibile visualizzare le unità di capacità Aurora minima e massima (ACUs) che Aurora Serverless v2 L'istanza DB può utilizzare. Ogni ACU è la combinazione di capacità di calcolo (CPU) e memoria (RAM). Questo intervallo di capacità si applica a ciascuno Aurora Serverless v2 Istanza DB nel cluster. Per la procedura di verifica dell'intervallo di capacità di un cluster o di uno qualsiasi Aurora Serverless v2 Istanza DB nel cluster, vedereVerifica dell'intervallo di capacità per Aurora Serverless v2.

Nel AWS Management Console, Aurora Serverless v2 Le istanze DB sono contrassegnate nella colonna Dimensione nella pagina Database. Le istanze database con provisioning riportano il nome di una classe di istanza database, ad esempio r6g.xlarge. Il Aurora Serverless Le istanze DB mostrano Serverless per la classe di istanze DB, insieme alla capacità minima e massima dell'istanza DB. Ad esempio, potresti vedere Serverless v2 (4—64 ACUs) o Serverless v2 (1—40). ACUs

Puoi trovare le stesse informazioni nella scheda Configurazione per ciascuno Aurora Serverless v2 Istanza DB nella console. Ad esempio, è possibile che venga visualizzata una sezione Tipo di istanza simile a quella illustrata di seguito. Qui, il valore del tipo di istanza è Serverless v2, il valore di capacità minima è 2 ACUs (4 GiB) e il valore di capacità massima è 64 (ACUs 128 GiB).



È possibile monitorare la capacità di ciascuno Aurora Serverless v2 Istanza DB nel tempo. In questo modo, puoi controllare il ACUs consumo minimo, massimo e medio di ogni istanza DB. È inoltre possibile verificare la prossimità dell'istanza database alla relativa capacità minima o massima. Per visualizzare tali dettagli in AWS Management Console, esamina i grafici delle CloudWatch metriche di Amazon nella scheda Monitoraggio per l'istanza DB. Per ulteriori informazioni su come visualizzare e interpretare i parametri, consulta CloudWatch Parametri Amazon importanti per Aurora Serverless v2.

Registrazione per Aurora Serverless v2

Per attivare la registrazione del database, specifica i registri da abilitare mediante i parametri di configurazione nel gruppo di parametri personalizzati.

Per Aurora MySQL, puoi abilitare i seguenti registri.

Aurora MySQL	Descrizione
`general_log`	Crea il log generale. Imposta su 1 per attivare questa opzione. Il valore predefinito è disattiva to (0).
`log_queries_not_using_indexes`	Registra tutte le query nel log delle query lente che non utilizzano un indice. Il valore predefini to è disattivato (0). Imposta su 1 per attivare questo log.
`long_query_time`	Impedisce che le query in esecuzione rapida vengano registrate nel log delle query lente. Può essere impostato su un valore variabile compreso tra 0 e 31536000. Il valore predefini to è 0 (non attivo).
`server_audit_events`	L'elenco degli eventi da catturare nei log. I valori supportati sono `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` e `TABLE`.
`server_audit_logging`	Imposta su 1 per attivare la registrazione di controllo del server. Se si attiva questa opzione, è possibile specificare gli eventi di controllo a cui inviare, CloudWatch elencandoli nel `server_audit_events` parametro.
`slow_query_log`	Crea un log di query lente. Imposta su 1 per attivare il log di query lente. Il valore predefinito è disattivato (0).

Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.

Per Aurora PostgreSQL, puoi abilitare i seguenti log sul tuo Aurora Serverless v2 Istanze DB.

Aurora PostgreSQL	Descrizione
`log_connections`	Registra ogni connessione riuscita.
`log_disconnections`	Registra la fine di una sessione, compresa la durata.
`log_lock_waits`	Il valore predefinito è 0 (disattivato). Imposta su 1 per registrare le attese di blocco.
`log_min_duration_statement`	La durata minima (in millisecondi) per l'esecuzione di un'istruzione prima della registrazione.
`log_min_messages`	Imposta i livelli dei messaggi che vengono registrati. I valori supportati sono `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Per registrare i dati delle prestazioni nel log `postgres`, imposta il valore su `debug1`.
`log_temp_files`	Registra l'utilizzo di file temporanei che si trovano al di sopra dei kilobyte (kB) specificati.
`log_statement`	Controlla le istruzioni SQL specifiche che vengono registrate. I valori supportati sono `none`, `ddl`, `mod` e `all`. Il valore predefinito è `none`.

Argomenti

- Registrazione con Amazon CloudWatch
- Visualizzazione Aurora Serverless v2 log in Amazon CloudWatch
- Capacità di monitoraggio con Amazon CloudWatch
- Monitoraggio Aurora Serverless v2 mettere in pausa e riprendere l'attività

Registrazione con Amazon CloudWatch

Dopo aver utilizzato la procedura Registrazione per Aurora Serverless v2 per scegliere quali log del database attivare, puoi scegliere quali log caricare («pubblicare») su Amazon. CloudWatch

Puoi usare Amazon CloudWatch per analizzare i dati di log, creare allarmi e visualizzare i parametri. Per impostazione predefinita, i log degli errori per Aurora Serverless v2 sono abilitati e caricati automaticamente su. CloudWatch Puoi anche caricare altri registri da Aurora Serverless v2 Istanze DB su. CloudWatch

Quindi scegli in quale di questi registri caricare CloudWatch, utilizzando le impostazioni di esportazione del registro in AWS Management Console o l'`--enable-cloudwatch-logs-exports`opzione in. AWS CLI

Puoi scegliere quale dei tuoi Aurora Serverless v2 registri su cui caricare. CloudWatch Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.

Come per qualsiasi tipo di cluster database Aurora, non è possibile modificare il gruppo di parametri del cluster database predefinito. Crea invece il tuo gruppo di parametri del cluster database in base a un parametro predefinito per il cluster database e il tipo di motore. Ti consigliamo di creare un gruppo di parametri del cluster DB personalizzato prima di creare il Aurora Serverless v2 Cluster DB, in modo che sia possibile scegliere quando creare un database sulla console.

> **Note**
>
> In Aurora Serverless v2, è possibile creare sia cluster DB che gruppi di parametri DB. Ciò contrasta con Aurora Serverless v1, dove è possibile creare solo gruppi di parametri del cluster DB.

Visualizzazione Aurora Serverless v2 log in Amazon CloudWatch

Dopo aver utilizzato la procedura descritta in Registrazione con Amazon CloudWatch per scegliere i registri di database attivare, è possibile visualizzare il contenuto di tali registri.

Per ulteriori informazioni sull'utilizzo CloudWatch con i log di Aurora MySQL e Aurora PostgreSQL, vedere e. Monitoraggio degli eventi di registro in Amazon CloudWatch Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs CloudWatch

Per visualizzare i log relativi ai Aurora Serverless v2 cluster di database

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli il tuo Regione AWS.
3. Scegli Log groups (Gruppi di log).
4. Scegli il tuo Aurora Serverless v2 Registro del cluster DB dall'elenco. Il modello di denominazione dei log è il seguente.

```
/aws/rds/cluster/cluster-name/log_type
```

> **Note**
>
> Compatibile con Aurora MySQL Aurora Serverless v2 Nei cluster DB, il log degli errori include gli eventi di scalabilità del buffer pool anche in assenza di errori.

Capacità di monitoraggio con Amazon CloudWatch

Con Aurora Serverless v2, è possibile utilizzare CloudWatch per monitorare la capacità e l'utilizzo di tutti i Aurora Serverless v2 istanze DB nel cluster. Puoi visualizzare le metriche a livello di istanza per verificare la capacità di ciascuna Aurora Serverless v2 Istanza DB man mano che si espande verso l'alto e verso il basso. Puoi inoltre confrontare i parametri a livello di capacità con altri parametri per verificare l'impatto delle modifiche dei carichi di lavoro sul consumo di risorse. Ad esempio, puoi confrontare `ServerlessDatabaseCapacity` con `DatabaseUsedMemory`, `DatabaseConnections` e `DMLThroughput` per valutare la risposta del cluster di database durante le operazioni. Per informazioni dettagliate sulle metriche relative alla capacità applicabili a Aurora Serverless v2, consulta CloudWatch Parametri Amazon importanti per Aurora Serverless v2.

Per monitorare il tuo Aurora Serverless v2 Capacità del cluster DB

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Seleziona Parametri. Tutti i parametri disponibili vengono visualizzati come schede nella console, raggruppate in base al nome del servizio.
3. Seleziona RDS.

4. (Facoltativo) Utilizza la casella di ricerca per trovare le metriche particolarmente importanti per Aurora Serverless v2:`ServerlessDatabaseCapacity`, `ACUUtilizationCPUUtilization`, `eFreeableMemory`.

Ti consigliamo di configurare una CloudWatch dashboard per monitorare i Aurora Serverless v2 Capacità del cluster DB utilizzando le metriche relative alla capacità. Per scoprire come, consulta Creazione di dashboard con. CloudWatch

Per ulteriori informazioni sull'uso di Amazon CloudWatch con Amazon Aurora, consulta. Pubblicazione dei log MySQL di Amazon Aurora su Amazon Logs CloudWatch

Monitoraggio Aurora Serverless v2 mettere in pausa e riprendere l'attività

Aurora scrive un file di registro separato per Aurora Serverless v2 Istanze DB con pausa automatica abilitata. Aurora scrive nel registro per ogni intervallo di 10 minuti in cui l'istanza non viene messa in pausa. Aurora conserva fino a sette di questi registri, ruotati ogni giorno. Il file di registro corrente viene denominato `instance.log` e i registri più vecchi vengono denominati utilizzando lo schema. `instance.`*`YYYY-MM-DD`*`.`*`N`*`.log`

Questo registro è abilitato di default per Aurora Serverless v2 Istanze DB con pausa automatica abilitata. È possibile visualizzare il contenuto di questo registro in AWS Management Console o utilizzando l' AWS CLI API o RDSP. Al momento, non è possibile caricare questo registro su. CloudWatch

Gli eventi elencati in Eventi di istanza database forniscono una panoramica di alto livello della pausa e della ripresa dell'attività, come i seguenti:

- Quando l'istanza inizia a mettersi in pausa e quando finisce la pausa.
- Quando l'istanza inizia a riprendere e quando finisce di riprenderla.
- Casi in cui l'istanza ha tentato di mettersi in pausa, ma alcune condizioni ne hanno impedito la pausa.

`instance.log`Fornisce dettagli più dettagliati sui motivi per cui un Aurora Serverless v2 l'istanza potrebbe o non potrebbe essere in grado di mettersi in pausa.

Il registro potrebbe indicare che un'istanza è stata ripresa per diversi motivi:

- attività dell'utente: una richiesta di connessione al database. Potrebbe provenire da una sessione client interattiva, da una chiamata RDS Data API o da una richiesta di download dei log dall'istanza.
- attività in background: un'ampia categoria che include tutti i motivi per cui Aurora riprende un'istanza. Ad esempio, quando una richiesta di connessione a un'istanza di lettura causa la ripresa dell'istanza writer, il registro del lettore indica l'attività dell'utente, mentre il registro per il writer classifica tale richiesta di ripristino come attività in background. Per i motivi per cui Aurora potrebbe riprendere un'istanza diversa da una richiesta di connessione utente, vedi. Ripresa di una pausa automatica Aurora Serverless v2 istanza

Quando Aurora non è a conoscenza di alcuna condizione che possa impedire l'interruzione dell'istanza alla scadenza dell'intervallo di pausa automatica, scrive periodicamente un messaggio informativo nel registro. Per i cluster con una sola istanza DB, il log contiene questo messaggio:

```
[INFO] No auto-pause blockers registered since time
```

Per i cluster con più istanze DB, il messaggio è leggermente diverso. Questo perché uno scrittore potrebbe non essere in grado di mettere in pausa a causa dell'attività su una qualsiasi delle istanze del lettore. Se l'attività sul lettore termina prima della scadenza dell'intervallo di pausa automatica sullo scrittore, lo scrittore è in grado di mettere in pausa all'ora prevista.

```
[INFO] No auto-pause blockers registered since time.
Database might be required to maintain compute capacity for high availability.
```

Se inizia un'operazione di pausa, ma arriva una nuova richiesta di connessione al database prima che l'istanza finisca la sospensione, il log contiene questo messaggio:

```
[INFO] Unable to pause database due to a new database activity
```

Se Aurora viene a conoscenza di condizioni che impediscono definitivamente la sospensione dell'istanza, il registro contiene questo messaggio che elenca tutte queste condizioni:

```
[INFO] Auto-pause blockers registered since time: list_of_conditions
```

In questo modo, Aurora non ti impedisce di attivare la replica, l'integrazione zero-ETL, Aurora Global Database e così via in combinazione con la funzione di pausa automatica. Il registro informa l'utente quando l'uso di tali funzionalità potrebbe impedire l'effetto della pausa automatica.

Di seguito sono riportati i motivi per cui un Aurora Serverless v2 l'istanza potrebbe superare l'intervallo di timeout della pausa automatica, ma impedirne la pausa:

- attività del database prima del timeout di pausa automatica: l'istanza DB ha ricevuto una richiesta di connessione prima della scadenza dell'intervallo di timeout.
- membro del database globale: se il cluster DB fa parte di un database globale Aurora, Aurora Serverless v2 le istanze nel cluster non si interrompono. Un cluster può passare da un cluster autonomo a un cluster di database globale. Pertanto, le istanze precedentemente sospese automaticamente potrebbero interrompere la pausa e riportare questo motivo nel registro. Una volta che un cluster diventa membro di un database globale, non torna a essere un cluster autonomo finché non viene scollegato esplicitamente. Il cluster primario è ancora considerato parte del database globale anche se si scollegano tutti i cluster secondari.
- funzionalità di replica configurata: l'istanza Writer DB ha abilitato la replica specifica del motore, la replica binlog per MySQL o la replica logica per PostgreSQL. Questa condizione potrebbe essere causata anche dall'utilizzo di un'altra funzionalità di Aurora che richiede l'attivazione della replica, come le integrazioni zero-ETL o i Database Activity Streams (DAS).
- ritardo di backup continuo: se il sistema di storage Aurora non ha completato l'applicazione delle modifiche allo storage fino al momento corrente, l'istanza writer non si ferma finché non recupera il ritardo. Questa condizione riguarda solo l'istanza writer e dovrebbe essere relativamente breve.
- intervento di assistenza o manutenzione da parte del cliente: se viene avviata un'operazione di manutenzione, l'istanza DB non verrà nuovamente messa in pausa fino al termine dell'operazione. Questa condizione include un'ampia varietà di operazioni che potrebbero essere avviate dall'utente o da Aurora, come aggiornamenti, clonazione, modifica delle impostazioni di configurazione, aggiornamenti, download di file di registro e così via. Questo evento si verifica anche quando richiedi di eliminare un'istanza e Aurora riprende brevemente l'istanza come parte del meccanismo di eliminazione.
- problema di comunicazione transitoria: se Aurora non è in grado di determinare se la configurazione di scalabilità ha attualmente un'impostazione di capacità minima pari a zero ACUs, non mette in pausa l'istanza. Si prevede che si tratti di un evento molto raro.

Prestazioni e dimensionamento per Aurora Serverless v2

Le seguenti procedure ed esempi mostrano come impostare l'intervallo di capacità per i cluster Aurora Serverless v2 e le istanze database associate. È inoltre possibile utilizzare le procedure

seguenti per monitorare il livello di utilizzo delle istanze database. Sarà quindi possibile utilizzare i risultati per determinare se è necessario aumentare o ridurre l'intervallo di capacità.

Prima di utilizzare queste procedure, assicurati di conoscere bene il funzionamento del dimensionamento in Aurora Serverless v2. Il funzionamento del dimensionamento è diverso da quello in Aurora Serverless v1. Per informazioni dettagliate, consultare Aurora Serverless v2 scalabilità.

Indice

- Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora
  - Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster
  - Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster
  - Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL
  - Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL
- Uso di gruppi di parametri per Aurora Serverless v2
  - Valori di parametro predefiniti
  - Numero massimo connessioni per Aurora Serverless v2
  - Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2
  - Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2
- Evitare gli errori out-of-memory
- CloudWatch Parametri Amazon importanti per Aurora Serverless v2
  - In che modo Aurora Serverless v2 le metriche si applicano alla fattura AWS
  - Esempi di CloudWatch comandi per Aurora Serverless v2 le metriche
- Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni
- Risoluzione dei problemi di capacità di Aurora Serverless v2

Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora

Con le istanze database Aurora Serverless v2, è possibile impostare l'intervallo di capacità applicabile a tutte le istanze database nel cluster di database mentre aggiungi la prima Istanza database Aurora Serverless v2 al cluster di database. Per eseguire questa procedura, consulta Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.

È inoltre possibile modificare l'intervallo di capacità per un cluster esistente. Nelle sezioni seguenti viene descritto più dettagliatamente come scegliere i valori minimi e massimi appropriati e cosa succede quando si modifica l'intervallo di capacità. Ad esempio, la modifica dell'intervallo di capacità può modificare i valori di default di alcuni parametri di configurazione. L'applicazione di tutte le modifiche ai parametri può richiedere il riavvio di ciascuna istanza database Aurora Serverless v2.

Argomenti

- Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster
- Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster
- Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL
- Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL

Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster

Si potrebbe essere portati a scegliere sempre 0,5 come impostazione minima della capacità per Aurora Serverless v2. Questo valore consente all'istanza DB di ridimensionarsi fino alla capacità minima quando è completamente inattiva, pur rimanendo attiva. È inoltre possibile abilitare il comportamento di pausa automatica specificando una capacità minima di 0 ACUs, come spiegato in. Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2 Tuttavia, a seconda di come si utilizza il cluster e delle altre impostazioni configurate, una capacità minima diversa potrebbe essere la più efficace. Considera i seguenti fattori nella scelta dell'impostazione della capacità minima:

- La velocità di dimensionamento per un'istanza database Aurora Serverless v2 dipende dalla sua capacità attuale. Maggiore è la capacità attuale, più veloce sarà il relativo dimensionamento verso l'alto. Se è necessario disporre di un rapido dimensionamento verso l'alto in modo che l'istanza database raggiunga capacità molto elevate, è consigliabile impostare la capacità minima su un valore in cui la velocità di dimensionamento soddisfi le esigenze specifiche.
- Se in genere si modifica la classe delle istanze database in previsione di un carico di lavoro particolarmente elevato o ridotto, è possibile utilizzare tale esperienza per calcolare una stima approssimativa dell'equivalente intervallo di capacità per Aurora Serverless v2. Per determinare la dimensione della memoria da utilizzare in periodi di traffico ridotto, consulta Specifiche hardware per le classi di istanza database per Aurora.

  Ad esempio, supponiamo di utilizzare la classe di istanza database db.r6g.xlarge quando il cluster ha un carico di lavoro ridotto. Questa classe di istanza database dispone di 32 GiB di

memoria. Pertanto, è possibile specificare un'impostazione minima dell'unità di capacità Aurora (ACU) pari a 16 per impostare un'istanza database Aurora Serverless v2 in grado di supportare un dimensionamento verticale avente all'incirca la stessa capacità. Questo perché ogni ACU corrisponde a circa 2 GiB di memoria. È possibile specificare un valore leggermente inferiore per consentire all'istanza database un'ulteriore riduzione verticale in caso di sottoutilizzo dell'istanza databse db.r6g.xlarge.

- Se l'applicazione funziona in modo più efficiente quando le istanze database contengono una certa quantità di dati nella cache del buffer, valuta la possibilità di specificare un'impostazione ACU minima in cui la memoria è sufficientemente grande da contenere i dati a cui si accede più di frequente. In caso contrario, alcuni dati vengono cancellati dalla cache del buffer quando le istanze database Aurora Serverless v2 vengono ridotte verticalmente a una dimensione di memoria inferiore. Quando le istanze database vengono successivamente dimensionate verso l'alto, le informazioni vengono lette nuovamente nella cache del buffer. Se la quantità di I/O per riportare i dati nella cache del buffer è rilevante, potrebbe essere più efficace scegliere un valore ACU minimo più alto.
- Se per la maggior parte del tempo le istanze database Aurora Serverless v2 vengono eseguite a una determinata capacità, valuta la possibilità di specificare un'impostazione di capacità minima inferiore, ma non di molto, a quella di base. Le istanze database Aurora Serverless v2 possono calcolare in modo più efficace una stima della quantità e velocità del dimensionamento verso l'alto (aumento) quando la capacità corrente non è sensibilmente inferiore alla capacità richiesta.
- Se il carico di lavoro con provisioning ha requisiti di memoria troppo elevati per classi di istanze database di piccole dimensioni come T3 o T4g, scegli un'impostazione ACU minima che fornisca memoria paragonabile a un'istanza database R5 o R6g.

  In particolare, raccomandiamo la seguente capacità minima per l'utilizzo con le caratteristiche specificate (queste raccomandazioni sono soggette a modifiche):

  - Performance Insights — 2 ACUs
  - Database globali Aurora — 8 ACUs (si applica solo al database primario) Regione AWS
- In Aurora, la replica avviene a livello di storage, quindi la capacità del lettore non influisce direttamente sulla replica. Tuttavia, per le istanze DB Aurora Serverless v2 Reader con scalabilità indipendente, assicurati che la capacità minima sia sufficiente per gestire i carichi di lavoro durante i periodi di intensa attività di scrittura per evitare la latenza delle query. Se le istanze Reader DB nei livelli di promozione 2—15 presentano problemi di prestazioni, valuta la possibilità di aumentare la capacità minima del cluster. Per informazioni dettagliate in merito alla decisione se le istanze database di lettura vengono dimensionate assieme alle istanze di scrittura oppure in modo indipendente, consulta Scelta del livello di promozione per un Aurora Serverless v2 reader.

- Se disponi di un cluster DB con istanze Aurora Serverless v2 Reader DB, i lettori non rispondono alla stessa scala dell'istanza DB Writer quando il livello di promozione dei lettori non è 0 o 1. In tal caso, l'impostazione di una capacità minima ridotta può comportare un eccessivo ritardo nella replica. Ciò è dovuto al fatto che le istanze di lettura potrebbero non avere capacità sufficiente per applicare le modifiche dell'istanza di scrittura quando il database è occupato. Ti consigliamo di impostare la capacità minima su un valore che rappresenti una quantità di memoria e CPU paragonabile a quella dell'istanza Writer DB.
- Il valore del `max_connections` parametro per le istanze Aurora Serverless v2 DB si basa sulla dimensione della memoria derivata dal valore massimo ACUs. Tuttavia, quando si specifica una capacità minima di 0 o 0,5 ACUs su istanze DB compatibili con PostgreSQL, il valore massimo di è limitato a 2.000. `max_connections`

  Se vuoi utilizzare il cluster Aurora PostgreSQL per un carico di lavoro con un numero elevato di connessioni, è consigliabile utilizzare un'impostazione ACU minima pari a 1 o superiore. Per informazioni dettagliate su come Aurora Serverless v2 gestisce il parametro di configurazione `max_connections`, consulta Numero massimo connessioni per Aurora Serverless v2.
- Il tempo necessario a un'istanza database Aurora Serverless v2 per il dimensionamento dalla capacità minima alla capacità massima dipende dalla differenza tra i valori ACU minimi e massimi. Quando la capacità corrente dell'istanza database è elevata, il dimensionamento verso l'alto di Aurora Serverless v2 avviene con incrementi più grandi rispetto a quando l'istanza database inizia da una capacità ridotta. Pertanto, se specifichi una capacità massima relativamente grande e l'istanza database rimane per la maggior parte del tempo vicino a tale capacità, è consigliabile aumentare l'impostazione ACU minima. In questo modo, un'istanza database inattiva può usufruire del dimensionamento verso l'alto e raggiungere più rapidamente la capacità massima.

Scelta dell'impostazione Aurora Serverless v2 massima della capacità per un cluster

Si potrebbe essere tentati di scegliere sempre un valore elevato per l'impostazione della capacità Aurora Serverless v2 massima. Un valore elevato per la capacità massima consente all'istanza database un dimensionamento massimo verso l'alto quando è in esecuzione un carico di lavoro con un uso intensivo di risorse. Un valore basso evita il rischio di dover sostenere di addebiti imprevisti. A seconda di come si utilizza il cluster e delle altre impostazioni configurate, il valore più efficace potrebbe essere maggiore o minore basso rispetto a quello originariamente individuato. Considera i seguenti fattori nella scelta dell'impostazione della capacità massima:

- La capacità massima deve essere almeno pari alla capacità minima. Puoi impostare lo stesso valore per la capacità minima e la capacità massima. Tuttavia, in questo caso la capacità non

aumenta né diminuisce mai. Pertanto, a parte gli scenari di test, l'uso di valori identici per la capacità minima e massima non è appropriato.

- La capacità massima deve essere superiore a 0,5. ACUs Nella maggior parte dei casi è possibile impostare la capacità minima e massima su un valore identico. Tuttavia, non è possibile specificare 0,5 sia per la capacità minima che per la la capacità massima. Utilizza un valore pari o superiore a 1 per la capacità massima.
- Se in genere modifichi la classe delle istanze database in previsione di un carico di lavoro particolarmente elevato o ridotto, è possibile utilizzare tale esperienza per calcolare la stima dell'equivalente intervallo di capacità Aurora Serverless v2. Per determinare la dimensione della memoria da utilizzare in periodi di traffico elevato, consulta Specifiche hardware per le classi di istanza database per Aurora.

  Ad esempio, supponiamo di utilizzare la classe di istanza database db.r6g.4xlarge quando il cluster ha un carico di lavoro elevato. Questa classe di istanza database dispone di 128 GiB di memoria. Pertanto, è possibile specificare un'impostazione ACU massima di 64 per impostare un'istanza database Aurora Serverless v2 in grado di eseguire un dimensionamento verso l'alto fino a circa la stessa capacità. Questo perché ogni ACU corrisponde a circa 2 GiB di memoria. È possibile specificare un valore leggermente più alto per consentire all'istanza database un ulteriore dimensionamento verso l'alto nel caso in cui l'istanza database db.r6g.4xlarge a volte non abbia capacità sufficiente per gestire efficacemente il carico di lavoro.
- Se hai un limite di budget per l'utilizzo del database, scegli un valore compreso in tale limite anche se tutte le istanze database Aurora Serverless v2 vengono eseguite sempre alla capacità massima. Ricorda che quando hain istanze database Aurora Serverless v2 nel cluster, la capacità Aurora Serverless v2 massima teorica che il cluster può consumare in qualsiasi momento è nvolte l'impostazione ACU massima per il cluster. La quantità effettiva consumata potrebbe essere inferiore, ad esempio se alcune istanze di lettura vengono dimensionate in modo indipendente rispetto all'istanza di scrittura.
- Se si utilizzano istanze database Aurora Serverless v2 di lettura per alleggerire parte del carico di lavoro di sola lettura dell'istanza database di scrittura, è possibile scegliere un'impostazione di capacità massima inferiore. Si procede in questo senso se il dimensionamento di ogni istanza database di lettura non deve raggiungere un valore così alto, come se il cluster contenga solo un'unica istanza database.
- Supponiamo di voler evitare un uso eccessivo a causa di parametri del database configurati in modo errato o di query inefficienti nell'applicazione. In tal caso, è possibile evitare un uso eccessivo accidentale scegliendo un'impostazione di capacità massima inferiore a quella più alta possibile.

- Se i picchi dovuti alla reale attività degli utenti sono rari ma si verificano, è possibile tenerne conto quando si sceglie l'impostazione della capacità massima. Se è prioritario che l'applicazione continui a funzionare con prestazioni e scalabilità complete, è possibile specificare un'impostazione di capacità massima superiore a quella osservata durante il normale utilizzo. Se l'applicazione può essere eseguita con una capacità di throughput effettiva ridotta durante picchi di attività molto estremi, è possibile scegliere un'impostazione di capacità massima leggermente inferiore. Assicurati di scegliere un'impostazione che disponga ancora di memoria e risorse CPU sufficienti per mantenere l'applicazione in esecuzione.
- Se attivi impostazioni nel cluster che aumentano l'utilizzo della memoria per ogni istanza database, considera tale capacità di memoria quando decidi il valore ACU massimo. Tali impostazioni includono quelle per Approfondimenti sulle prestazioni, le query parallele Aurora MySQL, lo schema di prestazioni Aurora MySQL e la replica del registro binario Aurora MySQL. Assicurati che il valore ACU massimo consenta un dimensionamento verso l'alto delle istanze di database Aurora Serverless v2 tale da gestire il carico di lavoro quando vengono utilizzate queste funzionalità. Per informazioni sulla risoluzione dei problemi causati dalla combinazione di un'impostazione ACU massima bassa e le funzionalità Aurora che impongono il sovraccarico della memoria, consulta Evitare gli errori out-of-memory.

Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL

L' AWS CLI esempio seguente mostra come aggiornare l'intervallo ACU per le istanze Aurora Serverless v2 DB in un cluster Aurora MySQL esistente. Inizialmente, l'intervallo di capacità per il cluster è compreso tra 8 e 32. ACUs

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

L'istanza DB è inattiva e ridotta a 8. ACUs A questo punto, le seguenti impostazioni relative alla capacità si applicano all'istanza database. Per rappresentare la dimensione del pool di buffer in unità facilmente leggibili, si divide questo valore per 2 elevato alla 30, ottenendo in questo modo una misurazione in gibibyte (GiB). Ciò è dovuto al fatto che le misurazioni relative alla memoria per Aurora utilizzano unità basate su potenze di 2, non potenze di 10.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|                9294577664 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|   8.65625 |
+-----------+
1 row in set (0.00 sec)
```

Successivamente, si modifica l'intervallo di capacità per il cluster. Al termine dell'esecuzione del comando `modify-db-cluster`, l'intervallo ACU per il cluster è 12.5-80.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

La modifica dell'intervallo di capacità ha causato modifiche ai valori di default di alcuni parametri di configurazione. Aurora può applicare immediatamente alcuni di questi nuovi valori di default. Tuttavia, alcune modifiche ai parametri hanno effetto solo dopo il riavvio. Lo stato `pending-reboot` indica che è necessario un riavvio per applicare alcune modifiche ai parametri.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

A questo punto, il cluster è inattivo e l'istanza `serverless-v2-instance-1` DB consuma 12,5. ACUs Il parametro `innodb_buffer_pool_size` è già modificato in base alla capacità corrente dell'istanza database. Il parametro `max_connections` riflette ancora il valore della capacità massima precedente. La reimpostazione di questo valore richiede il riavvio dell'istanza database.

> **Note**
>
> Se si imposta il `max_connections` parametro direttamente in un gruppo di parametri DB personalizzato, non è necessario il riavvio.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               15572402176 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
```

```
+---------------+
| gibibytes     |
+---------------+
| 14.5029296875 |
+---------------+
1 row in set (0.00 sec)
```

L'istanza viene riavviata e si attende che torni di nuovo disponibile.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Lo stato `pending-reboot` viene cancellato. Il valore `in-sync` conferma che Aurora ha applicato tutte le modifiche dei parametri in sospeso.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Il parametro `innodb_buffer_pool_size` è aumentato fino alla sua dimensione finale per un'istanza database inattiva. Il parametro `max_connections` è aumentato in base a un valore derivato dal valore ACU massimo. La formula usata da Aurora per `max_connections` comporta un aumento di 1.000 quando la dimensione della memoria raddoppia.

```
mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
```

```
|               16139681792 |
+----------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|  15.03125 |
+-----------+
1 row in set (0.00 sec)

mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              4000 |
+-------------------+
1 row in set (0.00 sec)
```

Impostiamo l'intervallo di capacità su 0,5-128 e riavviamo l' ACUsistanza DB. A questo punto, l'istanza database inattiva ha una dimensione della cache del buffer inferiore a 1 GiB, quindi la misurazione viene eseguita in mebibyte (MiB). Il valore di 5000 del parametro `max_connections` è derivato dalla dimensione della memoria dell'impostazione di capacità massima.

```
mysql> select @@innodb_buffer_pool_size / pow(2,20) as mebibytes, @@max_connections;
+-----------+-------------------+
| mebibytes | @@max_connections |
+-----------+-------------------+
|       672 |              5000 |
+-----------+-------------------+
1 row in set (0.00 sec)
```

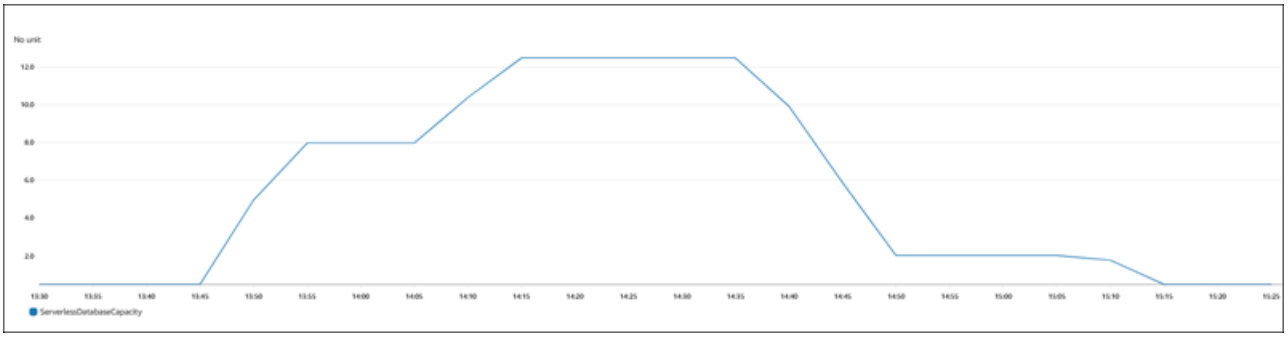
Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL

L'esempio per l'interfaccia della linea di comando seguente illustra come aggiornare l'intervallo ACU per le istanze database Aurora Serverless v2 in un cluster Aurora PostgreSQL esistente.

1. L'intervallo di capacità del cluster iniziale è compreso tra 0,5 e 1 ACU.
2. Modificate l'intervallo di capacità da 8 a 32. ACUs

3. Modificate l'intervallo di capacità a 12,5—80. ACUs
4. Modificate l'intervallo di capacità portandolo a 0,5-128. ACUs
5. Riporta la capacità all'intervallo iniziale compreso tra 0,5 e 1 ACU.

La figura seguente mostra le variazioni di capacità in Amazon CloudWatch.



L'istanza DB è inattiva e ridimensionata a 0,5. ACUs A questo punto, le seguenti impostazioni relative alla capacità si applicano all'istanza database.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Successivamente, si modifica l'intervallo di capacità per il cluster. Al termine dell'esecuzione del comando `modify-db-cluster`, l'intervallo ACU per il cluster è 8.0-32.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

La modifica dell'intervallo di capacità comporta modifiche ai valori di default di alcuni parametri di configurazione. Aurora può applicare immediatamente alcuni di questi nuovi valori di default. Tuttavia, alcune modifiche ai parametri hanno effetto solo dopo il riavvio. Lo stato `pending-reboot` indica che è necessario un riavvio per applicare le modifiche ad alcuni parametri.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

A questo punto, il cluster è inattivo e l'istanza `serverless-v2-instance-1` DB consuma 8.0. ACUs Il parametro `shared_buffers` è già modificato in base alla capacità corrente dell'istanza database. Il parametro `max_connections` riflette ancora il valore della capacità massima precedente. La reimpostazione di questo valore richiede il riavvio dell'istanza database.

> **Note**
>
> Se si imposta il `max_connections` parametro direttamente in un gruppo di parametri DB personalizzato, non è necessario il riavvio.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

L'istanza database viene riavviata e si attende che torni di nuovo disponibile.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Dopo il riavvio dell'istanza, lo stato `pending-reboot` è cancellato. Il valore `in-sync` conferma che Aurora ha applicato tutte le modifiche dei parametri in sospeso.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Dopo il riavvio, `max_connections` mostra il valore della nuova capacità massima.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Successivamente, modifichiamo l'intervallo di capacità per il cluster portandolo a ACUs 12,5-80.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

A questo punto, il cluster è inattivo e l'istanza `serverless-v2-instance-1` DB consuma 12,5. ACUs Il parametro `shared_buffers` è già modificato in base alla capacità corrente dell'istanza database. Il valore `max_connections` è ancora 5000.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Si esegue nuovamente il riavvio, ma i valori del parametro rimangono invariati. Questo perché `max_connections` ha un valore massimo di 5000 per un cluster di database Aurora Serverless v2 che esegue Aurora PostgreSQL.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Ora impostiamo l'intervallo di capacità da 0,5 a 128. ACUs Il cluster DB si ridimensiona fino a 10 ACUs, quindi a 2. Riavviamo l'istanza database.

```
postgres=> show max_connections;
 max_connections
-----------------
 2000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Il `max_connections` valore per le istanze Aurora Serverless v2 DB si basa sulla dimensione della memoria derivata dal valore massimo. ACUs Tuttavia, quando si specifica una capacità minima di 0 o 0,5 ACUs su istanze DB compatibili con PostgreSQL, il valore massimo di è limitato a 2.000. `max_connections`

Ora riportiamo la capacità all'intervallo iniziale compreso tra 0,5 e 1 ACU e riavviamo l'istanza database. Il parametro `max_connections` è tornato al suo valore originale.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Uso di gruppi di parametri per Aurora Serverless v2

Quando crei il tuo cluster di database Aurora Serverless v2, è possibile scegliere un motore del database Aurora specifico e un gruppo di parametri del cluster di database associato. Se non si ha familiarità con il modo in cui Aurora utilizza i gruppi di parametri per applicare le impostazioni di

configurazione in modo coerente tra cluster, consulta . Tutte queste procedure per la creazione, la modifica, l'applicazione e altre azioni per i gruppi di parametri sono valide per Aurora Serverless v2.

La funzione del gruppo di parametri funziona in genere allo stesso modo tra cluster con provisioning e cluster contenenti istanze database Aurora Serverless v2:

- Il gruppo di parametri del cluster di database include anche i valori di default per tutti i parametri a livello di istanza.
- È possibile sovrascrivere alcuni parametri per istanze database specifiche specificando un gruppo personalizzato di parametri del database per tali istanze database. È possibile eseguire questa operazione durante il debug o l'ottimizzazione delle prestazioni per istanze database specifiche. Ad esempio, supponiamo di avere un cluster contenente alcune istanze database Aurora Serverless v2 e alcune istanze database con provisioning. In questo caso, è possibile specificare alcuni parametri diversi per le istanze database con provisioning utilizzando un gruppo personalizzato di parametri del database.
- Per Aurora Serverless v2, è possibile utilizzare tutti i parametri con il valore `provisioned` nell'attributo `SupportedEngineModes` nel gruppo di parametri. In Aurora Serverless v1, è possibile utilizzare solo il sottoinsieme di parametri con `serverless` nellì'attributo `SupportedEngineModes`.

Argomenti

- Valori di parametro predefiniti
- Numero massimo connessioni per Aurora Serverless v2
- Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2
- Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2

Valori di parametro predefiniti

La sostanziale differenza tra le istanze database con provisioning e le istanze database Aurora Serverless v2 risiede nel fatto che Aurora sovrascrive tutti i valori dei parametri personalizzati per determinati parametri correlati alla capacità dell'istanza database. I valori dei parametri personalizzati si applicano ancora a tutte le istanze database con provisioning nel cluster. Per ulteriori informazioni su come le istanze database Aurora Serverless v2 interpretano i parametri dei gruppi di parametri Aurora, consulta Parametri di configurazione per i cluster Aurora. Per i parametri specifici sostituiti

da Aurora Serverless v2, consulta Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2 e Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2.

È possibile ottenere un elenco di valori predefiniti per i gruppi di parametri predefiniti per i vari motori Aurora DB utilizzando il comando describe-db-cluster-parametersCLI e interrogando il. Regione AWS Di seguito sono riportati i valori che è possibile utilizzare per le opzioni `--db-parameter-group-family` e `-db-parameter-group-name` per le versioni del motore compatibili con Aurora Serverless v2.

Motore del database e versione	Famiglia di gruppi di parametri	Nome del gruppo di parametri di default
Aurora MySQL versione 3	`aurora-mysql8.0`	`default.aurora-mysql8.0`
Aurora PostgreSQL versione 13.x	`aurora-postgresql13`	`default.aurora-postgresql13`
Aurora PostgreSQL versione 14.x	`aurora-postgresql14`	`default.aurora-postgresql14`
Aurora PostgreSQL versione 15.x	`aurora-postgresql15`	`default.aurora-postgresql15`
Aurora PostgreSQL versione 16.x	`aurora-postgresql16`	`default.aurora-postgresql16`
Aurora PostgreSQL versione 17.x	`aurora-postgresql17`	`default.aurora-postgresql17`

Nell'esempio seguente viene recuperato un elenco di parametri dal gruppo di cluster di database di default per Aurora MySQL versione 3 e Aurora PostgreSQL 13. Queste sono le versioni di Aurora MySQL e Aurora PostgreSQL usate con Aurora Serverless v2.

Per, o: Linux macOS Unix

```
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql8.0 \
```

```
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text

aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-postgresql13 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text
```

Per Windows:

```
aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-mysql8.0 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text

aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-postgresql13 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text
```

Numero massimo connessioni per Aurora Serverless v2

Per Aurora MySQL e Aurora PostgreSQL, le istanze database Aurora Serverless v2 contengono la costante del parametro `max_connections` in modo che le connessioni non vengano interrotte in caso di riduzione verticale dell'istanza database. Il valore predefinito per questo parametro è derivato da una formula che si basa sulle dimensioni della memoria dell'istanza database. Per informazioni dettagliate sulla formula e sui valori di default per le classi di istanza database con provisioning, consulta Numero massimo di connessioni a un'istanza database Aurora MySQL e Numero massimo di connessioni a un'istanza database Aurora PostgreSQL.

Quando Aurora Serverless v2 valuta la formula, utilizza la dimensione della memoria in base alle unità di capacità massima di Aurora ACUs () per l'istanza DB, non il valore ACU corrente. Se modifichi il valore di default, si consiglia di utilizzare una variante della formula anziché specificare

un valore costante. In questo modo, Aurora Serverless v2 può utilizzare un'impostazione adeguata in base alla capacità massima.

Quando si modifica la capacità massima di un cluster di database Aurora Serverless v2, è necessario riavviare le istanze database Aurora Serverless v2 per aggiornare il valore `max_connections`. Questo perché `max_connections` è un parametro statico per Aurora Serverless v2.

La tabella seguente mostra i valori predefiniti di `max_connections` per Aurora Serverless v2 in base al valore ACU massimo.

Massimo ACUs	Numero massimo di connessioni predefinito in Aurora MySQL	Numero massimo di connessioni predefinito in Aurora PostgreSQL
1	90	189
4	135	823
8	1.000	1.669
16	2.000	3.360
32	3.000	5.000
64	4.000	5.000
128	5.000	5.000
192	6.000	5.000
256	6.000	5.000

> **Note**
>
> Il `max_connections` valore per le istanze Aurora Serverless v2 DB si basa sulla dimensione della memoria derivata dal valore massimo. ACUs Tuttavia, quando si specifica una capacità minima di 0 o 0,5 ACUs su istanze DB compatibili con PostgreSQL, il valore massimo di è limitato a 2.000. `max_connections`

Per esempi specifici che mostrano come `max_connections` cambia con il valore ACU massimo, consulta Esempio: modificare l'intervallo di capacità Aurora Serverless v2 di un cluster Aurora MySQL e Esempio: modificare l'Intervallo di capacità Aurora Serverless v2 di un cluster Aurora PostgreSQL.

Parametri che Aurora adegua in caso di riduzione verticale e dimensionamento verso l'alto di Aurora Serverless v2

Quando viene eseguita la scalabilità automatica, Aurora Serverless v2 deve essere in grado di modificare i parametri affinché ciascuna istanza database funzioni in modo ottimale in base all'incremento o al decremento della capacità. Pertanto, non è possibile sovrascrivere alcuni parametri relativi alla capacità. Per alcuni parametri che è possibile sovrascrivere, evita di codificare valori fissi. Le seguenti considerazioni si applicano a queste impostazioni relative alla capacità.

Per Aurora MySQL, Aurora Serverless v2 esegue il dimensionamento dinamico di alcuni parametri durante il dimensionamento. Per i seguenti parametri, Aurora Serverless v2 non utilizza i valori di parametro personalizzati specificati:

- `innodb_buffer_pool_size`
- `innodb_purge_threads`
- `table_definition_cache`
- `table_open_cache`

Per Aurora PostgreSQL, Aurora Serverless v2 esegue il dimensionamento dinamico del seguente parametro durante il dimensionamento. Per i seguenti parametri, Aurora Serverless v2 non utilizza i valori di parametro personalizzati specificati:

- `shared_buffers`

Per tutti i parametri diversi da quelli elencati qui, le istanze database Aurora Serverless v2 hanno lo stesso funzionamento delle istanze database con provisioning. Il valore di default del parametro viene ereditato dal gruppo di parametri del cluster. È possibile modificare il valore di default per l'intero cluster utilizzando un gruppo di parametri personalizzato per il cluster. In alternativa, è possibile modificare il valore di default per alcune istanze database utilizzando un gruppo di parametri personalizzato per il database. I parametri dinamici vengono aggiornati immediatamente. Le modifiche apportate ai parametri statici hanno effetto solo dopo il riavvio dell'istanza database.

Parametri calcolati da Aurora in base alla capacità massima di Aurora Serverless v2

Per i seguenti parametri, Aurora PostgreSQL utilizza i valori predefiniti derivati dalle dimensioni della memoria in base all'impostazione ACU massima, in modo analogo a quanto avviene con `max_connections`:

- `autovacuum_max_workers`
- `autovacuum_vacuum_cost_limit`
- `autovacuum_work_mem`
- `effective_cache_size`
- `maintenance_work_mem`

Evitare gli errori out-of-memory

Se una delle istanze database Aurora Serverless v2 raggiunge costantemente il limite della capacità massima, Aurora indica questa condizione impostando l'istanza datanase sullo stato `incompatible-parameters`. Se lo stato dell'istanza database è `incompatible-parameters`, alcune operazioni sono bloccate. Ad esempio, non è possibile aggiornare la versione del motore.

In genere, l'istanza DB assume questo stato quando si riavvia frequentemente a causa di out-of-memory errori. Aurora registra un evento quando si verifica questo tipo di riavvio. È possibile visualizzare l'evento seguendo la procedura descritta in Visualizzazione degli RDS eventi Amazon. Un utilizzo della memoria insolitamente elevato può verificarsi a causa del sovraccarico dovuto all'attivazione di impostazioni come Approfondimenti sulle prestazioni e l'autenticazione IAM. Può anche essere causato da un carico di lavoro rilevante sull'istanza database o dalla gestione dei metadati associati a un gran numero di oggetti dello schema.

Se il sovraccarico sulla memoria diminuisce e pertanto l'istanza database non raggiunge la sua capacità massima molto spesso, Aurora imposta automaticamente lo stato dell'istanza database su `available`.

Per gestire in modo ottimale questa situazione, è possibile eseguire alcune o tutte le seguenti azioni:

- Aumentare il limite inferiore della capacità per le istanze database Aurora Serverless v2 modificando il valore minimo dell'unità di capacità Aurora (ACU) per il cluster. In questo modo si evitano i problemi in cui un database inattivo viene ridotto a una capacità di memoria inferiore a quella necessaria per le funzionalità attivate nel cluster. Dopo aver modificato le impostazioni ACU

per il cluster, riavviare l'istanza database Aurora Serverless v2. In questo modo Aurora valuta se è in grado di ripristinare lo stato `available`.

- Aumentare il limite superiore della capacità per le istanze database Aurora Serverless v2 modificando il valore ACU massimo per il cluster. Ciò consente di evitare problemi nel caso in cui un database occupato non sia in grado di eseguire il dimensionamento verso l'alto fino a una capacità con memoria sufficiente per le funzionalità attivate nel cluster e nel carico di lavoro del database. Dopo aver modificato le impostazioni ACU per il cluster, riavviare l'istanza database Aurora Serverless v2. In questo modo Aurora valuta se è in grado di ripristinare lo stato `available`.
- Disattivare le impostazioni di configurazione che richiedono un sovraccarico di memoria. Ad esempio, supponiamo di avere funzionalità come AWS Identity and Access Management (IAM), Performance Insights o la replica binaria dei log MySQL di Aurora attivate ma non le utilizzi. In questo caso, è possibile disattivarle. In alternativa, è possibile adeguare i valori di capacità minima e massima per il cluster in modo da considerare la memoria utilizzata da tali funzionalità. Per le linee guida sulla scelta delle impostazioni di capacità minima e massima, consulta Scelta dell'intervallo di capacità di Aurora Serverless v2 per un cluster Aurora.
- Ridurre il carico di lavoro sull'istanza database. Ad esempio, è possibile aggiungere istanze database di lettura al cluster per distribuire il carico dalle query di sola lettura su più istanze database.
- Ottimizzare il codice SQL utilizzato dall'applicazione in modo da usare meno risorse. Ad esempio, è possibile esaminare i piani di query, controllare il registro delle query caratterizzate da un'esecuzione lenta o modificare gli indici nelle tabelle. È inoltre possibile eseguire altri tipi comuni di sintonizzazione SQL.

CloudWatch Parametri Amazon importanti per Aurora Serverless v2

Per iniziare a usare Amazon CloudWatch per la tua istanza Aurora Serverless v2 DB, consultaVisualizzazione Aurora Serverless v2 log in Amazon CloudWatch. Per ulteriori informazioni su come monitorare i cluster Aurora DB, consulta CloudWatch. Monitoraggio degli eventi di registro in Amazon CloudWatch

Puoi visualizzare le tue istanze Aurora Serverless v2 DB CloudWatch per monitorare la capacità consumata da ciascuna istanza DB con la metrica. `ServerlessDatabaseCapacity` Puoi anche monitorare tutte le CloudWatch metriche Aurora standard, come e. `DatabaseConnections` `Queries` Per l'elenco completo delle CloudWatch metriche che puoi monitorare per Aurora, consulta. CloudWatch Parametri Amazon per Amazon Aurora I parametri metriche sono suddivise in parametri

a livello di cluster e parametri a livello di istanza in Parametri a livello di cluster per Amazon Aurora e Parametri a livello di istanza per Amazon Aurora.

Le seguenti metriche a CloudWatch livello di istanza sono importanti da monitorare per consentirti di capire in che modo le tue istanze Aurora Serverless v2 DB si stanno scalando verso l'alto e verso il basso. Tutti questi parametri vengono calcolati ogni secondo. In questo modo, puoi monitorare lo stato corrente delle istanze database Aurora Serverless v2. È possibile impostare allarmi per avvisarti se i parametri relativi alla capacità di una delle istanze database Aurora Serverless v2 stanno raggiungendo la soglia limite. È possibile determinare se le impostazioni di capacità minima e massima sono appropriate o se è necessario adeguarle. È possibile individuare le aree da valutare con maggiore attenzione per ottimizzare l'efficienza del database.

- `ServerlessDatabaseCapacity`. Come metrica a livello di istanza, riporta il numero ACUs rappresentato dalla capacità corrente dell'istanza DB. Come parametro a livello di cluster, restituisce la media dei valori di `ServerlessDatabaseCapacity` di tutte le istanze database Aurora Serverless v2 nel cluster. Questo parametro è un parametro solo a livello di cluster in Aurora Serverless v1. In Aurora Serverless v2, è disponibile a livello di istanza database e a livello di cluster.
- `ACUUtilization`. Questo parametro è nuovo in Aurora Serverless v2. Questo valore è rappresentato come percentuale. Viene calcolato come il valore del parametro `ServerlessDatabaseCapacity` diviso per il valore ACU massimo del cluster di database. Considera le seguenti linee guida per l'interpretazione di questo parametro e per valutare quale azione eseguire:
  - Se questo parametro si avvicina al valore `100.0`, il dimensionamento verso l'alto dell'istanza database ha raggiunto il valore massimo possibile. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. In questo modo, sia le istanze database di scrittura che quelle di lettura possono essere dimensionate in base a una capacità superiore.
  - Supponiamo che un carico di lavoro di sola lettura faccia sì che il valore del parametro `ACUUtilization` di un'istanza database di lettura si avvicini `100.0`, mentre l'istanza database di scrittura non raggiunge la relativa capacità massima. In questo caso, considera di aggiungere istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente il carico su ogni istanza database di lettura.
  - Supponiamo di eseguire un'applicazione di produzione, in cui prestazioni e scalabilità sono fattori di primaria importanza. In questo caso, puoi impostare il valore ACU massimo per il cluster su un numero elevato. Il tuo obiettivo è avere il parametro `ACUUtilization` su un valore sempre

inferiore a `100.0`. Con un valore ACU massimo elevato, puoi essere certo di disporre di spazio sufficiente in caso di picchi imprevisti a livello di attività del database. Dovrai sostenere solo i costi relativi al consumo effettivo di capacità del database.

- `CPUUtilization`. Questo parametro viene interpretato in modo diverso in Aurora Serverless v2 rispetto a quanto avviene per le istanze database con provisioning. Per Aurora Serverless v2, questo valore è una percentuale calcolata come la quantità di CPU attualmente utilizzata divisa per la capacità della CPU disponibile sotto il valore ACU massimo del cluster di database. Aurora monitora automaticamente questo valore e dimensiona verso l'alto l'istanza database Aurora Serverless v2 quando questa utilizza costantemente una percentuale elevata della capacità della CPU.

  Se il valore di questo parametro si avvicina a `100.0`, l'istanza database ha raggiunto la capacità massima della CPU. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. Se il valore di questo parametro si avvicina a `100.0` per un'istanza database di lettura, valuta la possibilità di aggiungere altre istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente il carico su ogni istanza database di lettura.
- `FreeableMemory`. Questo valore rappresenta la quantità di memoria inutilizzata disponibile quando viene eseguito il dimensionamento dell'istanza database Aurora Serverless v2 fino alla sua capacità massima. Per ogni ACU la cui capacità corrente è inferiore alla capacità massima, questo valore aumenta di circa 2 GiB. Pertanto, questo parametro non si avvicina a zero finché non viene eseguito il dimensionamento verso l'alto dell'istanza database fino al valore più alto possibile.

  Se il valore di questo parametro si avvicina a `0`, viene eseguito il dimensionamento verso l'alto dell'istanza database al valore massimo possibile, ovvero al valore più prossimo al limite di memoria disponibile. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. Se il valore di questo parametro si avvicina a `0` per un'istanza database di lettura, valuta la possibilità di aggiungere altre istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente l'utilizzo della memoria su ogni istanza database di lettura.
- `TempStorageIops`. Il numero di IOPS eseguiti sull'archiviazione locale collegata all'istanza database. Include gli IOPS per le operazioni sia di letture che di scrittura. Questo parametro rappresenta un conteggio e viene misurato una volta al secondo. Questo è un nuovo parametro per Aurora Serverless v2. Per informazioni dettagliate, consultare Parametri a livello di istanza per Amazon Aurora.

- `TempStorageThroughput`. Quantità di dati trasferiti da e verso l'archiviazione locale associata all'istanza database. Questo parametro rappresenta i byte e viene misurato una volta al secondo. Questo è un nuovo parametro per Aurora Serverless v2. Per informazioni dettagliate, consultare Parametri a livello di istanza per Amazon Aurora.

In genere, la maggior parte del dimensionamento verso l'alto delle istanze database Aurora Serverless v2 è causato dall'utilizzo della memoria e dall'attività della CPU. I parametri `TempStorageIops` e `TempStorageThroughput` possono aiutarti a diagnosticare i rari casi in cui l'attività di rete relativa ai trasferimenti tra l'istanza database e i dispositivi di archiviazione locale è responsabile di aumenti imprevisti della capacità. Per monitorare altre attività di rete, puoi utilizzare i seguenti parametri:

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Puoi fare in modo che Aurora pubblichi alcuni o tutti i log del database su Amazon Logs. CloudWatch Per istruzioni, consulta quanto segue a seconda del motore di database in uso:

- the section called “Pubblicazione dei log di Aurora PostgreSQL in Logs CloudWatch ”
- the section called “Pubblicazione dei log di Aurora MySQL su Logs CloudWatch ”

In che modo Aurora Serverless v2 le metriche si applicano alla fattura AWS

Gli Aurora Serverless v2 addebiti sulla AWS fattura vengono calcolati in base alla stessa `ServerlessDatabaseCapacity` metrica che puoi monitorare. Il meccanismo di fatturazione può differire dalla CloudWatch media calcolata per questa metrica nei casi in cui si utilizza la Aurora Serverless v2 capacità solo per una parte dell'ora. Può anche differire se i problemi di sistema rendono la CloudWatch metrica non disponibile per brevi periodi. Pertanto, sulla fattura potresti vedere un valore di ore ACU leggermente diverso rispetto al calcolo fatto sulla base del valore medio del parametro `ServerlessDatabaseCapacity`.

Esempi di CloudWatch comandi per Aurora Serverless v2 le metriche

AWS CLI Gli esempi seguenti mostrano come monitorare le CloudWatch metriche più importanti relative a. Aurora Serverless v2 In ogni caso, sostituire la stringa `Value=` per il parametro `--dimensions` con l'identificativo dell'istanza database Aurora Serverless v2 in uso.

Nell'esempio Linux seguente vengono visualizzati i valori di capacità minima, massima e media per un'istanza database, misurati ogni 10 minuti ogni ora. Il comando Linux `date` specifica l'ora di inizio e l'ora di fine rispetto alla data e all'ora correnti. La funzione `sort_by` nel parametro `--query` ordina i risultati cronologicamente in base al campo `Timestamp`.

```
aws cloudwatch get-metric-statistics --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Gli esempi di Linux riportati di seguito illustrano il monitoraggio della capacità di ogni istanza database in un cluster. Misurano l'utilizzo minimo, massimo e medio della capacità di ciascuna istanza database. Le misurazioni vengono effettuate una volta ogni ora per un periodo di tre ore. Questi esempi utilizzano la `ACUUtilization` metrica che rappresenta una percentuale del limite superiore di ACUs, anziché `ServerlessDatabaseCapacity` rappresentare un numero fisso di. ACUs In questo modo, non è necessario conoscere i numeri effettivi per i valori ACU minimi e massimi nell'intervallo di capacità. Puoi visualizzare percentuali che vanno da 0 a 100.

```
aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_writer_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table

aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_reader_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Il seguente esempio Linux esegue misurazioni simili a quelle precedenti. In questo caso, le misure fanno riferimento al parametro `CPUUtilization`. Le misurazioni vengono effettuate ogni 10 minuti per un periodo di 1 ora. I numeri rappresentano la percentuale di CPU disponibile utilizzata, in base alle risorse della CPU disponibili per l'impostazione di capacità massima per l'istanza database.

```
aws cloudwatch get-metric-statistics --metric-name "CPUUtilization" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Il seguente esempio Linux esegue misurazioni simili a quelle precedenti. In questo caso, le misure fanno riferimento al parametro `FreeableMemory`. Le misurazioni vengono effettuate ogni 10 minuti per un periodo di 1 ora.

```
aws cloudwatch get-metric-statistics --metric-name "FreeableMemory" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Monitoraggio delle prestazioni di Aurora Serverless v2 con Approfondimenti sulle prestazioni

Puoi utilizzare Approfondimenti sulle prestazioni per monitorare le prestazioni di istanze database Aurora Serverless v2. Per le procedure di Approfondimenti sulle prestazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

I seguenti nuovi contatori di Approfondimenti sulle prestazioni si applicano alle istanze database Aurora Serverless v2:

- `os.general.serverlessDatabaseCapacity`— La capacità corrente dell'istanza DB in ACUs. Il valore corrisponde alla `ServerlessDatabaseCapacity` CloudWatch metrica per l'istanza DB.
- `os.general.acuUtilization`: percentuale di capacità attuale oltre la capacità massima configurata. Il valore corrisponde alla `ACUUtilization` CloudWatch metrica per l'istanza DB.
- `os.general.maxConfiguredAcu`: capacità massima configurata per l'istanza database Aurora Serverless v2 corrente. Viene misurato in ACUs.

- `os.general.minConfiguredAcu`: capacità minima configurata per l'istanza database Aurora Serverless v2 corrente. È misurata in ACUs

Per l'elenco completo dei contatori di Approfondimenti sulle prestazioni, consulta Parametri contatore di Performance Insights.

Quando in Approfondimenti sulle prestazioni vengono visualizzati i valori `vCPU` per un'istanza database Aurora Serverless v2, tali valori rappresentano stime basate sul valore ACU per l'istanza database. Per intervalli di default di un minuto, tutti i valori vCPU frazionari vengono arrotondati al numero intero più vicino. Per intervalli più lunghi, il valore vCPU visualizzato è la media dei valori vCPU interi per ogni minuto.

Risoluzione dei problemi di capacità di Aurora Serverless v2

In alcuni casi, Aurora Serverless v2 non viene ridotto fino alla capacità minima, anche in assenza di carico sul database. Questo può accadere per i seguenti motivi:

- Alcune funzionalità possono aumentare l'utilizzo delle risorse e impedire il dimensionamento del database fino alla capacità minima. Le caratteristiche principali comprendono:
  - Database globali di Aurora
  - Esportazione dei registri CloudWatch
  - Abilitazione di `pg_audit` nei cluster database compatibili con Aurora PostgreSQL
  - Monitoraggio avanzato
  - Approfondimenti sulle prestazioni

  Per ulteriori informazioni, consulta Scelta dell'impostazione Aurora Serverless v2 minima della capacità per un cluster.
- Se un'istanza di lettura non si riduce fino alla capacità minima e mantiene la capacità uguale o superiore dell'istanza di scrittura, controlla il livello di priorità dell'istanza di scrittura. Le istanze database di lettura Aurora Serverless v2 di livello 0 o 1 vengono mantenute a una capacità minima pari ad almeno quella dell'istanza database di scrittura. Imposta il livello di priorità dell'istanza di lettura almeno su 2 in modo che possa essere aumentata o ridotta indipendentemente dall'istanza di scrittura. Per ulteriori informazioni, consulta Scelta del livello di promozione per un Aurora Serverless v2 reader.
- Imposta i valori predefiniti dei parametri del database che influiscono sulla dimensione della memoria condivisa. L'impostazione di un valore superiore a quello predefinito aumenta la richiesta

di memoria condivisa e impedisce la riduzione del database fino alla capacità minima. Alcuni esempi sono `max_connections` e `max_locks_per_transaction`.

> **Note**
>
> L'aggiornamento dei parametri della memoria condivisa richiede il riavvio del database per rendere effettive le modifiche.

- I carichi di lavoro di database gravosi possono aumentare l'utilizzo delle risorse.
- I volumi di database di grandi dimensioni possono aumentare l'utilizzo delle risorse.

  Amazon Aurora utilizza risorse di memoria e CPU per la gestione dei cluster database. Aurora richiede più CPU e memoria per gestire cluster database con volumi di database più grandi. Se la capacità minima del cluster è inferiore a quella minima richiesta per la gestione dei cluster, il cluster non verrà ridotto fino alla capacità minima.
- Anche i processi in background, come l'eliminazione, possono aumentare l'utilizzo delle risorse.

Se il database non riesce ancora a ridurre fino la capacità minima configurata, interrompi e riavvia il database per recuperare eventuali frammenti di memoria che potrebbero essersi accumulati nel tempo. L'arresto e l'avvio di un database comportano tempi di inattività, quindi consigliamo di farlo raramente.

Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2

Puoi specificarlo Aurora Serverless v2 Le istanze DB si ridimensionano fino a zero ACUs e si mettono automaticamente in pausa, se non hanno connessioni avviate dall'attività dell'utente entro un periodo di tempo specificato. A tale scopo, è necessario specificare un ACU valore minimo pari a zero per il cluster DB. Non ti viene addebitato alcun costo per la capacità dell'istanza mentre un'istanza è in pausa. L'attivazione della funzionalità di pausa e ripresa automatica (pausa automatica) per i cluster Aurora poco utilizzati o con lunghi periodi di inattività può aiutarti a gestire i costi per il tuo parco di database.

> **Note**
>
> La funzione di pausa automatica è disponibile per Aurora Serverless v2 sia con Aurora SQL Postgre che Aurora My. SQL Potrebbe essere necessario aggiornare la versione del motore di database Aurora per sfruttare questa funzionalità. Per le versioni del motore in cui ACUs è disponibile una capacità minima di 0, vedereAurora Serverless v2 capacità.

Argomenti

- Panoramica del Aurora Serverless v2 funzione di pausa automatica
- Prerequisiti e limitazioni per Aurora Serverless v2 funzione di pausa automatica
- Attivazione e disattivazione della funzione di pausa automatica
- In che modo Aurora Serverless v2 la funzione di pausa automatica funziona
- In che modo Aurora Serverless v2 la pausa automatica funziona per diversi tipi di cluster Aurora
- Monitoraggio dei cluster Aurora che utilizzano la pausa automatica
- Risoluzione dei problemi per Aurora Serverless v2 pausa automatica
- Considerazioni sulla progettazione delle applicazioni per Aurora Serverless v2 funzione di pausa automatica

Panoramica del Aurora Serverless v2 funzione di pausa automatica

Aurora Serverless v2 Le istanze DB possono interrompersi automaticamente dopo un periodo senza connessioni utente e riprendere automaticamente quando arriva una richiesta di connessione. Il Aurora Serverless v2 la funzione di pausa/ripresa automatica aiuta a gestire i costi per i sistemi

che non hanno un obiettivo di livello di servizio rigoroso (). SLO Ad esempio, è possibile abilitare questa funzionalità per i cluster utilizzati per lo sviluppo e il test o per le applicazioni interne in cui è accettabile una breve pausa durante la ripresa del database. Se il tuo carico di lavoro ha periodi di inattività e può tollerare lievi ritardi nella connessione mentre l'istanza riprende, prendi in considerazione l'utilizzo della pausa automatica con Aurora Serverless v2 istanze per ridurre i costi.

È possibile controllare questo comportamento specificando se Aurora Serverless v2 Le istanze DB in un cluster possono essere messe in pausa automaticamente o meno e per quanto tempo ciascuna istanza deve rimanere inattiva prima di mettersi in pausa. Per abilitare il comportamento di pausa automatica per tutti i Aurora Serverless v2 Per le istanze DB in un cluster Aurora, si imposta il valore di capacità minima per il cluster su zero. ACUs

Se in precedenza hai usufruito del Aurora Serverless v1 funzionalità che è stata ridimensionata a zero ACUs dopo un periodo di inattività, è possibile eseguire l'aggiornamento a Aurora Serverless v2 e usa la funzione di pausa automatica corrispondente.

I vantaggi in termini di risparmio dei costi della funzione di pausa automatica sono simili all'utilizzo della funzionalità stop/start del cluster. Pausa automatica per Aurora Serverless v2 offre i vantaggi aggiuntivi di una ripresa più rapida rispetto all'avvio di un cluster interrotto e di automatizzare il processo di determinazione del momento in cui mettere in pausa e riprendere ciascuna istanza DB.

La funzione di pausa automatica fornisce inoltre una maggiore granularità nel controllo dei costi per le risorse di elaborazione all'interno del cluster. È possibile attivare la pausa di alcune istanze di lettura anche se l'istanza Writer e gli altri lettori del cluster rimangono attivi in qualsiasi momento. A tale scopo, assegnate alle istanze di lettura che possono essere messe in pausa indipendentemente dalle altre istanze una priorità di failover compresa tra 2 e 15.

Le istanze Writer e tutte le istanze Reader con priorità di failover 0 e 1 vengono sempre messe in pausa e riprendono contemporaneamente. Pertanto, le istanze di questo gruppo vengono messe in pausa dopo che nessuna di esse ha avuto alcuna connessione per l'intervallo di tempo specificato.

I cluster Aurora DB possono contenere una combinazione di istanze DB writer e reader e sono predisposti e Aurora Serverless v2 Istanze DB. Pertanto, per utilizzare questa funzionalità in modo efficace, è utile comprendere i seguenti aspetti del meccanismo di pausa automatica:

- Le circostanze in cui un'istanza DB potrebbe interrompersi automaticamente.
- Quando è possibile impedire la pausa di un'istanza DB. Ad esempio, l'abilitazione di alcune funzionalità di Aurora o l'esecuzione di determinati tipi di operazioni sul cluster potrebbe impedire la sospensione delle istanze, anche senza alcuna connessione a tali istanze.

- Le conseguenze per il monitoraggio e la fatturazione mentre un'istanza è in pausa.
- Quali azioni fanno sì che un'istanza DB riprenda l'elaborazione.
- Come cambia la capacità di un'istanza DB nel momento in cui si verificano gli eventi di pausa e ripresa.
- Come controllare l'intervallo di inattività prima della pausa di un'istanza DB.
- Come codificare la logica dell'applicazione per gestire il periodo mentre un'istanza DB riprende l'elaborazione.

Prerequisiti e limitazioni per Aurora Serverless v2 funzione di pausa automatica

Prima di utilizzare la funzione di pausa automatica, controlla quali versioni del motore utilizzare. Inoltre, verifica se la pausa automatica funziona in combinazione con le altre funzionalità di Aurora che intendi utilizzare. Non puoi attivare la pausa automatica se utilizzi una versione del motore che non la supporta. Per le funzionalità incompatibili, non riceverai alcun errore se le utilizzi in combinazione con la pausa automatica. Se il cluster utilizza funzionalità o impostazioni incompatibili, Aurora Serverless v2 le istanze non verranno messe in pausa automaticamente.

- Se utilizzi Aurora PostgreSQL, il motore di database deve eseguire almeno la versione 16.3, 15.7, 14.12 o 13.15.
- Se utilizzi Aurora MySQL, il motore di database deve eseguire la versione 3.08.0 o successiva.
- Per l'elenco completo delle versioni del motore e delle AWS regioni in cui è disponibile questa funzionalità, consulta. Regioni supportate e motori Aurora DB per Aurora Serverless v2
- Quando un Aurora Serverless v2 l'istanza viene ripresa, la sua capacità potrebbe essere inferiore rispetto a quando l'istanza è stata messa in pausa. Per informazioni dettagliate, consultare Differenze nel comportamento della pausa automatica tra Aurora Serverless v2 e Aurora Serverless v1.

Alcune condizioni o impostazioni impediscono Aurora Serverless v2 le istanze si interrompano automaticamente. Per ulteriori informazioni, consulta Situazioni in cui Aurora Serverless v2 non si mette in pausa automaticamente.

Attivazione e disattivazione della funzione di pausa automatica

Puoi attivare e disattivare la funzione di pausa automatica a livello di cluster. A tale scopo, si utilizzano le stesse procedure utilizzate per regolare la capacità minima e massima del cluster. La funzione di pausa automatica è rappresentata da una capacità minima di 0. ACUs

Argomenti

- Attivazione della pausa automatica per Aurora Serverless v2 istanze in un cluster
- Configurabile Aurora Serverless v2 intervallo di timeout di pausa automatica
- Ripresa di una pausa automatica Aurora Serverless v2 istanza
- Disattivazione della pausa automatica per Aurora Serverless v2 istanze in un cluster

Attivazione della pausa automatica per Aurora Serverless v2 istanze in un cluster

Segui la procedura riportata in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster. Per la capacità minima, scegli 0. ACUs Quando scegli una capacità minima pari a 0ACUs, puoi anche specificare il periodo di inattività dell'istanza prima che venga messa automaticamente in pausa.

L'CLIesempio seguente mostra come creare un cluster Aurora con la funzionalità di pausa automatica abilitata e l'intervallo di pausa automatica impostato su dieci minuti (600 secondi).

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0 \
    --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=4,SecondsUntilAutoPause=600 \
    --master-username myuser \
    --manage-master-user-password
```

L'CLIesempio seguente mostra come attivare la funzionalità di pausa automatica per un cluster Aurora esistente. Questo esempio imposta l'intervallo di pausa automatica su un'ora (3600 secondi).

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=80,SecondsUntilAutoPause=3600
```

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 0,
    "MaxCapacity": 80.0,
    "SecondsUntilAutoPause": 3600
}
```

Configurabile Aurora Serverless v2 intervallo di timeout di pausa automatica

L'intervallo di timeout è rappresentato nell'`ServerlessV2ScalingConfiguration`attributo del cluster Aurora. È possibile specificare questo intervallo AWS Management Console durante la creazione o la modifica di un cluster Aurora, se la capacità minima è impostata su zero. ACUs È possibile specificarlo AWS CLI utilizzando il `--serverless-v2-scaling-configuration` parametro durante la creazione o la modifica di un cluster Aurora. È possibile specificarlo RDS API utilizzando il `ServerlessV2ScalingConfiguration` parametro durante la creazione o la modifica di un cluster Aurora.

L'intervallo minimo che è possibile impostare è di 300 secondi (cinque minuti). Questa è l'impostazione predefinita se non si specifica un intervallo. L'intervallo massimo che puoi impostare è 86.400 secondi (un giorno).

Si supponga di disattivare la funzionalità di pausa automatica per un cluster modificando la capacità minima del cluster su un valore diverso da zero. In tal caso, la proprietà interval viene rimossa dall'attributo. `ServerlessV2ScalingConfiguration` L'assenza di tale proprietà fornisce un'ulteriore conferma che la funzione di pausa automatica è disattivata per quel cluster. Se successivamente riattivi la pausa automatica, puoi specificare nuovamente qualsiasi intervallo personalizzato in quel momento.

Ripresa di una pausa automatica Aurora Serverless v2 istanza

- Quando ti connetti a un'unità in pausa Aurora Serverless v2 istanza, riprende e accetta automaticamente la connessione.
- Un tentativo di connessione che non include credenziali valide causa comunque la ripresa dell'istanza DB.
- Se ti connetti tramite l'endpoint writer, Aurora riprende l'istanza Writer DB se viene messa in pausa automaticamente. Allo stesso tempo, Aurora riattiva tutte le istanze di lettura in pausa automatica con priorità di failover 0 o 1, il che significa che la loro capacità è legata alla capacità dell'istanza di scrittura.

- Se ti connetti tramite l'endpoint del lettore, Aurora sceglie un'istanza del lettore in modo casuale. Se l'istanza del lettore è in pausa, Aurora la riprende. Aurora riprende inoltre per prima l'istanza writer, poiché l'istanza writer deve essere sempre attiva se sono attive delle istanze Reader. Quando Aurora riprende l'istanza Writer, vengono ripristinate anche tutte le istanze Reader nei livelli di promozione di failover zero e uno.
- Se invii una richiesta al tuo cluster tramite RDS DataAPI, Aurora riprende l'istanza writer se è in pausa. Quindi Aurora elabora la richiesta di datiAPI.
- Quando si modifica il valore di un parametro di configurazione in un gruppo di parametri del cluster DB, Aurora riprende automaticamente qualsiasi parametro in pausa Aurora Serverless v2 istanze in tutti i cluster che utilizzano quel gruppo di parametri del cluster. Allo stesso modo, quando modificate il valore di un parametro in un gruppo di parametri DB, Aurora riprende automaticamente qualsiasi valore in pausa Aurora Serverless v2 istanze che utilizzano quel gruppo di parametri DB. Lo stesso comportamento di ripristino automatico si applica quando si modifica un cluster per assegnare un gruppo di parametri del cluster diverso o quando si modifica un'istanza per assegnare un gruppo di parametri DB diverso.
- L'esecuzione di una richiesta di backtrack riprende automaticamente il Aurora Serverless v2 istanza writer se è in pausa. Aurora elabora la richiesta di backtrack dopo la ripresa dell'istanza writer. È possibile tornare indietro a un periodo in cui un Aurora Serverless v2 l'istanza è stata messa in pausa.
- L'acquisizione di un'istantanea del cluster o l'eliminazione di un'istantanea non ne causa alcuna Aurora Serverless v2 istanze da riprendere.
- La creazione di un clone di Aurora fa sì che Aurora riprenda l'istanza writer del cluster che viene clonato.
- Se un'istanza in pausa riceve un gran numero di richieste di connessione prima del termine della ripresa, alcune sessioni potrebbero non riuscire a connettersi. Si consiglia di implementare la logica di ripetizione per le connessioni ai cluster Aurora che ne hanno Aurora Serverless v2 istanze con pausa automatica abilitata. Ad esempio, è possibile riprovare una connessione fallita tre volte.
- Aurora può eseguire alcuni tipi di manutenzione interna minore senza riattivare un'istanza. Tuttavia, alcuni tipi di manutenzione che avvengono durante la finestra di manutenzione del cluster richiedono che Aurora riprenda l'istanza. Al termine della manutenzione, l'istanza viene nuovamente messa in pausa automaticamente se non c'è più attività dopo l'intervallo specificato.

> **Note**
>
> Aurora non riprende automaticamente un'istanza in pausa per i lavori pianificati specifici del motore, come quelli dell'estensione Postgre o SQL `pg_cron` dello scheduler My event. SQL Per garantire l'esecuzione di tali lavori, avvia manualmente una connessione all'istanza prima dell'orario pianificato. Aurora non mette in coda alcun lavoro in cui si verifica l'ora pianificata mentre l'istanza DB è in pausa. Tali processi vengono saltati quando l'istanza viene riavviata in un secondo momento.

- Se il cluster Aurora viene sottoposto a un failover durante un Aurora Serverless v2 l'istanza viene messa in pausa automaticamente, Aurora potrebbe riprendere un'istanza e quindi promuovere quell'istanza come scrittrice. Lo stesso potrebbe accadere se una o più istanze DB vengono rimosse dal cluster mentre un'istanza è in pausa. In questo caso, l'istanza diventa l'autore immediatamente quando viene ripresa.
- Le operazioni che modificano le proprietà del cluster causano anche eventuali pause automatiche Aurora Serverless v2 istanze da riprendere. Ad esempio, un'istanza in pausa automatica riprende per operazioni come le seguenti:
  - Modifica dell'intervallo di scalabilità del cluster.
  - Aggiornamento della versione del motore del cluster.
  - Descrizione o download di file di registro da un'istanza in pausa. È possibile esaminare i dati di registro storici delle istanze in pausa abilitando i caricamenti dei log CloudWatch e analizzando i log. CloudWatch

Disattivazione della pausa automatica per Aurora Serverless v2 istanze in un cluster

Segui la procedura riportata in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster. Per la capacità minima, scegli un valore pari o superiore a 0,5. Quando si disattiva la funzione di pausa automatica, l'intervallo di inattività dell'istanza viene reimpostato. Se riattivi la pausa automatica, specifichi un nuovo intervallo di timeout.

L'CLIesempio seguente mostra come disattivare la funzionalità di pausa automatica per un cluster Aurora esistente. L'`describe-db-clusters`output mostra che l'`SecondsUntilAutoPause`attributo viene rimosso quando la capacità minima è impostata su un valore diverso da zero.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
```

```
  --serverless-v2-scaling-configuration MinCapacity=2,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 2,
    "MaxCapacity": 80.0
}
```

In che modo Aurora Serverless v2 la funzione di pausa automatica funziona

Puoi utilizzare le seguenti informazioni per pianificare l'utilizzo della funzione di pausa automatica. Comprendere le circostanze in cui le istanze si interrompono, riprendono o rimangono attive può aiutarti a bilanciare i compromessi tra disponibilità, reattività e risparmio sui costi.

Argomenti

- Cosa succede quando Aurora Serverless v2 le istanze si interrompono
- Cosa succede quando viene messa in pausa automatica Aurora Serverless v2 le istanze riprendono
- Come funziona la fatturazione delle istanze per le istanze in pausa automatica Aurora Serverless v2 cluster
- Situazioni in cui Aurora Serverless v2 non si mette in pausa automaticamente
- Come funziona la pausa automatica con la funzione di stop/start del cluster
- Come funzionano la manutenzione e gli aggiornamenti per la pausa automatica Aurora Serverless v2 cluster
- Differenze nel comportamento della pausa automatica tra Aurora Serverless v2 e Aurora Serverless v1

Cosa succede quando Aurora Serverless v2 le istanze si interrompono

Quando un Aurora Serverless v2 L'istanza DB si interrompe dopo un periodo senza connessioni:

- Aurora inizia a mettere in pausa l'istanza dopo lo scadere dell'intervallo specificato senza alcuna connessione all'istanza, indipendentemente dal numero ACUs di istanze presenti in quel momento.
- Il meccanismo di pausa non è istantaneo. Un record Aurora Serverless v2 un'istanza che sta per essere messa in pausa automaticamente potrebbe attendere brevemente per recuperare il ritardo con tutte le modifiche allo storage Aurora.

- I costi di istanza relativi a quell'istanza vengono sospesi. La `ServerlessV2Usage` metrica ha un valore pari a 0 mentre l'istanza è in pausa.
- Il valore di stato dell'istanza non cambia. Lo stato viene comunque visualizzato come «disponibile».
- L'istanza interrompe la scrittura nei file di registro del database. Smette di inviare metriche a CloudWatch, ad eccezione della registrazione dello zero percento per `CPUUtilization` e `ACUUtilization` dello zero per. `ServerlessDatabaseCapacity`
- Aurora emette eventi quando un Aurora Serverless v2 L'istanza DB inizia la pausa, termina la pausa e se il meccanismo di pausa viene interrotto o non funziona correttamente. Per informazioni dettagliate su questi eventi, consulta. Eventi di istanza database

Cosa succede quando viene messa in pausa automatica Aurora Serverless v2 le istanze riprendono

Quando un Aurora Serverless v2 L'istanza DB riprende dopo essere stata messa in pausa automaticamente, si applicano le seguenti condizioni:

- Tutte le modifiche ai parametri incluse nelle `pending-reboot` modifiche vengono applicate alla ripresa dell'istanza.
- Aurora emette eventi a livello di istanza ogni volta che Aurora Serverless v2 L'istanza DB inizia a riprendere, termina la ripresa e se l'istanza non riesce a riprendere per qualche motivo. Per informazioni dettagliate su questi eventi, consulta. Eventi di istanza database
- Tutte le connessioni richieste vengono stabilite al termine della ripresa dell'istanza DB. Poiché il tempo di ripristino tipico potrebbe essere di circa 15 secondi, consigliamo di regolare le impostazioni di timeout del client in modo che superino i 15 secondi. Ad esempio, nelle impostazioni del JDBC driver puoi regolare i valori per le `sslResponseTimeout` impostazioni `connectTimeout` e in modo che siano più lunghi di 15 secondi.

> **Note**
>
> Se un Aurora Serverless v2 l'istanza rimane in pausa per più di 24 ore, Aurora può mettere l'istanza in uno stato di sonno più profondo che richiede più tempo per riprendere. In tal caso, il tempo di ripristino può essere di 30 secondi o più, all'incirca equivalente al riavvio dell'istanza. Se il database presenta periodi di inattività che durano più di un giorno, consigliamo di impostare i timeout di connessione su almeno 30 secondi.

Come funziona la fatturazione delle istanze per le istanze in pausa automatica Aurora Serverless v2 cluster

Mentre un Aurora Serverless v2 l'istanza viene messa in pausa automaticamente, il costo dell'istanza è zero. La `ServerlessV2Usage` metrica è zero durante quel periodo. AWS addebita ancora lo storage Aurora e altri aspetti del cluster che non sono legati a quella specifica istanza DB.

Situazioni in cui Aurora Serverless v2 non si mette in pausa automaticamente

- Se il valore di capacità minima per il cluster DB è superiore a zeroACUs, Aurora Serverless v2 le istanze nel cluster non vengono messe in pausa automaticamente. Se disponi di cluster esistenti con Aurora Serverless v2 nelle istanze precedenti alla disponibilità della funzione di pausa automatica, l'impostazione della capacità minima minima era 0,5. ACUs Per utilizzare la funzione di pausa automatica con tali cluster, modifica l'impostazione della capacità minima su zero. ACUs
- Se alcune connessioni avviate dall'utente sono aperte a Aurora Serverless v2 istanza, l'istanza non verrà messa in pausa. Inoltre, l'istanza non si interromperà mentre sono in corso attività come l'applicazione di patch e gli aggiornamenti. Le connessioni amministrative utilizzate da Aurora per i controlli di integrità non vengono conteggiate come attività e non impediscono la sospensione dell'istanza.
- In un cluster Aurora Postgre con replica logica abilitata o in un SQL cluster Aurora My SQL con replica binlog abilitata, l'istanza writer e tutte le istanze di lettura nei livelli di promozione del failover zero e uno non vengono messe in pausa automaticamente. Aurora esegue una quantità minima costante di attività per verificare lo stato della connessione di replica.

  Per i cluster con replica abilitata, puoi comunque avere istanze di lettore Aurora nel cluster di origine che si interrompono automaticamente. A tale scopo, imposta la loro priorità di failover su un valore diverso da zero o uno. In questo modo, possono essere messi in pausa indipendentemente dall'istanza di writer.
- Se al cluster Aurora è associato un RDS proxy, il proxy mantiene una connessione aperta a ogni istanza DB del cluster. Quindi, qualsiasi Aurora Serverless v2 le istanze in tale cluster non si interromperanno automaticamente.
- Se il cluster è il cluster principale di un database globale di Aurora, Aurora non sospende automaticamente il Aurora Serverless v2 istanza writer. Questo perché è necessario un livello costante di attività sull'istanza writer per gestire gli altri cluster nel database globale. Poiché l'istanza writer rimane attiva, qualsiasi Aurora Serverless v2 inoltre, le istanze di lettura con priorità di failover zero o uno non vengono messe in pausa automaticamente.

- Aurora Serverless v2 le istanze nei cluster secondari di un Aurora Global Database non vengono messe in pausa automaticamente. Se un cluster DB viene promosso a cluster autonomo, la funzionalità di pausa automatica diventa effettiva se vengono soddisfatte tutte le altre condizioni.
- In un cluster a cui è associata un'ETLintegrazione zero con Redshift, l'istanza writer e tutte le istanze Reader nei livelli di promozione del failover zero e uno non vengono messe in pausa automaticamente.
- Oltre all'attività sulla porta del database principale dell'istanza, se un'istanza di Aurora Postgre ha la funzionalità Babelfish abilitata, qualsiasi connessione e attività sulla SQL porta T impedisce che l'SQListanza venga messa in pausa automatica.

Come funziona la pausa automatica con la funzione di stop/start del cluster

Puoi arrestare e avviare un cluster Aurora quando la funzione di pausa automatica è abilitata. Non importa se alcune istanze vengono messe in pausa. Quando riavvii il cluster, qualsiasi elemento viene messo in pausa Aurora Serverless v2 le istanze vengono ripristinate automaticamente.

Mentre un cluster Aurora è fermo, qualsiasi cluster viene messo in pausa Aurora Serverless v2 le istanze non vengono riattivate automaticamente in base ai tentativi di connessione. Una volta riavviato il cluster, i soliti meccanismi di pausa e ripresa Aurora Serverless v2 si applicano le istanze.

Come funzionano la manutenzione e gli aggiornamenti per la pausa automatica Aurora Serverless v2 cluster

- Mentre un Aurora Serverless v2 l'istanza viene messa in pausa automaticamente, se si tenta di aggiornare il cluster Aurora, Aurora riprende l'istanza e la aggiorna.
- Aurora riprende periodicamente qualsiasi pausa automatica Aurora Serverless v2 istanze per eseguire la manutenzione, ad esempio aggiornamenti di versioni minori e modifiche alle proprietà, ad esempio i gruppi di parametri.
- Dopo un Aurora Serverless v2 l'istanza si riattiva per un'operazione amministrativa come un aggiornamento o l'applicazione della manutenzione, Aurora attende almeno 20 minuti prima di sospendere nuovamente l'istanza. Questo per consentire il completamento di qualsiasi operazione in background. Il periodo di venti minuti evita inoltre di sospendere e riprendere l'istanza più volte se l'istanza viene sottoposta a più operazioni amministrative in successione.

Differenze nel comportamento della pausa automatica tra Aurora Serverless v2 e Aurora Serverless v1

- Il tempo di ripresa è migliorato in Aurora Serverless v2 rispetto a Aurora Serverless v1. Il tempo di ripresa è in genere di circa 15 secondi se l'istanza è stata messa in pausa per meno di 24 ore. Se l'istanza viene messa in pausa per più di 24 ore, il tempo di ripresa potrebbe essere più lungo.
- Il modo in cui Aurora Serverless v2 se si applica ai cluster Multi-AZ, significa che alcune istanze DB nel cluster potrebbero essere messe in pausa mentre altre sono attive. L'istanza writer riprende ogni volta che un lettore è in esecuzione, perché lo writer è necessario per coordinare determinate attività all'interno del cluster. Perché Aurora Serverless v1 non utilizza istanze di lettura, l'intero cluster sarebbe sempre in pausa o attivo.
- Quando l'endpoint reader sceglie casualmente un'istanza del lettore a cui connettersi, quell'istanza del lettore potrebbe essere già attiva o potrebbe essere messa in pausa automaticamente. Pertanto, il tempo necessario per accedere all'istanza del lettore potrebbe variare ed essere più difficile da prevedere. Cluster Multi-AZ che utilizzano Aurora Serverless v2 e la pausa automatica potrebbero quindi trarre vantaggio dalla configurazione di endpoint personalizzati per casi d'uso specifici di sola lettura, invece di indirizzare tutte le sessioni di sola lettura all'endpoint del lettore.
- Aurora Serverless v2 le istanze vengono sottoposte a operazioni di manutenzione con la stessa frequenza delle istanze fornite. Poiché Aurora riprende automaticamente le istanze in cui è necessaria tale manutenzione, potresti scoprire che Aurora Serverless v2 le istanze vengono riattivate più frequentemente di Aurora Serverless v1 i cluster lo hanno fatto.

In che modo Aurora Serverless v2 la pausa automatica funziona per diversi tipi di cluster Aurora

Le considerazioni relative alla funzionalità di pausa automatica dipendono dal numero di istanze presenti nel cluster Aurora, dai livelli di promozione del failover delle istanze Reader e dal fatto che tutte le istanze siano Aurora Serverless v2 o una combinazione di Aurora Serverless v2 e fornito.

Argomenti

- Layout di cluster Aurora consigliati quando si utilizza la pausa automatica
- In che modo Aurora Serverless v2 la pausa automatica funziona per l'istanza writer in un cluster DB
- In che modo Aurora Serverless v2 la pausa automatica funziona per i cluster Multi-AZ

- In che modo Aurora Serverless v2 la pausa automatica funziona per i cluster con istanze assegnate

Layout di cluster Aurora consigliati quando si utilizza la pausa automatica

Quando la funzione di pausa automatica è abilitata, puoi organizzare il cluster Aurora per ottenere il giusto equilibrio tra alta disponibilità, risposta rapida e scalabilità in base al tuo caso d'uso. Puoi farlo scegliendo la combinazione di Aurora Serverless v2 istanze, istanze assegnate e livelli di promozione del failover per le istanze DB del cluster.

I seguenti tipi di configurazioni dimostrano diversi compromessi tra alta disponibilità e ottimizzazione dei costi per il cluster:

- Per un sistema di sviluppo e test, è possibile configurare un cluster DB Single-AZ con un Aurora Serverless v2 Istanza DB. La singola istanza serve tutte le richieste di lettura e scrittura. Quando il cluster non viene utilizzato per intervalli di tempo significativi, l'istanza DB viene messa in pausa. A quel punto, anche i costi di elaborazione DB per il cluster vengono sospesi.
- Per un sistema che esegue un'applicazione in cui l'alta disponibilità è una priorità, ma il cluster presenta ancora periodi in cui è completamente inattivo, puoi configurare un cluster Multi-AZ in cui si trovano sia le istanze DB di scrittura che quelle di lettura Aurora Serverless v2. Imposta l'istanza del lettore sulla priorità di failover zero o uno, in modo che l'istanza writer e l'istanza reader vengano messe in pausa e riprendano contemporaneamente. Ora ottieni il vantaggio di un failover rapido mentre il cluster è attivo. Quando il cluster rimane inattivo per un periodo superiore alla soglia di pausa automatica, i costi dell'istanza DB per entrambe le istanze vengono sospesi. Quando il cluster riprende l'elaborazione, la prima sessione del database impiega un breve periodo di tempo per connettersi.
- Supponiamo che il cluster sia costantemente attivo con una quantità minima di attività e richieda una risposta rapida per qualsiasi connessione. In tal caso, puoi creare un cluster con più di uno Aurora Serverless v2 istanza reader e disaccoppia le capacità di alcune istanze Reader da quelle di chi scrive. Specificate la priorità di failover zero o una per l'istanza writer e un'istanza reader. Specificate una priorità maggiore di una per le altre istanze del lettore. In questo modo, le istanze di lettura nei livelli di priorità più alta possono essere messe in pausa automaticamente, anche mentre lo scrittore e uno dei lettori rimangono attivi.

  In questo caso, puoi utilizzare altre tecniche per garantire che il cluster rimanga sempre disponibile pur continuando a ridurlo a una capacità ridotta durante i periodi di inattività:

- È possibile utilizzare le istanze assegnate allo scrittore e al lettore con priorità 0 o 1. In questo modo, due istanze DB non vengono mai messe in pausa automatica e sono sempre disponibili per servire il traffico del database ed eseguire failover.
- È possibile configurare un endpoint personalizzato che includa Aurora Serverless v2 istanze nei livelli di priorità più elevati, ma non lo scrittore o i lettori di livello di promozione 0 o 1. In questo modo, puoi indirizzare sessioni di sola lettura che non sono sensibili alla latenza ai lettori che potrebbero essere messe in pausa automaticamente. È possibile evitare di utilizzare l'endpoint reader per tali richieste, poiché Aurora potrebbe indirizzare le connessioni dell'endpoint reader all'istanza del lettore always-awake o a una delle istanze in pausa automatica. L'utilizzo dell'endpoint personalizzato consente di indirizzare le connessioni a diversi gruppi di istanze in base alle proprie preferenze in termini di risposta rapida o capacità di scalabilità aggiuntiva.

In che modo Aurora Serverless v2 la pausa automatica funziona per l'istanza writer in un cluster DB

Quando un cluster Aurora DB contiene solo una singola istanza DB, il meccanismo per mettere in pausa e riprendere automaticamente l'istanza DB è semplice. Dipende solo dall'attività sull'istanza writer. Potresti avere una configurazione di questo tipo per i cluster utilizzati per lo sviluppo e il test o per l'esecuzione di applicazioni in cui l'elevata disponibilità non è fondamentale. Nota che in un cluster a istanza singola, Aurora indirizza le connessioni attraverso l'endpoint reader all'istanza Writer DB. Pertanto, per un cluster DB a istanza singola, il tentativo di connettersi all'endpoint del lettore causa la ripresa dell'istanza DB writer messa in pausa automaticamente.

I seguenti fattori aggiuntivi si applicano ai cluster Aurora con più istanze DB:

- All'interno di un cluster Aurora DB, in genere si accede frequentemente all'istanza Writer DB. Pertanto, è possibile che l'istanza Writer DB rimanga attiva anche mentre una o più istanze DB Reader si interrompe automaticamente.
- Alcune attività sulle istanze Reader DB richiedono che l'istanza DB Writer sia disponibile. Pertanto, le istanze DB di Writer non possono essere messe in pausa finché non vengono messe in pausa anche tutte le istanze di Reader. La ripresa di qualsiasi istanza di reader riattiva automaticamente l'istanza di writer, anche se l'applicazione non accede direttamente all'istanza writer.
- Aurora Serverless v2 le istanze reader in fase di promozione con failover hanno un livello zero e uno scalabile per mantenere la loro capacità sincronizzata con l'istanza writer. Quindi, quando un Aurora Serverless v2 l'istanza dello scrittore riprende, così come qualsiasi altra Aurora Serverless v2 istanze reader che rientrano nei livelli di promozione zero o uno.

In che modo Aurora Serverless v2 la pausa automatica funziona per i cluster Multi-AZ

All'interno di un cluster Aurora DB contenente sia un writer che una o più istanze DB reader, alcune Aurora Serverless v2 Le istanze DB potrebbero essere messe in pausa mentre altre istanze DB sono attive. L'istanza writer e tutte le istanze di lettura con priorità di failover 0 e 1 vengono sempre messe in pausa e riprendono tutte contemporaneamente. Le istanze Reader con priorità diversa da 0 o 1 possono essere messe in pausa e riprendere indipendentemente dalle altre istanze.

Quando utilizzi questa funzionalità per cluster con più istanze Reader, puoi gestire i costi senza sacrificare l'elevata disponibilità. L'istanza writer e un'altra istanza di lettura di una o due istanze possono rimanere sempre attive, mentre le istanze di lettura aggiuntive possono essere messe in pausa quando non sono necessarie per gestire volumi di traffico di lettura elevati.

La possibilità di mettere in pausa automaticamente un'istanza reader dipende dal fatto che la sua capacità sia scalabile indipendentemente o dal fatto che la capacità sia legata a quella dell'istanza Writer DB. Questa proprietà di scalabilità dipende dalla priorità di failover dell'istanza DB del lettore. Quando la priorità del lettore è zero o uno, la capacità del lettore tiene traccia della capacità dell'istanza DB di scrittura. Pertanto, per consentire alle istanze DB Reader di interrompersi automaticamente nella più ampia gamma di situazioni, impostate la loro priorità su un valore superiore a uno.

Il tempo necessario per riprendere un'istanza Reader potrebbe essere leggermente più lungo rispetto a quello necessario per riprendere un'istanza Writer. Per una risposta più rapida in caso di interruzione delle istanze, connettiti all'endpoint del cluster.

In che modo Aurora Serverless v2 la pausa automatica funziona per i cluster con istanze assegnate

Qualsiasi istanza DB fornita nel cluster Aurora DB non verrà messa in pausa automaticamente. Solo Aurora Serverless v2 Le istanze DB, con la classe `db.serverless` instance, possono utilizzare la funzione di pausa automatica.

Quando il cluster Aurora contiene istanze DB di cui è stato effettuato il provisioning, qualsiasi Aurora Serverless v2 l'istanza writer non viene messa in pausa automaticamente. Ciò è dovuto al requisito che l'istanza writer rimanga disponibile mentre tutte le istanze di Reader sono attive. Il fatto che Aurora Serverless v2 lo scrittore rimane attivo significa anche che qualsiasi Aurora Serverless v2 le istanze reader con priorità di failover 0 e 1 non verranno messe in pausa automaticamente in un cluster ibrido contenente istanze di cui è stato eseguito il provisioning.

Monitoraggio dei cluster Aurora che utilizzano la pausa automatica

Per monitorare Aurora, è necessario conoscere già le procedure di monitoraggio Monitoraggio dei parametri di Amazon Aurora con Amazon CloudWatch e le CloudWatch metriche elencate in. Riferimento alle metriche per Tieni presente che ci sono alcune considerazioni speciali quando monitori i cluster Aurora che utilizzano la funzione di pausa automatica:

- Possono esserci periodi di tempo in cui Aurora Serverless v2 le istanze non registrano i dati di registro e la maggior parte delle metriche perché le istanze sono in pausa. Le uniche metriche inviate CloudWatch mentre un'istanza è in pausa sono lo zero percento per e e lo zero per`CPUUtilization`. `ACUUtilization` `ServerlessDatabaseCapacity`
- Puoi verificare se Aurora Serverless v2 le istanze si interrompono più o meno frequentemente del previsto. A tale scopo, controlla la frequenza con cui la `ServerlessDatabaseCapacity` metrica passa da un valore diverso da zero a zero e per quanto tempo rimane zero. Se le istanze non rimangono in pausa per tutto il tempo previsto, non risparmierai quanto potresti sui costi. Se le istanze vengono messe in pausa e riprendono più frequentemente del previsto, il cluster potrebbe avere una latenza non necessaria nel rispondere alle richieste di connessione. Per informazioni sui fattori che influiscono sulla frequenza e sulla frequenza Aurora Serverless v2 le istanze possono mettere in pausa, vedere Prerequisiti e limitazioni per Aurora Serverless v2 funzione di pausa automaticaSituazioni in cui Aurora Serverless v2 non si mette in pausa automaticamente, e. Risoluzione dei problemi per Aurora Serverless v2 pausa automatica
- È inoltre possibile esaminare un file di registro che registra le operazioni automatiche di pausa e ripresa per un Aurora Serverless v2 istanza. Se un'istanza non si è messa in pausa dopo la scadenza dell'intervallo di timeout, questo file di registro include anche il motivo per cui la pausa automatica non è avvenuta. Per ulteriori informazioni, consulta Monitoraggio Aurora Serverless v2 mettere in pausa e riprendere l'attività.

Argomenti

- Verifica se un Aurora Serverless v2 l'istanza è in pausa
- Eventi per le operazioni di pausa e ripresa automatica
- Come funziona la pausa automatica con Performance Insights e Enhanced Monitoring
- In che modo Aurora Serverless v2 la funzione di pausa automatica interagisce con le metriche Aurora

Verifica se un Aurora Serverless v2 l'istanza è in pausa

Per determinare se un Aurora Serverless v2 l'istanza è in stato di pausa, puoi osservare la `ACUUtilization` metrica dell'istanza. Tale metrica ha un valore pari a zero mentre l'istanza è in pausa.

Mentre un Aurora Serverless v2 l'istanza è in pausa, il suo valore di stato è ancora indicato come Disponibile. Lo stesso vale quando è in pausa Aurora Serverless v2 l'istanza è in fase di ripresa. Questo perché puoi connetterti correttamente a tale istanza, anche se la connessione presenta un leggero ritardo.

Qualsiasi metrica relativa alla disponibilità delle istanze Aurora considera il periodo in cui l'istanza è in pausa come il periodo in cui l'istanza era disponibile.

Eventi per le operazioni di pausa e ripresa automatica

Aurora emette eventi per Aurora Serverless v2 istanze in cui le operazioni di pausa e ripresa automatica iniziano, terminano o vengono annullate. Gli eventi relativi alla funzione di pausa automatica sono `RDS-EVENT-0370` terminati. `RDS-EVENT-0374` Per informazioni dettagliate su questi eventi, vedere. Eventi di istanza database

Come funziona la pausa automatica con Performance Insights e Enhanced Monitoring

Mentre un Aurora Serverless v2 l'istanza è in pausa, Aurora non raccoglie informazioni di monitoraggio per quell'istanza tramite Performance Insights o Enhanced Monitoring. Quando l'istanza viene ripresa, potrebbe verificarsi un breve ritardo prima che Aurora riprenda a raccogliere tali informazioni di monitoraggio.

In che modo Aurora Serverless v2 la funzione di pausa automatica interagisce con le metriche Aurora

Mentre un Aurora Serverless v2 l'istanza è in pausa, non emette la maggior parte delle CloudWatch metriche né scrive alcuna informazione nei registri del database. Le uniche metriche inviate CloudWatch mentre un'istanza è in pausa sono lo zero percento per e e lo zero per`CPUUtilization`. `ACUUtilization` `ServerlessDatabaseCapacity`

Quando CloudWatch le statistiche di calcolo riguardano la disponibilità e l'operatività dell'istanza o del cluster, Aurora Serverless v2 le istanze sono considerate disponibili durante il periodo di pausa.

Quando si avvia un'RDSAPIazione AWS CLI o per descrivere o scaricare i log per una pausa Aurora Serverless v2 istanza, l'istanza viene riattivata automaticamente per rendere disponibili le informazioni di registro.

Esempio di metriche CloudWatch

AWS CLI Gli esempi seguenti mostrano come è possibile osservare la variazione della capacità di un'istanza nel tempo. Durante il periodo di tempo, l'istanza si interrompe automaticamente e poi riprende. Mentre è in pausa, la `ServerlessDatabaseCapacity` metrica riporta un valore pari a zero. Per determinare se l'istanza è stata messa in pausa in qualsiasi momento durante il periodo di tempo, controlliamo se la capacità minima durante quel periodo di tempo era pari a zero.

Il seguente esempio di Linux rappresenta un Aurora Serverless v2 istanza che è stata messa automaticamente in pausa per qualche tempo. Li campioniamo `ServerlessDatabaseCapacity` ogni minuto, per un periodo di tre minuti. Il ACU valore minimo di 0,0 conferma che l'istanza è stata messa in pausa a un certo punto durante ogni minuto.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '3 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:11:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:12:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
```

Successivamente, tentiamo di stabilire una connessione all'elemento in pausa Aurora Serverless v2 istanza. In questo esempio, utilizziamo intenzionalmente una password errata in modo che il tentativo di connessione non abbia successo. Nonostante l'errore, il tentativo di connessione fa sì che Aurora riprenda l'istanza in pausa.

```
$ mysql -h my_cluster_endpoint.rds.amazonaws.com -u admin -pwrong-password

ERROR 1045 (28000): Access denied for user 'admin'@'ip_address' (using password: YES)
```

Il seguente esempio di Linux dimostra che l'istanza in pausa è stata ripresa, è rimasta inattiva per circa cinque minuti e poi è tornata allo stato di pausa. L'istanza è ripresa con una capacità di 2.0. ACUs Quindi si è leggermente ridimensionata, ad esempio per eseguire alcune operazioni di pulizia interna. Poiché non ha ricevuto alcun tentativo di connessione utente entro il periodo di timeout di cinque minuti, è passato dalla versione 4.0 ACUs direttamente allo stato di pausa.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '8 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
DATAPOINTS 2.0 2024-08-02T22:14:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:15:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:16:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:17:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:18:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:19:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:20:00+00:00 None
```

Se il rapporto avesse lo scopo di mostrare la velocità con cui l'istanza si è scalata per gestire il carico di lavoro, potremmo specificare `Maximum` la statistica anziché. `Minimum` Per la pianificazione della capacità e la stima dei costi, potremmo specificare un periodo di tempo più lungo e utilizzare la statistica. `Average` In questo modo, potremmo determinare i valori di capacità tipici durante i periodi di alta attività, bassa attività e stato di pausa. Per esaminare il comportamento durante gli orari precisi di pausa e ripresa, potremmo specificare un periodo di un secondo ed esaminare un intervallo di tempo più breve. I valori del timestamp nell'output, ad esempio`2024-08-02T22:13:00+00:00`, mostrano il formato per specificare parametri precisi per le opzioni and. `--start-time` `--end-time`

Risoluzione dei problemi per Aurora Serverless v2 pausa automatica

Se lo trovi Aurora Serverless v2 le istanze non si interrompono tutte le volte che ti aspetti, verifica le seguenti possibili cause:

- Verifica che la versione di Aurora in esecuzione supporti una capacità minima pari a zero. ACUs Per le gamme di capacità delle diverse versioni di Aurora, vedere. Aurora Serverless v2 capacità
- Verificare che il valore di capacità minima per il cluster sia impostato su zeroACUs.
- Verifica che l'istanza in questione stia effettivamente utilizzando il Aurora Serverless v2 classe di istanza`db.serverless`, non una delle classi di istanze fornite.
- Verifica che il cluster non stia utilizzando nessuna delle funzionalità o impostazioni incompatibili di. Prerequisiti e limitazioni per Aurora Serverless v2 funzione di pausa automatica
- Esamina il file di registro che mostra quando Aurora Serverless v2 le istanze sono state messe in pausa, riprese o Aurora non è stata in grado di mettere in pausa o riprendere un'istanza per qualche motivo. Per ulteriori informazioni, consulta Monitoraggio Aurora Serverless v2 mettere in pausa e riprendere l'attività.
- Controlla se alcuni client o applicazioni mantengono le connessioni aperte per lunghi periodi di tempo. Al contrario, controlla se le applicazioni che utilizzano le funzioni RDS Data API o Lambda inviano richieste frequenti in modo che l'istanza non rimanga mai inattiva abbastanza a lungo da poter essere messa in pausa. Puoi esaminare CloudWatch metriche come e. `ConnectionAttempts` `DatabaseConnections` Per ulteriori informazioni, consulta Parametri a livello di istanza per Amazon Aurora.
- Se un'istanza Reader si interrompe raramente, se non mai, verificate la sua priorità di failover. Se il lettore viene utilizzato per il ridimensionamento della lettura e non come standby in caso di failover, impostatelo su una priorità compresa tra 2 e 15.
- Se l'istanza Writer si interrompe raramente, se non mai, verificate l'utilizzo delle istanze Reader. Lo scrittore può mettersi in pausa solo se l'intero cluster è inattivo. Questo perché una connessione a qualsiasi istanza del lettore fa sì che lo scrittore riprenda.
- Se l'applicazione riceve dei timeout durante le richieste di connessione mentre Aurora Serverless v2 le istanze sono in fase di ripresa, valuta la possibilità di allungare l'intervallo di timeout utilizzato dal codice dell'applicazione o dal framework di database sottostante. Timeout di connessione più lunghi riducono la possibilità di connessioni non riuscite mentre Aurora Serverless v2 le istanze stanno riprendendo. Tuttavia, i timeout più lunghi possono anche rallentare l'individuazione di problemi di disponibilità nel cluster da parte dell'applicazione.

  Al contrario, valuta la possibilità di allungare l'intervallo di pausa automatica in modo che le applicazioni non incontrino istanze in pausa con la stessa frequenza.

  Se non esiste un equilibrio logico tra il comportamento di pausa automatica e la reattività del cluster per l'applicazione, quel cluster potrebbe non essere un buon candidato per l'utilizzo della funzionalità di pausa automatica.

- Se stimate per quanto tempo Aurora Serverless v2 i casi verranno messi in pausa, tieni presente che ci sono fattori che rendono poco pratico fare previsioni precise.
  - Le istanze potrebbero riprendere periodicamente per eseguire operazioni di manutenzione, aggiornamenti di versioni minori o applicare modifiche ai gruppi di parametri.
  - Per i cluster Multi-AZ, ci sono situazioni in cui la ripresa di un'istanza causa il ripristino anche di altre istanze. La ripresa di un lettore fa sempre sì che lo scrittore riprenda. La ripresa del writer fa sì che tutti i lettori con priorità di failover 0 e 1 riprendano sempre.

  Consigliamo di misurare il tempo di pausa effettivo su un periodo di diversi giorni, con un carico di lavoro realistico. Utilizza quindi queste misurazioni per impostare una linea di base per la percentuale di tempo che puoi aspettarti che un'istanza venga messa in pausa.
- Potresti scoprire che le operazioni interne come My SQL purge, My SQL event scheduler, Postgre SQL autovacuum o Postgre SQL job programmati tramite l'estensione non sono in esecuzione o non vengono completate. `pg_cron` L'istanza non riprende automaticamente a eseguire tali operazioni che non comportano una connessione utente al database. Se tali operazioni interne sono in corso alla scadenza del timeout di pausa automatica, le attività di manutenzione come My SQL purge e Postgre autovacuum vengono annullate. SQL I lavori pianificati dall'utilità di pianificazione dei miei SQL eventi o dall'SQL`pg_cron`estensione Postgre vengono annullati anche se sono in corso quando Aurora avvia l'operazione di pausa.

  Se devi assicurarti che l'istanza sia periodicamente attiva per eseguire operazioni pianificate, puoi avviare una connessione per riprendere l'istanza prima dell'inizio del processo. Puoi anche aumentare l'intervallo di timeout della pausa automatica in modo che operazioni come l'autovacuum possano essere eseguite più a lungo dopo che l'attività dell'utente è terminata. È inoltre possibile utilizzare meccanismi come le funzioni Lambda per eseguire le operazioni del database in base a una pianificazione, in modo da riattivare automaticamente l'istanza se necessario.

Considerazioni sulla progettazione delle applicazioni per Aurora Serverless v2 funzione di pausa automatica

Quando un Aurora Serverless v2 L'istanza DB riprende dopo essere stata messa in pausa automaticamente, inizia con una capacità relativamente piccola e da lì si espande verso l'alto. Questa capacità iniziale si applica anche se l'istanza DB aveva una capacità maggiore immediatamente prima di essere messa in pausa automaticamente.

Utilizzate questa funzionalità con applicazioni che possono tollerare un intervallo di circa 15 secondi durante la creazione di una connessione. Ciò rappresenta il caso tipico in cui un Aurora Serverless v2 l'istanza viene ripresa a causa di una o un numero limitato di connessioni in entrata. Se l'istanza viene messa in pausa per più di 24 ore, il tempo di ripresa potrebbe essere più lungo.

Se l'applicazione era già in uso Aurora Serverless v1 e la sua funzione di pausa automatica, è possibile che questi intervalli di timeout siano già in vigore per i tentativi di connessione. Se lo stai già utilizzando Aurora Serverless v2 in combinazione con la funzione Aurora stop/start cluster, il tempo di ripresa per la pausa automatica Aurora Serverless v2 le istanze sono in genere molto più brevi del tempo necessario per avviare un cluster interrotto.

Quando codifichi la logica di connessione nell'applicazione, riprova la connessione se il primo tentativo restituisce un errore che ha una causa temporanea. (Se l'errore è dovuto a un errore di autenticazione, correggete le credenziali prima di riprovare.) Un errore che si verifica immediatamente dopo la ripresa potrebbe essere un timeout o un errore relativo ai limiti del database. Un nuovo tentativo può risolvere i problemi nei casi più rari in cui Aurora Serverless v2 l'istanza viene ripresa a causa di un numero elevato di richieste di connessione simultanee. In tal caso, l'elaborazione di alcune connessioni potrebbe richiedere più tempo del normale o superare il limite di connessioni simultanee al primo tentativo.

Durante lo sviluppo e il debug delle applicazioni, non lasciate le sessioni client o gli strumenti di programmazione con connessioni aperte al database. Aurora non metterà in pausa un'istanza se sono aperte connessioni avviate dall'utente, indipendentemente dal fatto che le connessioni non eseguano istruzioni o transazioni. SQL Quando uno Aurora Serverless v2 l'istanza in un cluster Aurora non può essere messa in pausa, inoltre potrebbe essere impedita la sospensione di altre istanze del cluster. Per ulteriori informazioni, consulta Situazioni in cui Aurora Serverless v2 non si mette in pausa automaticamente.

Aurora emette eventi quando un Aurora Serverless v2 L'istanza DB inizia a riprendere, termina la ripresa e se l'istanza non riesce a riprendere per qualche motivo. Per informazioni dettagliate su questi eventi, consulta. Eventi di istanza database

Migrazione verso Aurora Serverless v2

Per convertire un cluster DB esistente da utilizzare Aurora Serverless v2, puoi fare quanto segue:

- Eseguire l'aggiornamento da un cluster Aurora con provisioning.
- Effettua l'aggiornamento da un Aurora Serverless v1 cluster.

- Esegui la migrazione da un database locale a un Aurora Serverless v2 cluster.

Quando il cluster aggiornato esegue la versione del motore appropriata, come elencato in Requisiti e limitazioni per Aurora Serverless v2, puoi iniziare ad aggiungere Aurora Serverless v2 Istanze DB ad esso. La prima istanza database aggiunta al cluster aggiornato deve essere un'istanza database con provisioning. Quindi puoi passare l'elaborazione per il carico di lavoro di scrittura, il carico di lavoro di lettura o entrambi a Aurora Serverless v2 Istanze DB.

Indice

- Aggiornamento o modifica dei cluster esistenti da utilizzare Aurora Serverless v2
  - Percorsi di aggiornamento per i cluster compatibili con MySQL da utilizzare Aurora Serverless v2
  - Percorsi di aggiornamento per i cluster compatibili con PostgreSQL da utilizzare Aurora Serverless v2
- Passaggio da un cluster fornito a Aurora Serverless v2
- Confronto di Aurora Serverless v2 e Aurora Serverless v1
  - Confronto tra Aurora Serverless v2 e Aurora Serverless v1 requisiti
  - Confronto tra Aurora Serverless v2 e Aurora Serverless v1 scalabilità e disponibilità
  - Confronto tra Aurora Serverless v2 e Aurora Serverless v1 supporto delle funzionalità
  - Adattamento Aurora Serverless v1 casi d'uso per Aurora Serverless v2
- Aggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2
  - Cluster database compatibili con Aurora MySQL
  - Cluster database compatibili con Aurora PostgreSQL
- Migrazione da un database locale a Aurora Serverless v2

> **Note**
>
> Questi argomenti descrivono come convertire un cluster database esistente. Per informazioni sulla creazione di un nuovo Aurora Serverless v2 Cluster DB, vedere Creazione di un cluster DB che utilizza Aurora Serverless v2.

Aggiornamento o modifica dei cluster esistenti da utilizzare Aurora Serverless v2

Se il cluster predisposto dispone di una versione del motore che supporta Aurora Serverless v2, passaggio a Aurora Serverless v2 non richiede un aggiornamento. In tal caso, puoi aggiungere Aurora Serverless v2 Istanze DB al cluster originale. Puoi cambiare il cluster per utilizzarli tutti Aurora Serverless v2 Istanze DB. Puoi anche usare una combinazione di Aurora Serverless v2 e ha effettuato il provisioning di istanze DB nello stesso cluster DB. Per le versioni del motore Aurora che supportano Aurora Serverless v2, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v2.

Se utilizzi una versione inferiore del motore che non supporta Aurora Serverless v2, esegui questi passaggi generali:

1. Aggiornare il cluster.
2. Creare un'istanza database di scrittura con provisioning per il cluster aggiornato.
3. Modifica il cluster da utilizzare Aurora Serverless v2 Istanze DB.

> **Important**
>
> Quando si esegue l'aggiornamento di una versione principale a una Aurora Serverless v2-versione compatibile che utilizza il ripristino o la clonazione di istantanee, la prima istanza DB da aggiungere al nuovo cluster deve essere un'istanza DB di cui è stato effettuato il provisioning. Questa aggiunta avvia la fase finale del processo di aggiornamento.
> Fino a quando non si verifica la fase finale, il cluster non dispone dell'infrastruttura necessaria per Aurora Serverless v2 supporto. Pertanto, i cluster aggiornati iniziano sempre con un'istanza database di scrittura con provisioning. È quindi possibile convertire o eseguire il failover dell'istanza DB fornita in un Aurora Serverless v2 uno.

Aggiornamento da Aurora Serverless v1 in Aurora Serverless v2 prevede la creazione di un cluster fornito come fase intermedia. Vengono quindi eseguiti gli stessi passaggi di aggiornamento utilizzati quando si inizia con un cluster con provisioning.

Percorsi di aggiornamento per i cluster compatibili con MySQL da utilizzare Aurora Serverless v2

Se il cluster originale esegue Aurora MySQL, è necessario scegliere la procedura appropriata in base alla versione e alla modalità del motore del cluster.

Se il cluster Aurora MySQL originale è	Fate questo per passare a Aurora Serverless v2
Cluster con provisioning che esegue Aurora MySQL versione 3, compatibile con MySQL 8.0	Questa è la fase finale per tutte le conversioni da cluster Aurora MySQL esistenti. Se necessario, eseguire un aggiornamento della versione secondaria alla versione 3.02.0 o successiva. Utilizzare un'istanza database con provisioning per l'istanza database di scrittura. Aggiungine uno Aurora Serverless v2 istanza DB del lettore. Eseguire un failover per convertire l'istanza in istanza database di scrittura. (Facoltativo) Converti altre istanze DB assegnate nel cluster in Aurora Serverless v2. Oppure aggiungi nuovo Aurora Serverless v2 istanze DB e rimuovi le istanze DB assegnate. Per la procedura completa e alcuni esempi, consultare Passaggio da un cluster fornito a Aurora Serverless v2.
Cluster con provisioning che esegue Aurora MySQL versione 2, compatibile con MySQL 5.7	Eseguire un aggiornamento della versione principale ad Aurora MySQL versione 3.02.0 o successive. Quindi segui la procedura per Aurora MySQL versione 3 per cambiare il cluster da utilizzare Aurora Serverless v2 Istanze DB.

Se il cluster Aurora MySQL originale è	Fate questo per passare a Aurora Serverless v2
Aurora Serverless v1 cluster che esegue Aurora MySQL versione 2, compatibile con MySQL 5.7	Per aiutarti a pianificare la conversione da Aurora Serverless v1, consulta Confronto di Aurora Serverless v2 e Aurora Serverless v1 prima. Quindi segui la procedura in Aggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2.

Percorsi di aggiornamento per i cluster compatibili con PostgreSQL da utilizzare Aurora Serverless v2

Se il cluster originale esegue Aurora PostgreSQL, scegliere la procedura appropriata in base alla versione e alla modalità del motore del cluster.

Se il cluster Aurora PostgreSQL originale è	Fate questo per passare a Aurora Serverless v2
Cluster con provisioning che esegue Aurora PostgreSQL versione 13	Questa è la fase finale per tutte le conversioni da cluster Aurora PostgreSQL esistenti. Se necessario, eseguire un aggiornamento della versione secondaria alla versione 13.6 o successiva. Aggiungere un'istanza database con provisioning per l'istanza database di scrittura. Aggiungine uno Aurora Serverless v2 istanza DB del lettore. Esegui un failover per farlo Aurora Serverless v2 installa l'istanza Writer DB. (Facoltativo) Converti altre istanze DB assegnate nel cluster in Aurora Serverless v2. Oppure aggiungi nuovo Aurora Serverless v2 istanze DB e rimuovi le istanze DB assegnate.

Se il cluster Aurora PostgreSQL originale è	Fate questo per passare a Aurora Serverless v2
	Per la procedura completa e alcuni esempi, consultare Passaggio da un cluster fornito a Aurora Serverless v2.
Cluster con provisioning che esegue Aurora PostgreSQL versione 11 o 12	Eseguire un aggiornamento della versione principale ad Aurora PostgreSQL versione 13.6 o successiva. Quindi segui la procedura per Aurora PostgreSQL versione 13 per cambiare il cluster da utilizzare Aurora Serverless v2 Istanze DB.
Aurora Serverless v1 cluster che esegue Aurora PostgreSQL versione 11 o 13	Per aiutarti a pianificare la conversione da Aurora Serverless v1, consulta Confronto di Aurora Serverless v2 e Aurora Serverless v1 prima. Quindi segui la procedura in Aggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2.

Passaggio da un cluster fornito a Aurora Serverless v2

Per passare da un cluster predisposto all'uso Aurora Serverless v2, segui questi passaggi:

1. Verifica se il cluster fornito deve essere aggiornato per essere utilizzato con Aurora Serverless v2 Istanze DB. Per le versioni Aurora compatibili con Aurora Serverless v2, consulta Requisiti e limitazioni per Aurora Serverless v2.

   Se il cluster fornito utilizza una versione del motore che non è disponibile per Aurora Serverless v2, aggiorna la versione del motore del cluster:

   - Se disponi di un cluster con provisioning compatibile con MySQL 5.7, segui le istruzioni di aggiornamento per Aurora MySQL versione 3. Utilizzare la procedura descritta in Come eseguire un aggiornamento in loco.

- Se disponi di un cluster con provisioning compatibile con PostgreSQL che esegue PostgreSQL versione 11 o 12, segui le istruzioni di aggiornamento per Aurora PostgreSQL versione 13. Utilizzare la procedura descritta in Esecuzione di un aggiornamento della versione principale.

2. Configura qualsiasi altra proprietà del cluster in modo che corrisponda a Aurora Serverless v2 requisiti daRequisiti e limitazioni per Aurora Serverless v2.
3. Definire la configurazione di dimensionamento per il cluster. Segui la procedura riportata in Impostazione del Aurora Serverless v2 intervallo di capacità per un cluster.
4. Aggiungi uno o più Aurora Serverless v2 Istanze DB al cluster. Eseguire la procedura generale descritta in Aggiunta di repliche di Aurora a un cluster di database. Per ogni nuova istanza DB, specifica il nome speciale della classe di istanza DB Serverless nella AWS Management Console AWS CLI o `db.serverless` nell'API Amazon RDS.

   In alcuni casi, è possibile che nel cluster siano già presenti una o più istanze database di lettura con provisioning. In tal caso, puoi convertire uno dei lettori in Aurora Serverless v2 Istanza DB invece di creare una nuova istanza DB. A tale scopo, segui la procedura in Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2.
5. Esegui un'operazione di failover per creare una delle Aurora Serverless v2 DB installa l'istanza DB writer per il cluster.
6. (Facoltativo) Converti qualsiasi istanza DB fornita in Aurora Serverless v2o rimuoverle dal cluster. Eseguire la procedura generale descritta in Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2 o Eliminazione di un'istanza database da un cluster database Aurora.

> **Tip**
>
> La rimozione delle istanze database con provisioning non è obbligatoria. È possibile configurare un cluster contenente entrambi Aurora Serverless v2 e istanze DB predisposte. Tuttavia, finché non si acquisisce dimestichezza con le prestazioni e le caratteristiche di scalabilità di Aurora Serverless v2 Istanze DB, ti consigliamo di configurare i cluster con istanze DB tutte dello stesso tipo.

L' AWS CLI esempio seguente mostra il processo di switchover utilizzando un cluster con provisioning che esegue Aurora MySQL versione 3.02.0. Il cluster è denominato `mysql-80`. Inizialmente il cluster include due istanze database con provisioning denominate `provisioned-instance-1` e `provisioned-instance-2`, ovvero un'istanza di scrittura e una di lettura. Entrambe usano la classe di istanza database `db.r6g.large`.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' --output text
mysql-80
provisioned-instance-2     False
provisioned-instance-1     True

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-1 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-1     db.r6g.large

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-2 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-2     db.r6g.large
```

Viene creata una tabella contenente alcuni dati. In questo modo, è possibile confermare che i dati e il funzionamento del cluster sono gli stessi prima e dopo il passaggio.

```
mysql> create database serverless_v2_demo;
mysql> create table serverless_v2_demo.demo (s varchar(128));
mysql> insert into serverless_v2_demo.demo values ('This cluster started with a
 provisioned writer.');
Query OK, 1 row affected (0.02 sec)
```

Come prima cosa, al cluster viene aggiunto un intervallo di capacità. Altrimenti, viene visualizzato un errore quando ne aggiungiamo uno Aurora Serverless v2 Istanze DB al cluster. Se utilizziamo il AWS Management Console per questa procedura, quel passaggio è automatico quando aggiungiamo il primo Aurora Serverless v2 Istanza DB.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql

An error occurred (InvalidDBClusterStateFault) when calling the CreateDBInstance
 operation:
Set the Serverless v2 scaling configuration on the parent DB cluster before creating a
 Serverless v2 DB instance.

$ # The blank ServerlessV2ScalingConfiguration attribute confirms that the cluster
 doesn't have a capacity range set yet.
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 --query
 'DBClusters[*].ServerlessV2ScalingConfiguration'
```

```
[]

$ aws rds modify-db-cluster --db-cluster-identifier mysql-80 \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
{
  "DBClusterIdentifier": "mysql-80",
  "ServerlessV2ScalingConfiguration": {
    "MinCapacity": 0.5,
    "MaxCapacity": 16
  }
}
```

Ne creiamo due Aurora Serverless v2 lettori che sostituiranno le istanze DB originali. Questa operazione viene eseguita specificando la classe di istanza database `db.serverless` per le nuove istanze database.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 --db-
cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-2 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-2",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ # Wait for both DB instances to finish being created before proceeding.
$ aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
 && \
  aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-2
```

Eseguiamo un failover per creare uno dei Aurora Serverless v2 DB installa il nuovo writer per il cluster.

```
$ aws rds failover-db-cluster --db-cluster-identifier mysql-80 \
  --target-db-instance-identifier serverless-v2-instance-1
{
  "DBClusterIdentifier": "mysql-80",
  "DBClusterMembers": [
    {
      "DBInstanceIdentifier": "serverless-v2-instance-1",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "serverless-v2-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-1",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "Status": "available"
}
```

Sono necessari alcuni secondi per rendere effettiva la modifica. A quel punto, abbiamo un Aurora Serverless v2 scrittore e un Aurora Serverless v2 lettore. Pertanto, le istanze database con provisioning originali non sono più necessarie.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
mysql-80
```

```
serverless-v2-instance-1        True
serverless-v2-instance-2        False
provisioned-instance-2          False
provisioned-instance-1          False
```

L'ultimo passaggio della procedura di conversione prevede l'eliminazione delle istanze database con provisioning.

```
$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-2 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-2",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}

$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-1 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-1",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}
```

Come ultimo controllo, confermiamo che la tabella originale è accessibile e scrivibile dal Aurora Serverless v2 istanza Writer DB.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
+--------------------------------------------------+
1 row in set (0.00 sec)

mysql> insert into serverless_v2_demo.demo values ('And it finished with a Serverless
 v2 writer.');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
2 rows in set (0.01 sec)
```

Ci colleghiamo anche a Aurora Serverless v2 leggiamo l'istanza DB e confermiamo che i dati appena scritti siano disponibili anche lì.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
2 rows in set (0.01 sec)
```

Confronto di Aurora Serverless v2 e Aurora Serverless v1

Se lo stai già utilizzando Aurora Serverless v1, puoi scoprire le principali differenze tra Aurora Serverless v1 e Aurora Serverless v2. Le differenze architetturali, come il supporto per le istanze Reader DB, aprono nuovi tipi di casi d'uso.

È possibile utilizzare le seguenti tabelle per comprendere le differenze più importanti tra Aurora Serverless v2 e Aurora Serverless v1.

Argomenti

- Confronto tra Aurora Serverless v2 e Aurora Serverless v1 requisiti
- Confronto tra Aurora Serverless v2 e Aurora Serverless v1 scalabilità e disponibilità
- Confronto tra Aurora Serverless v2 e Aurora Serverless v1 supporto delle funzionalità
- Adattamento Aurora Serverless v1 casi d'uso per Aurora Serverless v2

Confronto tra Aurora Serverless v2 e Aurora Serverless v1 requisiti

La tabella seguente riassume i diversi requisiti per eseguire il database utilizzando Aurora Serverless v2 oppure Aurora Serverless v1. Aurora Serverless v2 offre versioni superiori dei motori Aurora MySQL e Aurora PostgreSQL DB rispetto a Aurora Serverless v1 fa.

Funzionalità	Aurora Serverless v2 requisito	Aurora Serverless v1 requisito
Motori database	Aurora MySQL, Aurora PostgreSQL	Aurora MySQL, Aurora PostgreSQL
Versioni supportate di Aurora MySQL	Consultare Aurora Serverless v2 con Aurora MySQL.	Consultare Aurora Serverless v1 con Aurora MySQL.
Versioni supportate di Aurora PostgreSQL	Consultare Aurora Serverless v2 con Aurora PostgreSQL.	Consultare Aurora Serverless v1 con Aurora PostgreSQL.
Aggiornamento di un cluster di database	Come per i cluster di database sottoposti a provisioning, puoi eseguire gli aggiornamenti manualmente senza dover attendere che Aurora aggiorni il cluster di database automaticamente. Per ulteriori informazioni, consulta Modifica di un cluster database Amazon Aurora. **Note** Per eseguire un aggiornamento della versione principale da 13.x a 14.x o 15.x per un cluster DB compatibile con Aurora PostgreSQL, la capacità massima del	Gli aggiornamenti della versione secondaria vengono applicati automaticamente non appena diventano disponibili. Per ulteriori informazioni, consulta Aurora Serverless v1 e versioni del motore di database Aurora. È possibile eseguire manualmente gli aggiornamenti della versione principale. Per ulteriori informazioni, consulta Modificare un Aurora Serverless v1 cluster di database.

Funzionalità	Aurora Serverless v2 requisito	Aurora Serverless v1 requisito
	cluster deve essere almeno 2. ACUs	

Funzionalità	Aurora Serverless v2 requisito	Aurora Serverless v1 requisito
Conversione da cluster di database con provisioning	È possibile utilizzare i seguenti metodi: • Aggiungi uno o più Aurora Serverless v2 Reader istanze DB a un cluster predisposto esistente. Per utilizzare Aurora Serverless v2 per chi scrive, esegui un failover su uno dei Aurora Serverless v2 istanze DB. Da utilizzare per l'intero cluster Aurora Serverless v2 Istanze DB, rimuovete tutte le istanze DB Writer fornite dopo aver promosso il Aurora Serverless v2 Istanza DB allo scrittore. • Creare un nuovo cluster con il motore di database e la versione del motore appropriati. Utilizzare un qualsiasi metodo standard. Ad esempio, ripristinare uno snapshot del cluster o creare un clone di un cluster esistente. Scegliere Aurora Serverless v2 per alcune o tutte le istanze DB del nuovo cluster. Se si crea il nuovo cluster tramite la clonazione, non è possibile aggiornar e contemporaneamente	Ripristina l'istantanea del cluster fornito per crearne uno nuovo Aurora Serverless v1 cluster.

Funzionalità	Aurora Serverless v2 requisito	Aurora Serverless v1 requisito
	la versione del motore. Assicurati che sul cluster originale sia già in esecuzione una versione del motore compatibile con Aurora Serverless v2.	
Conversione da Aurora Serverless v1 cluster	Segui la procedura riportata in Aggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2.	Non applicabile
Classi di istanze database disponibili	Classe di istanza database speciale `db.serverless` . Nel AWS Management Console, è etichettato come Serverless.	Non applicabile. Aurora Serverless v1 utilizza la modalità `serverless` motore.
Porta	Qualsiasi porta compatibile con MySQL o PostgreSQL	Solo porta MySQL o PostgreSQL di default
Indirizzo IP pubblico consentito?	Sì	No
Cloud privato virtuale (VPC, Virtual private cloud) obbligatorio?	Sì	Sì. Ciascuno Aurora Serverless v1 il cluster utilizza 2 endpoint di interfaccia e Gateway Load Balancer allocati al tuo VPC.

Confronto tra Aurora Serverless v2 e Aurora Serverless v1 scalabilità e disponibilità

La tabella seguente riassume le differenze tra Aurora Serverless v2 e Aurora Serverless v1 per scalabilità e disponibilità.

Aurora Serverless v2 la scalabilità è più reattiva, più granulare e meno dirompente rispetto alla scalabilità in Aurora Serverless v1. Aurora Serverless v2 può scalare sia modificando la dimensione dell'istanza DB sia aggiungendo altre istanze DB al cluster DB. Può anche scalare aggiungendo cluster in altro Regioni AWS a un database globale Aurora. Al contrario, Aurora Serverless v1 scala solo aumentando o diminuendo la capacità dello scrittore. Tutta la potenza di calcolo per un Aurora Serverless v1 il cluster viene eseguito in un'unica zona di disponibilità e in un'unica Regione AWS.

Funzionalità di scalabilità e alta disponibilità	Aurora Serverless v2 comportamento	Aurora Serverless v1 comportamento
Unità di capacità Aurora minima (ACUs) (Aurora MySQL)	0,5 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.	1 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.
Minimo ACUs (Aurora PostgreSQL)	0,5 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.	2 quando il cluster è in esecuzione, 0 quando il cluster è in pausa.
Massimo ACUs (Aurora MySQL)	256	256
Massimo ACUs (Aurora PostgreSQL)	256	384
Arresto di un cluster	È possibile arrestare e avviare manualmente il cluster utilizzando la stessa funzione di arresto e avvio del cluster valida per i cluster con provisioning.	Il cluster si interrompe automaticamente dopo un timeout. Alla ripresa dell'atti vità, è necessario attendere un po' di tempo prima che il cluster torni disponibile.
Dimensionamento delle istanze database	Scalabilità verso l'alto e verso il basso con un incremento minimo di 0,5. ACUs	Scala verso l'alto e verso il basso raddoppiando o dimezzando il. ACUs
Numero di istanze database	Stesso valore valido per un cluster con provisioning: 1 istanza database di scrittura,	1 istanza database che gestisce sia le operazioni di

Funzionalità di scalabilità e alta disponibilità	Aurora Serverless v2 comportamento	Aurora Serverless v1 comportamento
	fino a 15 istanze database di lettura.	lettura che le operazioni di scrittura.
Il dimensionamento può avvenire durante l'esecuzione di istruzioni SQL?	Sì. Aurora Serverless v2 non richiede l'attesa di un punto di quiete.	No. Ad esempio, il dimension amento attende il completam ento di transazioni con tempi di esecuzione lunghi, tabelle temporanee e blocchi di tabella.
Le istanze database di lettura vengono dimensionate assieme all'istanza di scrittura	Facoltativo	Non applicabile
Spazio di archiviazione massimo	128 TiB	128 TiB
Conservazione della cache del buffer durante il dimension amento	Sì. La cache del buffer viene dimensionata dinamicamente.	No. La cache del buffer viene riavviata a caldo dopo il dimensionamento.
Failover	Sì, con procedura analoga a quella valida per i cluster con provisioning.	Solo modalità "Best effort", soggetto alla disponibilità della capacità. Più lento che in Aurora Serverless v2.
Funzionalità Multi-AZ	Sì, come per le istanze con provisioning. Un cluster Multi-AZ richiede un'istanz a database di lettura in una seconda zona di disponibi lità. Per un cluster Multi-AZ, Aurora esegue il failover Multi-AZ in caso di guasto a livello di zona di disponibilità.	Aurora Serverless v1 i cluster eseguono tutta la loro elaborazione in un'unica AZ. Il ripristino in caso di guasto a livello di zona di disponibilità viene eseguito solo in modalità "Best effort" ed è soggetto alla disponibilità della capacità.

Funzionalità di scalabilità e alta disponibilità	Aurora Serverless v2 comportamento	Aurora Serverless v1 comportamento
Database globali di Aurora	Sì	No
Dimensionamento basato sul carico della memoria	Sì	No
Dimensionamento basato sul carico della CPU	Sì	Sì
Dimensionamento basato sul traffico di rete	Sì, in base alla memoria e al sovraccarico della CPU del traffico di rete. Il parametro `max_connections` rimane costante per evitare l'interruzione delle connessioni durante il dimensionamento.	Sì, in base al numero di connessioni.
Operazione di timeout per gli eventi di dimensionamento	No	Sì
Aggiungere nuove istanze DB al cluster tramite AWS Auto Scaling	Non applicabile. È possibile creare Aurora Serverless v2 leggi le istanze DB nei livelli di promozione da 2 a 15 e lasciale ridimensionate a bassa capacità.	No. Le istanze database di lettura non sono disponibili.

Confronto tra Aurora Serverless v2 e Aurora Serverless v1 supporto delle funzionalità

La tabella seguente si riferisce ai seguenti argomenti:

- Funzionalità disponibili in Aurora Serverless v2 ma non Aurora Serverless v1
- Funzionalità che funzionano in modo diverso tra Aurora Serverless v1 e Aurora Serverless v2
- Funzionalità che al momento non sono disponibili in Aurora Serverless v2

Aurora Serverless v2 include molte funzionalità dei cluster predisposti che non sono disponibili per Aurora Serverless v1.

Funzionalità	Aurora Serverless v2 supporto	Aurora Serverless v1 supporto
Topologia dei cluster	Aurora Serverless v2 è una proprietà delle singole istanze DB. Un cluster può contenere più Aurora Serverless v2 Istanze DB o una combinazi one di Aurora Serverless v2 e istanze DB fornite.	Aurora Serverless v1 i cluster non utilizzano la nozione di istanze DB. Non è possibile modificare il Aurora Serverless v1 proprietà dopo aver creato il cluster.
Parametri di configurazione	È possibile modificare quasi tutti gli stessi parametri come avviene per i cluster con provisioning. Per informazioni dettagliate, consultare Uso di gruppi di parametri per Aurora Serverless v2.	È possibile modificare solo un sottoinsieme di parametri.
Gruppi di parametri	Uso di gruppi di parametri a livello di cluster e gruppi di parametri a livello di database Sono disponibili i parametri con il valore `provisioned` nell'attributo `Supported EngineModes` . Sono molti più parametri rispetto a Aurora Serverless v1.	Solo gruppo di parametri a livello di cluster. Sono disponibili i parametri con il valore `serverless` nell'attributo `Supported EngineModes` .
Crittografia per volume di cluster	Facoltativo	Obbligatorio. Le limitazio ni Limiti relativi a istanze database crittografate Amazon Aurora valgono per tutti Aurora Serverless v1 i cluster.

Funzionalità	Aurora Serverless v2 supporto	Aurora Serverless v1 supporto
Snapshot tra regioni	Sì	L'istantanea deve essere crittografata con la tua chiave AWS Key Management Service (AWS KMS).
Backup automatici conservati dopo l'eliminazione del cluster DB	Sì	No
TLS/SSL	Sì. Stesso supporto disponibile per i cluster con provisioning. Per informazioni sull'utilizzo, consulta Utilizzo di TLS/SSL con Aurora Serverless v2.	Sì. Esistono alcune differenze rispetto al supporto TLS per i cluster con provisioning. Per informazioni sull'utilizzo, consulta Usare TLS/SSL con Aurora Serverless v1.
Clonazione	Solo da e verso versioni del motore DB compatibili con Aurora Serverless v2. Non è possibile utilizzare la clonazione per eseguire l'aggiornamento da Aurora Serverless v1 o da una versione precedente di un cluster fornito.	Solo da e verso versioni del motore DB compatibili con Aurora Serverless v1.
Integrazione con Amazon S3	Sì	Sì
Integrazione con AWS Secrets Manager	Sì	No
Esportazione di snapshot cluster DB in S3	Sì	No
Associazione di un ruolo IAM	Sì	No

Funzionalità	Aurora Serverless v2 supporto	Aurora Serverless v1 supporto
Caricamento dei log su Amazon CloudWatch	Facoltativo. Sei tu a scegliere quali registri attivare e su quali registri caricare. CloudWatch	Tutti i log attivati vengono caricati automaticamente. CloudWatch
API di dati disponibile	Sì	Sì
Editor di query disponibile	Sì	Sì
Approfondimenti sulle prestazioni	Sì	No
Amazon RDS Proxy disponibile	Sì	No
Babelfish per Aurora PostgreSQL disponibile	Sì. Supportato per le versioni PostgreSQL di Aurora compatibili con Babelfish e Aurora Serverless v2.	No

Adattamento Aurora Serverless v1 casi d'uso per Aurora Serverless v2

A seconda del caso d'uso per Aurora Serverless v1, potresti adattare questo approccio per trarne vantaggio Aurora Serverless v2 presenta le seguenti caratteristiche.

Supponiamo di avere un Aurora Serverless v1 cluster con un carico leggero e la cui priorità è mantenere la disponibilità continua riducendo al minimo i costi. Con Aurora Serverless v2, è possibile configurare un'impostazione ACU minima di 0,5, rispetto a un minimo di 1 ACU per Aurora Serverless v1. È possibile aumentare la disponibilità creando una configurazione Multi-AZ, con anche l'istanza Reader DB con un minimo di 0,5 ACUs.

Supponiamo di avere un Aurora Serverless v1 cluster da utilizzare in uno scenario di sviluppo e test. In questo caso, anche il costo è una priorità elevata, ma il cluster non deve essere sempre disponibile. Aurora Serverless v2 può mettere automaticamente in pausa ogni istanza quando è completamente inattiva. A tale scopo, è necessario specificare una capacità minima di 0 ACUs per il cluster, come spiegato in. Ridimensionamento a zero ACUs con pausa e ripresa automatiche per Aurora Serverless v2 È inoltre possibile arrestare manualmente il cluster quando non è necessario e avviarlo quando è il momento del successivo ciclo di test o sviluppo.

Supponiamo di avere un Aurora Serverless v1 cluster con un carico di lavoro pesante. Un cluster equivalente che utilizza Aurora Serverless v2 può scalare con maggiore granularità. Ad esempio, Aurora Serverless v1 scala raddoppiando la capacità, ad esempio da 64 a 128. ACUs Al contrario, il tuo Aurora Serverless v2 L'istanza DB può scalare con incrementi di 0,5 ACU.

Supponiamo che il carico di lavoro richieda una capacità totale superiore a quella disponibile in Aurora Serverless v1. Puoi usarne più Aurora Serverless v2 istanze Reader DB per scaricare le parti del carico di lavoro ad alta intensità di lettura dall'istanza DB di Writer. È inoltre possibile suddividere il carico di lavoro a uso più intensivo di operazioni di lettura tra più istanze database di lettura.

Per un carico di lavoro intensivo di scrittura, potrebbe essere necessario configurare il cluster con un'istanza database di grandi dimensioni di cui è stato effettuato il provisioning come l'istanza di scrittura. Potresti farlo insieme a uno o più Aurora Serverless v2 istanze Reader DB.

Aggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2

> **Important**
>
> AWS ha annunciato la data per end-of-life Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Il processo di aggiornamento di un cluster DB da Aurora Serverless v1 in Aurora Serverless v2 prevede più passaggi. Questo perché non puoi convertire direttamente da Aurora Serverless v1 in Aurora Serverless v2. C'è sempre un passaggio intermedio che prevede la conversione di Aurora Serverless v1 Cluster DB in un cluster predisposto.

Cluster database compatibili con Aurora MySQL

Puoi convertire i tuoi Aurora Serverless v1 Un cluster DB in un cluster DB fornito, quindi utilizza una distribuzione blu/verde per aggiornarlo e convertirlo in un Aurora Serverless v2 Cluster DB. Consigliamo questa procedura per gli ambienti di produzione. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Per utilizzare una distribuzione blu/verde per aggiornare un Aurora Serverless v1 cluster che esegue Aurora MySQL versione 2 (compatibile con MySQL 5.7)

1. Convertire il Aurora Serverless v1 Cluster DB su un cluster Aurora MySQL versione 2 fornito. Segui la procedura riportata in Conversione da Aurora Serverless v1 a fornito.
2. Crea una distribuzione blu/verde. Segui la procedura riportata in Creazione di una distribuzione blu/verde in Amazon .
3. Scegli una versione di Aurora MySQL per il cluster verde compatibile con Aurora Serverless v2, ad esempio 3.04.1.

   Per le versioni compatibili, consulta Aurora Serverless v2 con Aurora MySQL.
4. Modifica l'istanza Writer DB del cluster verde per utilizzare la classe di istanze DB Serverless v2 (db.serverless).

   Per informazioni dettagliate, consultare Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2.
5. Quando hai effettuato l'aggiornamento Aurora Serverless v2 Il cluster DB è disponibile, passa dal cluster blu al cluster verde.

Cluster database compatibili con Aurora PostgreSQL

Puoi convertire i tuoi Aurora Serverless v1 Un cluster DB in un cluster DB fornito, quindi utilizza una distribuzione blu/verde per aggiornarlo e convertirlo in un Aurora Serverless v2 Cluster DB. Consigliamo questa procedura per gli ambienti di produzione. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Blue/Green Deployments per gli aggiornamenti del database.

Per utilizzare una distribuzione blu/verde per aggiornare un Aurora Serverless v1 cluster che esegue Aurora PostgreSQL versione 11

1. Convertire il Aurora Serverless v1 Cluster DB su un cluster Aurora PostgreSQL fornito. Segui la procedura riportata in Conversione da Aurora Serverless v1 a fornito.
2. Crea una distribuzione blu/verde. Segui la procedura riportata in Creazione di una distribuzione blu/verde in Amazon .
3. Scegli una versione di Aurora PostgreSQL per il cluster verde compatibile con Aurora Serverless v2, ad esempio 15.3.

   Per le versioni compatibili, consulta Aurora Serverless v2 con Aurora PostgreSQL.

4. Modifica l'istanza Writer DB del cluster verde per utilizzare la classe di istanze DB Serverless v2 (db.serverless).

   Per informazioni dettagliate, consultare Conversione di uno scrittore o lettore predisposto in Aurora Serverless v2.

5. Quando hai effettuato l'aggiornamento Aurora Serverless v2 Il cluster DB è disponibile, passa dal cluster blu al cluster verde.

Puoi anche aggiornare il tuo Aurora Serverless v1 Cluster DB direttamente da Aurora PostgreSQL versione 11 alla versione 13, convertilo in un cluster DB con provisioning e quindi converti il cluster di cui è stato eseguito il provisioning in un Aurora Serverless v2 Cluster DB.

Per eseguire l'aggiornamento, converti un Aurora Serverless v1 cluster che esegue Aurora PostgreSQL versione 11

1. Convertire il Aurora Serverless v1 Cluster DB su un cluster Aurora PostgreSQL fornito. Segui la procedura riportata in Conversione da Aurora Serverless v1 a fornito.
2. Aggiorna il Aurora Serverless v1 raggruppare in una versione 13 di Aurora PostgreSQL compatibile con Aurora Serverless v2, ad esempio, 13.12. Segui la procedura riportata in Aggiornamento della versione principale.

   Per le versioni compatibili, consulta Aurora Serverless v2 con Aurora PostgreSQL.

3. Aggiungere un Aurora Serverless v2 istanza Reader DB al cluster. Per ulteriori informazioni, consulta Aggiungere un Aurora Serverless v2 reader.
4. Failover su Aurora Serverless v2 Istanza DB:

   a. Seleziona l'istanza Writer DB del cluster DB.

   b. Per Actions (Operazioni), scegliere Failover.

   c. Nella pagina di conferma, scegli Failover.

In Aurora Serverless v1 Cluster DB che eseguono Aurora PostgreSQL versione 13, converti il Aurora Serverless v1 cluster in un cluster DB di cui è stato eseguito il provisioning, quindi converte il cluster di cui è stato eseguito il provisioning in un Aurora Serverless v2 Cluster DB.

Per aggiornare un Aurora Serverless v1 cluster che esegue Aurora PostgreSQL versione 13

1. Convertire il Aurora Serverless v1 Cluster DB su un cluster Aurora PostgreSQL fornito. Segui la procedura riportata in Conversione da Aurora Serverless v1 a fornito.
2. Aggiungi un Aurora Serverless v2 istanza Reader DB al cluster. Per ulteriori informazioni, consulta Aggiungere un Aurora Serverless v2 reader.
3. Failover su Aurora Serverless v2 Istanza DB:
    a. Seleziona l'istanza Writer DB del cluster DB.
    b. Per Actions (Operazioni), scegliere Failover.
    c. Nella pagina di conferma, scegli Failover.
4. Rimuovi l'istanza del lettore.

Migrazione da un database locale a Aurora Serverless v2

Puoi migrare i tuoi database locali a Aurora Serverless v2, proprio come con Aurora MySQL e Aurora PostgreSQL con provisioning.

- Per i database MySQL, puoi usare il comando `mysqldump`. Per ulteriori informazioni, consulta Importazione dei dati in un'istanza database MariaDB o MySQL di Amazon RDS, riducendo i tempi di inattività.
- Per i database PostgreSQL, puoi utilizzare i comandi `pg_dump` e `pg_restore`. Per ulteriori informazioni, consulta il blog post Best practices for migrating PostgreSQL databases to Amazon RDS and Amazon Aurora.

Usare Amazon Aurora Serverless v1

> **Important**
>
> AWS ha annunciato la data per end-of-life Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Amazon Aurora Serverless v1 (Amazon Aurora Serverless versione 1) è una configurazione di scalabilità automatica su richiesta per Amazon Aurora. Un Aurora Serverless v1 Il cluster DB è un cluster DB che aumenta e riduce la capacità di calcolo in base alle esigenze dell'applicazione. Ciò contrasta con i cluster DB forniti da Aurora, per i quali è possibile gestire manualmente la capacità. Aurora Serverless v1 offre un'opzione relativamente semplice ed economica per carichi di lavoro poco frequenti, intermittenti o imprevedibili. È economico perché avvia e dimensiona automaticamente la capacità di calcolo in base all'uso dell'applicazione e si chiude quando non viene utilizzato.

Per ulteriori informazioni sui prezzi, consulta la sezione Prezzi Serverless in MySQL Compatible Edition o PostgreSQL Compatible Edition nel Amazon Aurora pricing pagina.

Aurora Serverless v1 i cluster hanno lo stesso tipo di volume di storage ad alta capacità, distribuito e ad alta disponibilità utilizzato dai cluster DB assegnati.

Per un Aurora Serverless v1 cluster, il volume del cluster è sempre crittografato. Puoi scegliere la chiave di crittografia, ma non puoi disabilitare la crittografia. Ciò significa che è possibile eseguire le stesse operazioni su un Aurora Serverless v1 ciò è possibile su istantanee crittografate. Per ulteriori informazioni, consulta Aurora Serverless v1 e istantanee.

Argomenti

- Disponibilità di aree e versioni per Aurora Serverless v1
- Vantaggi di Aurora Serverless v1
- Casi d'uso per Aurora Serverless v1

- Limitazioni di Aurora Serverless v1
- Requisiti di configurazione per Aurora Serverless v1
- Usare TLS/SSL con Aurora Serverless v1
- In che modo Aurora Serverless v1 funzionamento
- Creare un Aurora Serverless v1 cluster di database
- Ripristino di un Aurora Serverless v1 cluster di database
- Modificare un Aurora Serverless v1 cluster di database
- Dimensionamento Aurora Serverless v1 Capacità del cluster DB manualmente
- Visualizzazione Aurora Serverless v1 Cluster database
- Eliminazione di un Aurora Serverless v1 cluster di database
- Aurora Serverless v1 e versioni del motore di database Aurora

Disponibilità di aree e versioni per Aurora Serverless v1

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni con Aurora e Aurora Serverless v1, consulta Aurora Serverless v1.

Vantaggi di Aurora Serverless v1

Aurora Serverless v1 offre i seguenti vantaggi:

- Più semplice di quanto fornito: Aurora Serverless v1 rimuove gran parte della complessità della gestione delle istanze e della capacità del database.
- Scalabile: Aurora Serverless v1 ridimensiona senza interruzioni la capacità di calcolo e di memoria in base alle esigenze, senza interrompere le connessioni dei client.
- Conveniente: quando si utilizza Aurora Serverless v1, paghi solo per le risorse di database che utilizzi, in base al secondo.
- Storage ad alta disponibilità: Aurora Serverless v1 utilizza lo stesso sistema di storage distribuito con tolleranza ai guasti con replica a sei vie di Aurora per proteggere dalla perdita di dati.

Casi d'uso per Aurora Serverless v1

Aurora Serverless v1 è progettato per i seguenti casi d'uso:

- Applicazioni poco utilizzate – Hai un'applicazione che viene usata soltanto per pochi minuti diverse volte al giorno o alla settimana, come il sito di un blog a volume ridotto. Con Aurora Serverless v1, paghi solo per le risorse di database che utilizzi al secondo.
- Nuove applicazioni – Stai distribuendo una nuova applicazione e non sei sicuro delle dimensioni dell'istanza necessarie. Utilizzando Aurora Serverless v1, è possibile creare un endpoint del database e ridimensionare automaticamente il database in base ai requisiti di capacità dell'applicazione.
- Carichi di lavoro variabili – Esegui un'applicazione poco utilizzata, con picchi che vanno da 30 minuti a diverse ore poche volte al giorno o diverse volte all'anno. Esempi di questo tipo sono le applicazioni per le risorse umane, la redazione del budget e la creazione di report operativi. Con Aurora Serverless v1, non è più necessario predisporre la capacità di picco o media.
- Carichi di lavoro imprevedibili – Esegui carichi di lavoro giornalieri con aumenti improvvisi e imprevedibili dell'attività. Un esempio può essere quello di un sito sul traffico in cui c'è un aumento dell'attività quando inizia a piovere. Con Aurora Serverless v1, il database ridimensiona automaticamente la capacità per soddisfare le esigenze di picco di carico dell'applicazione e si ridimensiona nuovamente al termine del picco di attività.
- Database di sviluppo e test – Gli sviluppatori utilizzano i database durante l'orario di lavoro, ma non ne hanno bisogno nelle notti o nei fine settimana. Con Aurora Serverless v1, il database si spegne automaticamente quando non è in uso.
- Applicazioni multi-tenant: con Aurora Serverless v1, non è necessario gestire individualmente la capacità del database per ogni applicazione del parco applicazioni. Aurora Serverless v1 gestisce la capacità del database individuale per te.

Limitazioni di Aurora Serverless v1

Le seguenti limitazioni si applicano a Aurora Serverless v1:

- **Important**

  AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile,

Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

- Aurora Serverless v1 non supporta le seguenti funzionalità:
  - Database globali di Aurora
  - Repliche di Aurora
  - AWS Identity and Access Management autenticazione del database (IAM)
  - Backtrack in Aurora
  - Flussi di attività di database
  - Autenticazione Kerberos
  - Approfondimenti sulle prestazioni
  - Server proxy per RDS
  - Visualizzazione dei log in AWS Management Console
- Connessioni a un Aurora Serverless v1 Il cluster DB viene chiuso automaticamente se mantenuto aperto per più di un giorno.
- Tutti Aurora Serverless v1 I cluster DB presentano le seguenti limitazioni:
  - Non puoi esportare Aurora Serverless v1 istantanee su bucket Amazon S3.
  - Non puoi usare AWS Database Migration Service e Change Data Capture (CDC) con Aurora Serverless v1 Cluster DB. Solo i cluster Aurora DB con provisioning supportano CDC AWS DMS come origine.
  - Non puoi salvare dati in file di testo in Amazon S3 o caricare dati di file di testo su Aurora Serverless v1 da S3.
  - Non è possibile associare un ruolo IAM a un Aurora Serverless v1 Cluster DB. Tuttavia, è possibile caricare dati su Aurora Serverless v1 da Amazon S3 utilizzando l'`aws_s3`estensione con la `aws_s3.table_import_from_s3` funzione e il `credentials` parametro. Per ulteriori informazioni, consulta Importazione di dati da Amazon S3 in un cluster database Aurora PostgreSQL.
  - Quando si utilizza l'editor di query, viene creato un segreto Secrets Manager con le credenziali per accedere al database. Eliminando le credenziali dall'editor di query, anche il segreto associato viene eliminato da Secrets Manager. Una volta eliminato, però, questo segreto non può essere recuperato.
- Cluster DB Aurora basati su MySQL in esecuzione Aurora Serverless v1 non supportano quanto segue:

- Richiamo di AWS Lambda funzioni dall'interno del cluster Aurora MySQL DB. Tuttavia, AWS Lambda le funzioni possono effettuare chiamate al Aurora Serverless v1 Cluster DB.
- Ripristino di uno snapshot da un'istanza database che non è Aurora MySQL o RDS for MySQL.
- Replica dei dati mediante la replica basata su registri binari (binlog). Questa limitazione è valida indipendentemente dal fatto che il cluster DB Aurora sia basato su MySQL Aurora Serverless v1 è l'origine o la destinazione della replica. Per replicare i dati in un Aurora Serverless v1 Prendi in considerazione l'utilizzo di un cluster DB da un'istanza DB MySQL esterna ad Aurora, ad esempio un'istanza in esecuzione su Amazon. EC2 AWS Database Migration Service Per ulteriori informazioni, consulta la AWS Database Migration Service Guida per l'utente di .
- Creazione di utenti con accesso basato su host (`'username'@'IP_address'`). Questo è perché Aurora Serverless v1 utilizza una flotta di router tra il client e l'host del database per una scalabilità senza interruzioni. L'indirizzo IP che Aurora Serverless Il cluster DB vede è quello dell'host del router e non del tuo client. Per ulteriori informazioni, consulta Aurora Serverless v1 architecture.

  Utilizzare, invece, il carattere jolly (`'username'@'%'`).

- Cluster DB basati su Aurora PostgreSQL in esecuzione Aurora Serverless v1 presentano le seguenti limitazioni:
  - La gestione del piano di query di Aurora PostgreSQL (`apg_plan_management`) non è supportata.
  - La funzionalità di replica logica disponibile in Amazon RDS PostgreSQL e Aurora PostgreSQL non è supportata.
  - Le comunicazioni in uscita come quelle abilitate dalle estensioni Amazon RDS for PostgreSQL non sono supportate. Ad esempio, non è possibile accedere ai dati esterni con l'estensione `postgres_fdw/dblink`. Per ulteriori informazioni sulle estensioni PostgreSQL di RDS, consulta PostgreSQL su Amazon RDS nella Guida per l'utente di RDS.
  - Al momento, alcune query SQL e comandi non sono consigliati. Questi includono blocchi di consulenza a livello di sessione, relazioni temporanee, notifiche asincrone (`LISTEN`) e cursori con hold (`DECLARE name ... CURSOR WITH HOLD FOR query`). Inoltre, i comandi `NOTIFY` impediscono il dimensionamento e non sono consigliati.

    Per ulteriori informazioni, consulta Scalabilità automatica per Aurora Serverless v1.

- Non è possibile impostare la finestra di backup automatico preferita per un Aurora Serverless v1 Cluster DB.

- È possibile impostare la finestra di manutenzione per un Aurora Serverless v1 Cluster DB. Per ulteriori informazioni, consulta Impostazione della finestra di manutenzione preferita del cluster database.

Requisiti di configurazione per Aurora Serverless v1

Quando si crea un Aurora Serverless v1 Cluster DB, presta attenzione ai seguenti requisiti:

- Utilizza i seguenti numeri di porta specifici per ogni motore database:
  - Aurora MySQL – `3306`
  - Aurora PostgreSQL – `5432`
- Crea il tuo Aurora Serverless v1 Cluster DB in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Quando crei un Aurora Serverless v1 Nel cluster DB del tuo VPC, utilizzi due (2) dei cinquanta (50) endpoint Interface e Gateway Load Balancer assegnati al tuo VPC. Questi endpoint vengono creati automaticamente per l'utente. Per aumentare la tua quota, puoi contattare. Supporto Per ulteriori informazioni, consulta la pagina relativa alle quote di Amazon VPC.
- Non puoi dare un Aurora Serverless v1 Cluster DB con un indirizzo IP pubblico. È possibile accedere a un Aurora Serverless v1 Cluster DB solo dall'interno di un VPC.
- Crea sottoreti in diverse zone di disponibilità per il gruppo di sottoreti DB che utilizzi per Aurora Serverless v1 Cluster DB. In altre parole, non è possibile avere più di una sottorete nella stessa zona di disponibilità.
- Modifiche a un gruppo di sottoreti utilizzato da un Aurora Serverless v1 Il cluster DB non viene applicato al cluster.
- È possibile accedere a un Aurora Serverless v1 Cluster DB da AWS Lambda. A tale scopo, è necessario configurare la funzione Lambda in modo che venga eseguita nello stesso VPC del Aurora Serverless v1 Cluster DB. Per ulteriori informazioni su come lavorare con AWS Lambda, consulta Configurazione di una funzione Lambda per accedere alle risorse in un Amazon VPC nella Developer Guide.AWS Lambda

Usare TLS/SSL con Aurora Serverless v1

Per impostazione predefinita, Aurora Serverless v1 utilizza il protocollo Transport Layer Security/ Secure Sockets Layer (TLS/SSL () per crittografare le comunicazioni tra i client e i Aurora Serverless v1 cluster DB. Supporta TLS/SSL versione 1.0, 1.1 e 1.2. Non è necessario configurare il Aurora Serverless v1 Cluster DB per utilizzare TLS/SSL.

Si applicano le seguenti limitazioni:

- Supporto TLS/SSL per Aurora Serverless v1 I cluster DB non sono attualmente disponibili in Cina (Pechino). Regione AWS
- Quando si creano utenti di database per un database basato su Aurora MySQL Aurora Serverless v1 Cluster DB, non utilizzate la clausola per le autorizzazioni SSL. `REQUIRE` In questo modo si impedisce agli utenti di connettersi all'istanza database Aurora.
- Sia per le utilità MySQL Client che PostgreSQL Client, le variabili di sessione che è possibile utilizzare in altri ambienti non hanno alcun effetto quando si utilizza TLS/SSL tra client e Aurora Serverless v1.
- Per MySQL Client, quando ti connetti con la modalità `VERIFY_IDENTITY` di TLS/SSL, devi utilizzare il comando `mysql` compatibile con MySQL 8.0. Per ulteriori informazioni, consulta Connessione a un'istanza database che esegue il motore del database MySQL.

A seconda del client a cui ti connetti Aurora Serverless v1 Cluster DB, potrebbe non essere necessario specificare TLS/SSL per ottenere una connessione crittografata. Ad esempio, per utilizzare il client PostgreSQL per connettersi a un Aurora Serverless v1 Cluster DB che esegue Aurora PostgreSQL Compatible Edition, connettiti normalmente.

```
psql -h endpoint -U user
```

Dopo aver inserito la password, PostgreSQL Client mostra i dettagli della connessione, tra cui la versione TLS/SSL e il codice.

```
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1), server 10.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.
```

Important

Aurora Serverless v1 utilizza le terminazioni di Security/Secure Sockets Layer (TLS/SSL) protocol to encrypt connections by default unless SSL/TLS is disabled by the client application. The TLS/SSL connessione Transport Layer nel parco router. La comunicazione tra il parco router e il Aurora Serverless v1 Il cluster DB si verifica all'interno dei confini della rete interna del servizio.

È possibile controllare lo stato della connessione del client per verificare se la connessione a Aurora Serverless v1 è crittografato con TLS/SSL. `pg_stat_ssl`PostgreSQL e le tabelle `pg_stat_activity` e la `ssl_is_used` sua funzione non mostrano lo stato TLS/SSL per la comunicazione tra l'applicazione client e Aurora Serverless v1. Allo stesso modo, lo stato TLS/SSL non può essere derivato dall'istruzione MySQL. `status`
I parametri del cluster Aurora per `force_ssl` PostgreSQL e per MySQL in precedenza non erano supportati per `require_secure_transport` Aurora Serverless v1. Questi parametri sono ora disponibili per Aurora Serverless v1. Per un elenco completo dei parametri supportati da Aurora Serverless v1, chiama l'operazione API. DescribeEngineDefaultClusterParameters Per ulteriori informazioni sui gruppi di parametri e su Aurora Serverless v1, consulta Gruppi di parametri per Aurora Serverless v1.

Per utilizzare il client MySQL per connettersi a un Aurora Serverless v1 Cluster DB che esegue Aurora MySQL Compatible Edition, specifichi TLS/SSL nella richiesta. L'esempio seguente include il trust store principale di Amazon CA 1 scaricato da Amazon Trust Services, necessario per la riuscita della connessione.

```
mysql -h endpoint -P 3306 -u user -p --ssl-ca=amazon-root-CA-1.pem --ssl-mode=REQUIRED
```

Specifica la password, quando richiesto. Verrà avviato il monitor MySQL. Puoi verificare che la sessione è crittografata utilizzando il comando `status`.

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.5.62, for Linux (x86_64) using readline 5.1
Connection id:          19
Current database:
Current user:           ***@*******
SSL:                    Cipher in use is ECDHE-RSA-AES256-SHA
...
```

Per maggiori informazioni sulla connessione al database Aurora MySQL con MySQL Client, consulta Connessione a un'istanza database che esegue il motore del database MySQL.

Aurora Serverless v1 supporta tutte le modalità TLS/SSL disponibili per MySQL Client (`mysql`) e PostgreSQL Client (), incluse quelle elencate nella tabella seguente. `psql`

Descrizione della modalità TLS/SSL	mysql	psql
Connettiti senza utilizzare TLS/SSL.	DISABLED	disattiva
Prova prima la connessione utilizzando TLS/SSL, ma se necessario, torna all'uso senza SSL.	PREFERRED	prefer (impostazione predefinita)
Applica l'uso di TLS/SSL.	REQUIRED	require
Applica TLS/SSL e verifica la CA.	VERIFY_CA	verify-ca
Applica TLS/SSL, verifica la CA e verifica il nome host della CA.	VERIFY_IDENTITY	verify-full

Aurora Serverless v1 utilizza certificati wildcard. Se specifichi l'opzione "verifica CA" o "verifica CA e nome host CA" quando utilizzi TLS/SSL, scarica innanzitutto il trust store CA 1 radice di Amazon da Amazon Trust Services. Dopo aver fatto ciò, puoi identificare questo file in formato PEMA nel comando client. Per farlo utilizzando PostgreSQL Client:

In Linux, macOS, oppure Unix:

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Per ulteriori informazioni sull'utilizzo del database Aurora PostgreSQL utilizzando Postgres Client, consulta Connessione a un'istanza database che esegue il modulo di motore del database PostgreSQL.

Per informazioni generali sulla connessione ai cluster database Aurora, consulta Connessione a un cluster database Amazon Aurora.

Suite di crittografia supportate per connessioni a Aurora Serverless v1 Cluster database

Utilizzando suite di cifratura configurabili, è possibile avere maggiore controllo sulla sicurezza delle connessioni al database. È possibile specificare un elenco di suite di crittografia che si desidera abilitare per proteggere le connessioni TLS/SSL client al database. Con le suite di cifratura, è possibile controllare la crittografia di connessione accettata dal server di database. In questo modo si impedisce l'uso di sistemi di crittografia non sicuri o obsoleti.

Aurora Serverless v1 I cluster DB basati su Aurora MySQL supportano le stesse suite di crittografia dei cluster DB forniti da Aurora MySQL. Per informazioni su queste suite di crittografia, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora MySQL.

Aurora Serverless v1 I cluster DB basati su Aurora PostgreSQL non supportano le suite di crittografia.

In che modo Aurora Serverless v1 funzionamento

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Di seguito, puoi scoprire come Aurora Serverless v1 funziona.

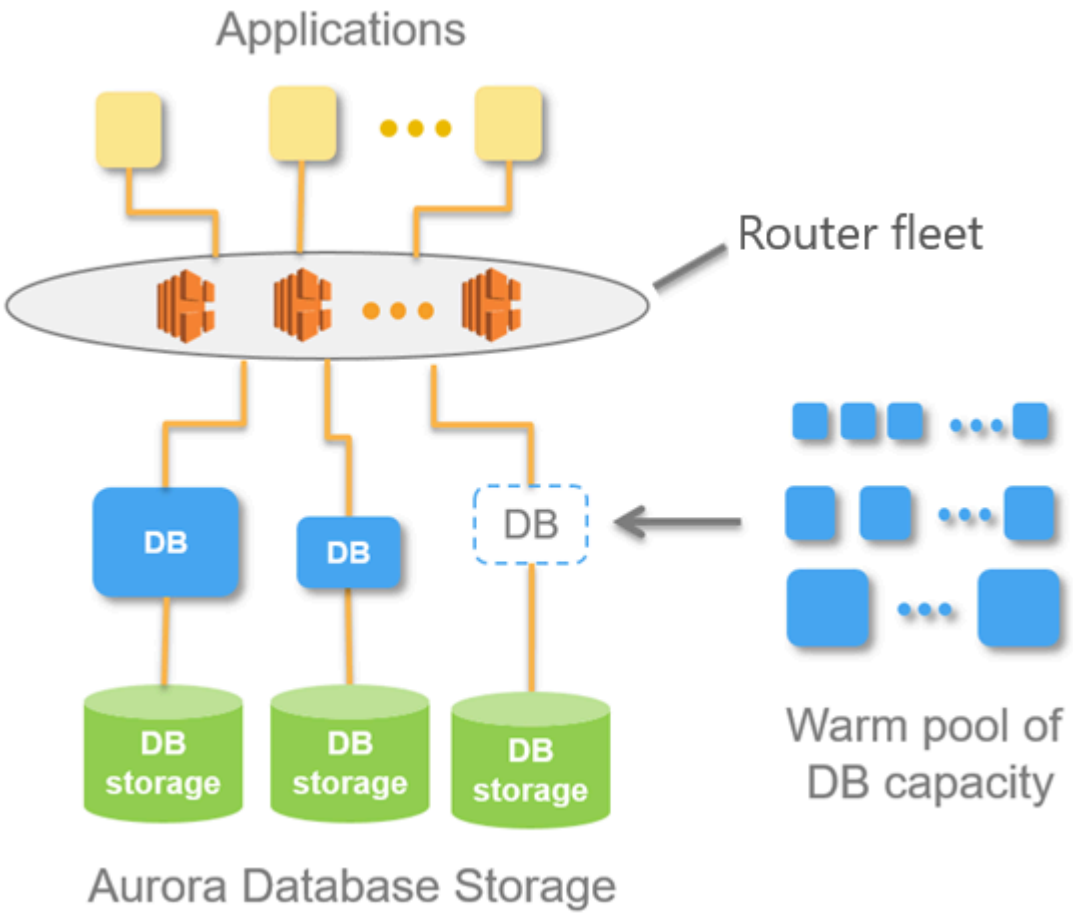
Argomenti

- Aurora Serverless v1 architecture
- Scalabilità automatica per Aurora Serverless v1
- Operazione di timeout per le modifiche di capacità
- Metti in pausa e riprendi per Aurora Serverless v1
- Determinazione del numero massimo di connessioni al database per Aurora Serverless v1
- Gruppi di parametri per Aurora Serverless v1

- Registrazione per Aurora Serverless v1
- Aurora Serverless v1 e manutenzione
- Aurora Serverless v1 e failover
- Aurora Serverless v1 e istantanee

Aurora Serverless v1 architecture

L'immagine seguente mostra una panoramica del Aurora Serverless v1 architettura.



Invece di effettuare il provisioning e gestire i server di database, si specificano le unità ACUs di capacità Aurora (). Ogni ACU è una combinazione di circa 2 gigabyte (GB) di memoria, CPU corrispondente e rete. Lo storage del database esegue automaticamente il dimensionamento da 10 gibibyte (GiB) a 128 tebibytes (TiB), lo stesso storage di un cluster di database Aurora standard.

Puoi specificare l'ACU minima e massima. L'unità di capacità Aurora minima è l'ACU più bassa a cui un cluster di database può essere ridotto. L'unità di capacità Aurora massima è l'ACU più alta a cui un cluster di database può essere aumentato. In base alle tue impostazioni, Aurora Serverless v1

crea automaticamente regole di ridimensionamento per le soglie relative all'utilizzo della CPU, alle connessioni e alla memoria disponibile.

Aurora Serverless v1 gestisce il pool di risorse disponibili in modo da ridurre Regione AWS al minimo i tempi di scalabilità. Quando Aurora Serverless v1 aggiunge nuove risorse al cluster Aurora DB, utilizza la flotta di router per passare le connessioni client attive alle nuove risorse. In qualsiasi momento specifico, ti vengono addebitati solo i costi ACUs che vengono utilizzati attivamente nel tuo cluster Aurora DB.

Scalabilità automatica per Aurora Serverless v1

La capacità assegnata al tuo Aurora Serverless v1 Il cluster DB si ridimensiona facilmente verso l'alto e verso il basso in base al carico generato dall'applicazione client. Qui, il carico è l'utilizzo della CPU e il numero di connessioni. Quando la capacità è limitata da uno di questi fattori, Aurora Serverless v1 aumenta. Aurora Serverless v1 si espande inoltre quando rileva problemi di prestazioni che possono essere risolti in questo modo.

Puoi visualizzare gli eventi di scalabilità per i tuoi Aurora Serverless v1 cluster in. AWS Management Console Durante la scalabilità automatica, Aurora Serverless v1 ripristina la metrica. `EngineUptime` Il valore del valore della metrica di ripristino non significa che il ridimensionamento senza interruzioni abbia avuto problemi o altro Aurora Serverless v1 connessioni interrotte. È semplicemente il punto di partenza per i tempi di attività della nuova capacità. Per maggiori informazioni sui parametri, consulta Monitoraggio dei parametri in un cluster di database Amazon Aurora.

Quando il tuo Aurora Serverless v1 Il cluster DB non ha connessioni attive, può essere ridimensionato fino a una capacità zero (0 ACUs). Per ulteriori informazioni, consulta Metti in pausa e riprendi per Aurora Serverless v1.

Quando è necessario eseguire un'operazione di scalabilità, Aurora Serverless v1 cerca innanzitutto di identificare un punto di scala, un momento in cui non viene elaborata alcuna richiesta. Aurora Serverless v1 potrebbe non essere in grado di trovare un punto di scala per i seguenti motivi:

- Query di lunga durata
- Transazioni in corso
- Tabelle temporanee o blocchi di tabelle

Per aumentare il tuo Aurora Serverless v1 La percentuale di successo del cluster DB nel trovare un punto di scalabilità, ti consigliamo di evitare query e transazioni di lunga durata. Per ulteriori

informazioni sulle operazioni che bloccano la scalabilità e su come evitarle, consulta Best practice for working with Aurora Serverless v1.

Per impostazione predefinita, Aurora Serverless v1 tenta di trovare un punto di scala per 5 minuti (300 secondi). Puoi specificare un periodo di timeout diverso quando crei o modifichi il cluster. Il periodo di timeout può essere compreso tra 60 secondi e 10 minuti (600 secondi). Se Aurora Serverless v1 non riesce a trovare un punto di scala entro il periodo specificato, l'operazione di scalabilità automatica scade.

Per impostazione predefinita, se la scalabilità automatica non trova un punto di scala prima del timeout, Aurora Serverless v1 mantiene il cluster alla capacità attuale. È possibile modificare questo comportamento predefinito quando si crea o si modifica il Aurora Serverless v1 Cluster DB selezionando l'opzione Forza la modifica della capacità. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.

Operazione di timeout per le modifiche di capacità

Se la scalabilità automatica raggiunge il timeout senza trovare un punto di dimensionamento, per impostazione predefinita Aurora mantiene la capacità corrente. Se desideri che Aurora forzi la modifica, abilita l'opzione Force the capacity change (Forza la modifica della capacità). Questa opzione è disponibile nella sezione Autoscaling timeout and action (Timeout e operazione di scalabilità automatica) della pagina Create database (Crea database) quando crei il cluster.

L'opzione Force the capacity change (Forza la modifica della capacità) è disabilitata di default. Mantieni deselezionata questa opzione per avere la tua Aurora Serverless v1 La capacità del cluster DB rimane invariata se l'operazione di scalabilità scade senza trovare un punto di scalabilità.

La selezione di questa opzione causa Aurora Serverless v1 Cluster DB per imporre la modifica della capacità, anche senza un punto di scalabilità. Prima di selezionare questa opzione, devi essere consapevole delle conseguenze che essa può avere:

- Tutte le transazioni in corso vengono interrotte e viene visualizzato il seguente messaggio di errore.

  Aurora MySQL versione 2 - ERRORE 1105 (HY000): L'ultima transazione è stata interrotta a causa del dimensionamento continuo. Riprova.

  Puoi inviare nuovamente le transazioni non appena Aurora Serverless v1 Il cluster DB è disponibile.
- Le connessioni a tabelle temporanee e ai blocchi vengono eliminate.

Ti consigliamo di selezionare Force the capacity change (Forza la modifica della capacità) solo se è possibile recuperare l'applicazione in seguito a connessioni interrotte o transazioni incomplete.

Le scelte che fai AWS Management Console quando crei un Aurora Serverless v1 I cluster DB sono archiviati nell'`ScalingConfigurationInfo`oggetto, nelle `TimeoutAction` proprietà `SecondsBeforeTimeout` and. Quando si crea il cluster, il valore della proprietà `TimeoutAction` viene impostato su uno dei seguenti valori:

- `RollbackCapacityChange`: questo valore viene impostato quando selezioni l'opzione Roll back the capacity change (Eseguire il rollback della modifica della capacità). Questo è il comportamento che segue di default.
- `ForceApplyCapacityChange`: questo valore viene impostato quando selezioni l'opzione Force the capacity change (Forza la modifica della capacità).

È possibile ottenere il valore di questa proprietà su una proprietà esistente Aurora Serverless v1 Cluster DB utilizzando il describe-db-clusters AWS CLI comando, come illustrato di seguito.

In Linux, macOS, oppure Unix:

```
aws rds describe-db-clusters --region region \
  --db-cluster-identifier your-cluster-name \
  --query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'
```

In Windows:

```
aws rds describe-db-clusters --region region ^
  --db-cluster-identifier your-cluster-name ^
  --query "*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}"
```

A titolo di esempio, di seguito vengono illustrate la query e la risposta per un Aurora Serverless v1 Cluster DB denominato `west-coast-sles` nella regione Stati Uniti occidentali (California settentrionale).

```
$ aws rds describe-db-clusters --region us-west-1 --db-cluster-identifier west-coast-
sles
--query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'
```

```
[
    {
        "ScalingConfigurationInfo": {
            "MinCapacity": 1,
            "MaxCapacity": 64,
            "AutoPause": false,
            "SecondsBeforeTimeout": 300,
            "SecondsUntilAutoPause": 300,
            "TimeoutAction": "RollbackCapacityChange"
        }
    }
]
```

Come mostra la risposta, questo Aurora Serverless v1 Il cluster DB utilizza l'impostazione predefinita.

Per ulteriori informazioni, consulta Creare un Aurora Serverless v1 cluster di database. Dopo aver creato il tuo Aurora Serverless v1, puoi modificare l'azione di timeout e altre impostazioni di capacità in qualsiasi momento. Per scoprire come, consulta Modificare un Aurora Serverless v1 cluster di database.

Metti in pausa e riprendi per Aurora Serverless v1

Puoi scegliere di mettere in pausa il tuo Aurora Serverless v1 Cluster DB dopo un determinato periodo di tempo senza attività. Specifica il periodo di tempo in cui non viene registrata alcuna attività prima che il cluster di database venga messo in pausa. Quando si seleziona questa opzione, il tempo di inattività predefinito è di cinque minuti, ma è possibile modificare questo valore. Questa è un'impostazione facoltativa.

Quando il cluster di database viene messo in pausa, non avvengono attività di elaborazione o memoria e vengono addebitati soltanto i costi per lo storage. Se vengono richieste connessioni al database quando un Aurora Serverless v1 Il cluster DB è in pausa, il cluster DB riprende automaticamente e soddisfa le richieste di connessione.

Quando il cluster di database riprende l'attività, ha la stessa capacità che aveva quando Aurora ha messo in pausa il cluster. Il numero di ACUs dipende da quanto Aurora ha aumentato o ridotto il cluster prima di metterlo in pausa.

> **Note**
>
> Se un cluster di database viene messo in pausa per oltre sette giorni potrebbe essere sottoposto a backup con uno snapshot. In questo caso, Aurora ripristina il cluster di database dalla snapshot quando viene avanzata una richiesta di connessione.

Determinazione del numero massimo di connessioni al database per Aurora Serverless v1

I seguenti esempi riguardano un Aurora Serverless v1 Cluster DB compatibile con MySQL 5.7. Puoi utilizzare un client MySQL o l'editor di query, se è stato configurato l'accesso. Per ulteriori informazioni, consulta Esecuzione di query nell'editor della query.

Trovare il numero massimo di connessioni al database

1. Trova la gamma di capacità adatta al tuo Aurora Serverless v1 Cluster DB che utilizza il AWS CLI.

   ```
   aws rds describe-db-clusters \
       --db-cluster-identifier my-serverless-57-cluster \
       --query 'DBClusters[*].ScalingConfigurationInfo|[0]'
   ```

   Il risultato mostra che il suo intervallo di capacità è compreso tra 1 ACUs e 4.

   ```
   {
       "MinCapacity": 1,
       "AutoPause": true,
       "MaxCapacity": 4,
       "TimeoutAction": "RollbackCapacityChange",
       "SecondsUntilAutoPause": 3600
   }
   ```

2. Esegui la seguente query SQL per trovare il numero massimo di connessioni.

   ```
   select @@max_connections;
   ```

   Il risultato mostrato è per la capacità minima del cluster, 1 ACU.

```
@@max_connections
90
```

3. Ridimensiona il cluster a 8—32. ACUs

   Per ulteriori informazioni sul dimensionamento, consultare Modificare un Aurora Serverless v1 cluster di database.

4. Conferma l'intervallo di capacità.

```
{
    "MinCapacity": 8,
    "AutoPause": true,
    "MaxCapacity": 32,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

5. Trova il numero massimo di connessioni.

```
select @@max_connections;
```

   Il risultato mostrato si riferisce alla capacità minima del cluster, 8. ACUs

```
@@max_connections
1000
```

6. Ridimensiona il cluster fino al massimo possibile, 256—256 ACUs.
7. Conferma l'intervallo di capacità.

```
{
    "MinCapacity": 256,
    "AutoPause": true,
    "MaxCapacity": 256,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

8. Trova il numero massimo di connessioni.

```
select @@max_connections;
```

Il risultato mostrato è per 256. ACUs

```
@@max_connections
6000
```

> **Note**
>
> Il `max_connections` valore non viene scalato linearmente con il numero di ACUs.

9. Ridimensiona il cluster fino a ACUs 1—4.

```
{
    "MinCapacity": 1,
    "AutoPause": true,
    "MaxCapacity": 4,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

Questa volta, il `max_connections` valore è 4. ACUs

```
@@max_connections
270
```

10. Riduci il cluster a 2 ACUs.

```
@@max_connections
180
```

Se hai configurato il cluster in modo che si interrompa dopo un certo periodo di inattività, il cluster viene ridimensionato a 0. ACUs Tuttavia, `max_connections` non scende sotto il valore per 1 ACU.

```
@@max_connections
90
```

Gruppi di parametri per Aurora Serverless v1

Quando crei il tuo Aurora Serverless v1 Cluster DB, scegli uno specifico motore Aurora DB e un gruppo di parametri del cluster DB associato. A differenza dei cluster Aurora DB forniti, un Aurora Serverless v1 Il cluster DB dispone di una singola istanza DB di lettura/scrittura configurata solo con un gruppo di parametri del cluster DB e non dispone di un gruppo di parametri DB separato. Durante la scalabilità automatica, Aurora Serverless v1 deve essere in grado di modificare i parametri affinché il cluster funzioni al meglio in caso di aumento o diminuzione della capacità. Quindi, con un Aurora Serverless v1 Cluster DB, alcune delle modifiche che potresti apportare ai parametri per un particolare tipo di motore DB potrebbero non essere applicabili.

Ad esempio, una versione Aurora basata su PostgreSQL Aurora Serverless v1 Il cluster DB non può utilizzare `apg_plan_mgmt.capture_plan_baselines` e altri parametri che potrebbero essere utilizzati sui cluster Aurora PostgreSQL DB forniti per la gestione dei piani di query.

È possibile ottenere un elenco di valori predefiniti per i gruppi di parametri predefiniti per i vari motori Aurora DB utilizzando il comando CLI describe-engine-default-cluster-parameters e interrogando il. Regione AWS Di seguito sono riportati i valori che è possibile utilizzare per l'opzione `--db-parameter-group-family`.

Aurora MySQL versione 2	`aurora-mysql5.7`
Aurora PostgreSQL versione 11	`aurora-postgresql11`
Aurora PostgreSQL versione 13	`aurora-postgresql13`

Ti consigliamo di configurarlo AWS CLI con l'ID della chiave di AWS accesso e la chiave di accesso AWS segreta e di impostarlo prima di utilizzare i Regione AWS comandi. AWS CLI Fornire la regione alla configurazione CLI consente di evitare di immettere il parametro `--region` l'esecuzione dei comandi. Per ulteriori informazioni sulla configurazione AWS CLI, consulta Nozioni di base sulla configurazione nella Guida per l'AWS Command Line Interface utente.

Nell'esempio seguente si recupera un elenco di parametri dal gruppo di cluster di database predefinito per Aurora MySQL versione 2.

In Linux, macOS, oppure Unix:

```
aws rds describe-engine-default-cluster-parameters \
```

```
  --db-parameter-group-family aurora-mysql5.7 --query \
  'EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, `serverless`) == `true`] | [*].{param:ParameterName}' \
  --output text
```

In Windows:

```
aws rds describe-engine-default-cluster-parameters ^
   --db-parameter-group-family aurora-mysql5.7 --query ^
   "EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, 'serverless') == `true`] | [*].{param:ParameterName}" ^
   --output text
```

Modifica dei valori dei parametri per Aurora Serverless v1

Come spiegato in , non è possibile modificare direttamente i valori in un gruppo di parametri predefinito, indipendentemente dal tipo (gruppo di parametri del cluster di database, gruppo di parametri DB). Al contrario, si crea un gruppo di parametri personalizzato basato sul gruppo di parametri cluster di database predefinito per il motore database Aurora e modificare le impostazioni in base alle esigenze su quel gruppo di parametri. Ad esempio, potresti voler modificare alcune impostazioni del Aurora Serverless v1 Cluster DB per registrare le query o caricare log specifici del motore DB su Amazon. CloudWatch

Per creare un gruppo di parametri del cluster di database personalizzato

1. Accedi a AWS Management Console e quindi apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegliere Gruppi di parametri.
3. Seleziona Crea gruppo di parametri per aprire il riquadro dei dettagli del gruppo di parametri.
4. Scegli il gruppo di cluster DB predefinito appropriato per il motore DB che desideri utilizzare per Aurora Serverless v1 Cluster DB. Assicurati di scegliere le seguenti opzioni:

    a. Per Famiglia gruppo parametri, seleziona la famiglia appropriata per il motore del database scelto. Assicurati che la selezione abbia un nome contenente il prefisso `aurora-`.

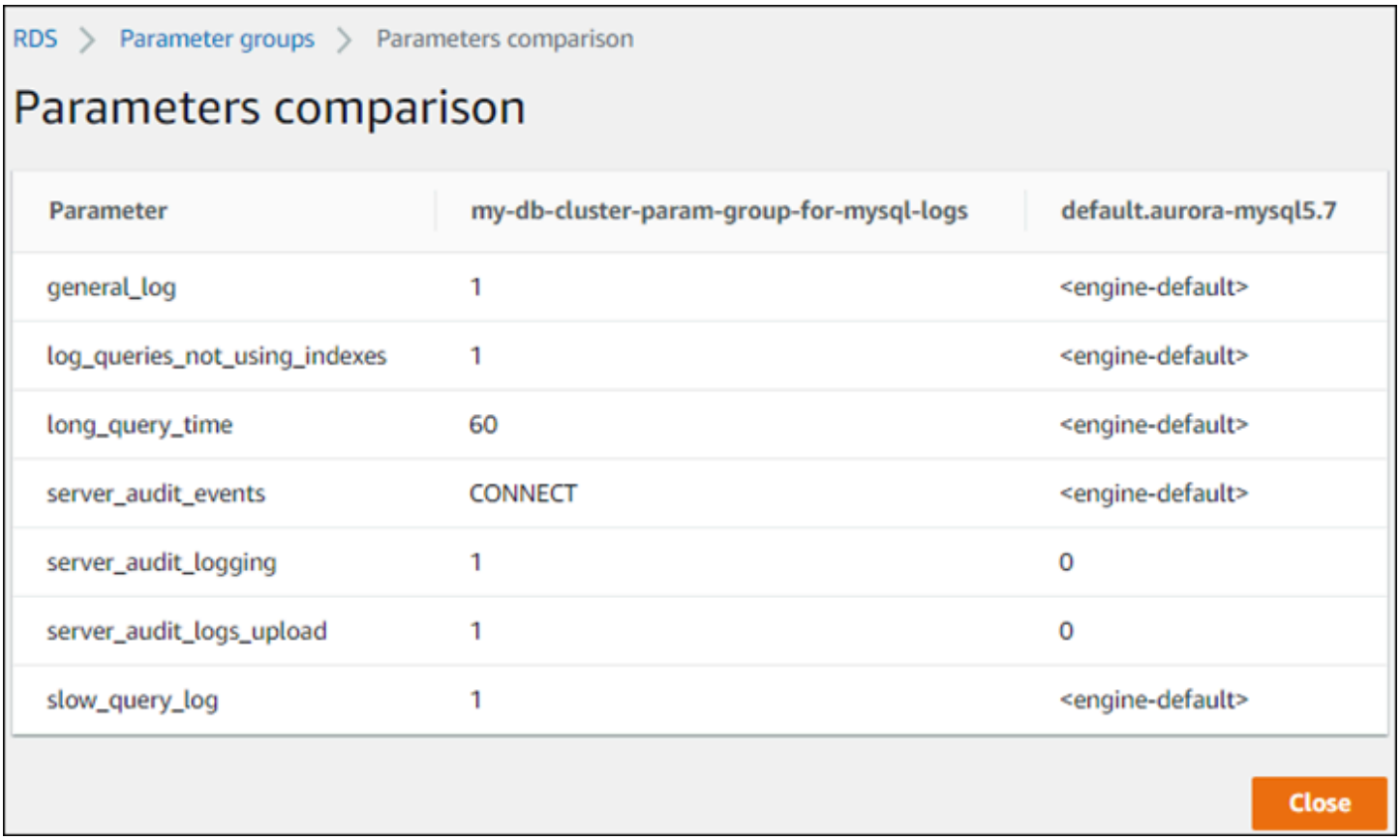
    b. Per Type (Tipo), scegli DB Cluster Parameter Group (Gruppo di parametri del cluster di database).

c. Per Nome e descrizione del gruppo, inserisci nomi significativi per te o per altre persone che potrebbero aver bisogno di lavorare con Aurora Serverless v1 Cluster DB e relativi parametri.

d. Scegli Create (Crea).

Il gruppo di parametri del cluster di database personalizzato viene aggiunto all'elenco dei gruppi di parametri disponibili per l'account nella Regione AWS. È possibile utilizzare il gruppo di parametri del cluster DB personalizzato quando si crea un nuovo Aurora Serverless v1 Cluster DB. È inoltre possibile modificare un file esistente Aurora Serverless v1 Cluster DB per utilizzare il gruppo di parametri del cluster DB personalizzato. Dopo il tuo Aurora Serverless v1 Il cluster DB inizia a utilizzare il gruppo di parametri del cluster DB personalizzato, è possibile modificare i valori per i parametri dinamici utilizzando il AWS Management Console o il AWS CLI.

È inoltre possibile utilizzare la console per visualizzare un side-by-side confronto tra i valori nel gruppo di parametri del cluster DB personalizzato rispetto al gruppo di parametri del cluster DB predefinito, come mostrato nella schermata seguente.



Quando si modificano i valori dei parametri su un cluster DB attivo, Aurora Serverless v1 avvia una scala senza interruzioni per applicare le modifiche ai parametri. Se le ricette di Aurora Serverless v1 Il cluster DB è in uno stato di pausa, riprende e inizia a scalare in modo da poter apportare la modifica. L'operazione di dimensionamento per la modifica di un gruppo di parametri forza sempre la modifica della capacità, quindi tieni presente che la modifica dei parametri potrebbe comportare la perdita di connessioni se non è possibile trovare un punto di dimensionamento durante il periodo di dimensionamento.

Registrazione per Aurora Serverless v1

Per impostazione predefinita, i registri degli errori per Aurora Serverless v1 sono abilitati e caricati automaticamente su Amazon CloudWatch. Puoi anche avere il tuo Aurora Serverless v1 Il cluster DB carica i log specifici del motore di database Aurora su. CloudWatch Per fare ciò, abilita i parametri di configurazione nel gruppo di parametri del cluster di database personalizzati. Il tuo Aurora Serverless v1 Il cluster DB carica quindi tutti i log disponibili su Amazon. CloudWatch A questo punto, puoi utilizzarli CloudWatch per analizzare i dati di registro, creare allarmi e visualizzare le metriche.

Per Aurora MySQL, la tabella seguente mostra i log che è possibile abilitare. Se abilitati, vengono caricati automaticamente dal Aurora Serverless v1 Cluster DB su Amazon CloudWatch.

Log Aurora MySQL	Descrizione
`general_log`	Crea il log generale. Imposta su 1 per attivare questa opzione. Il valore predefinito è disattiva to (0).
`log_queries_not_using_indexes`	Registra tutte le query nel log delle query lente che non utilizzano un indice. Il valore predefini to è disattivato (0). Imposta su 1 per attivare questo log.
`long_query_time`	Impedisce che le query in esecuzione rapida vengano registrate nel log delle query lente. Può essere impostato su un valore variabile compreso tra 0 e 31.536.000. Il valore predefini to è 0 (non attivo).
`server_audit_events`	L'elenco degli eventi da catturare nei log. I valori supportati sono `CONNECT`, `QUERY`,

Log Aurora MySQL	Descrizione
	`QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` e `TABLE`.
`server_audit_logging`	Imposta su 1 per attivare la registrazione di controllo del server. Se attivi questa opzione, puoi specificare gli eventi di controllo a cui inviare CloudWatch elencandoli nel `server_audit_events` parametro.
`slow_query_log`	Crea un log di query lente. Imposta su 1 per attivare il log di query lente. Il valore predefinito è disattivato (0).

Per ulteriori informazioni, consulta Utilizzo del controllo avanzato con un cluster Amazon Aurora My DB SQL.

Per Aurora PostgreSQL, la tabella seguente mostra i log che è possibile abilitare. Se abilitati, vengono caricati automaticamente dal Aurora Serverless v1 Cluster DB su Amazon CloudWatch insieme ai normali log degli errori.

Log Aurora PostgreSQL	Descrizione
`log_connections`	Attivato di default e non può essere modificato. Registra i dettagli per tutte le nuove connessio ni client.
`log_disconnections`	Attivato di default e non può essere modificato. Registra tutte le disconnessioni client.
`log_hostname`	È disattivata per impostazione predefinita e non può essere modificata. I nomi host non vengono registrati.
`log_lock_waits`	Il valore predefinito è 0 (disattivato). Imposta su 1 per registrare le attese di blocco.

Log Aurora PostgreSQL	Descrizione
`log_min_duration_statement`	La durata minima (in millisecondi) per l'esecuzione di un'istruzione prima della registrazione.
`log_min_messages`	Imposta i livelli dei messaggi che vengono registrati. I valori supportati sono `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Per registrare i dati delle prestazioni nel log `postgres`, imposta il valore su `debug1`.
`log_temp_files`	Registra l'utilizzo di file temporanei che si trovano al di sopra dei kilobyte (kB) specificati.
`log_statement`	Controlla le istruzioni SQL specifiche che vengono registrate. I valori supportati sono `none`, `ddl`, `mod` e `all`. Il valore predefinito è `none`.

Dopo aver attivato i log per Aurora MySQL o Aurora PostgreSQL per il Aurora Serverless v1 Cluster DB, puoi visualizzare i log in. CloudWatch

Visualizzazione Aurora Serverless v1 registri con Amazon CloudWatch

Aurora Serverless v1 carica automaticamente («pubblica») su Amazon CloudWatch tutti i log abilitati nel gruppo di parametri del cluster DB personalizzato. Non è necessario scegliere o specificare i tipi di log. Il caricamento dei log inizia non appena si attiva il parametro di configurazione log. Se in seguito questo parametro viene disattivato, gli altri caricamenti saranno interrotti. Tuttavia, tutti i log che sono già stati pubblicati CloudWatch rimarranno finché non li elimini.

Per ulteriori informazioni sull'utilizzo CloudWatch con i log MySQL di Aurora, consulta. Monitoraggio degli eventi di registro in Amazon CloudWatch

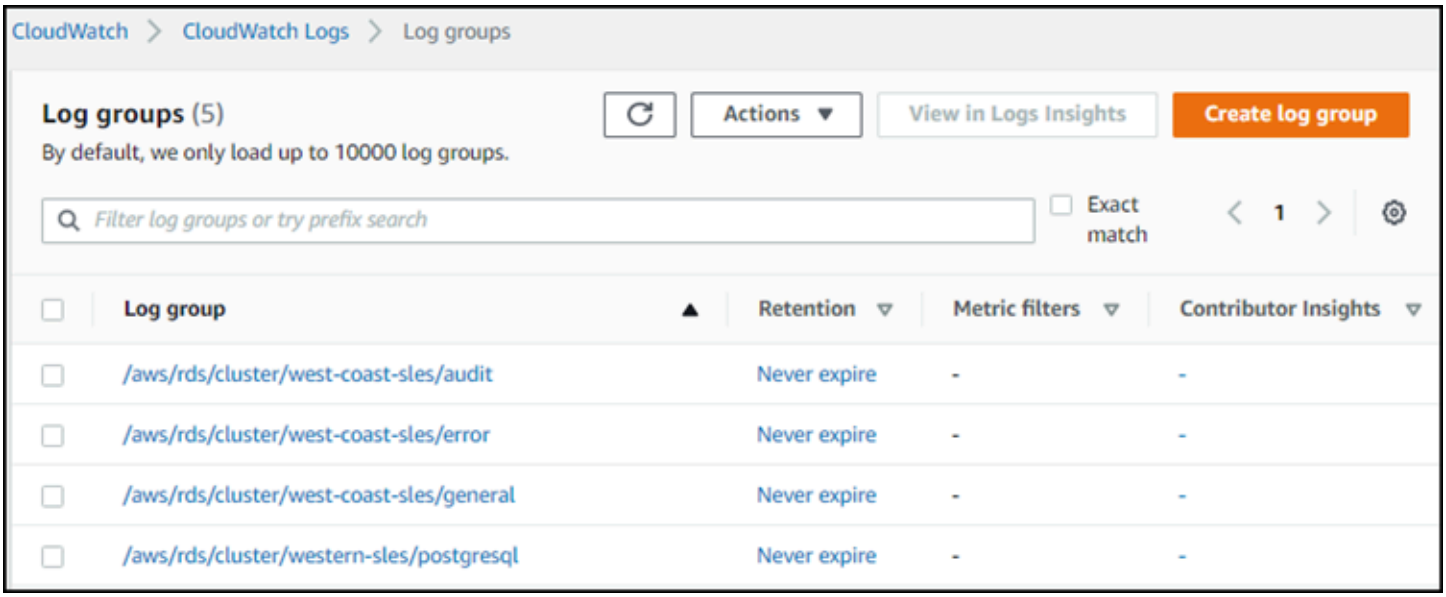
Per ulteriori informazioni su CloudWatch Aurora PostgreSQL, vedere. Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs CloudWatch

Per visualizzare i log relativi al Aurora Serverless v1 cluster di database

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli il tuo Regione AWS.
3. Scegli Log groups (Gruppi di log).
4. Scegli il tuo Aurora Serverless v1 Registro del cluster DB dall'elenco. Per i log di errori, il modello di denominazione è il seguente.

```
/aws/rds/cluster/cluster-name/error
```

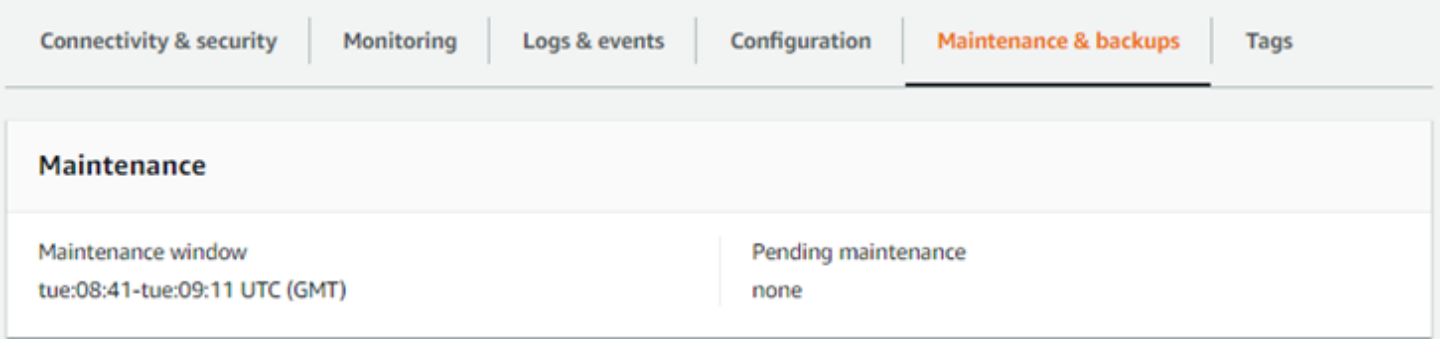
Ad esempio, nella schermata seguente è possibile trovare elenchi di log pubblicati per Aurora PostgreSQL Aurora Serverless v1 Cluster DB denominato. `western-sles` Puoi anche trovare diversi elenchi per Aurora MySQL Aurora Serverless v1 Cluster DB,. `west-coast-sles` Seleziona il log desiderato per iniziare ad esplorarne il contenuto.



Aurora Serverless v1 e manutenzione

Manutenzione per Aurora Serverless v1 Il cluster DB, ad esempio l'applicazione delle funzionalità, delle correzioni e degli aggiornamenti di sicurezza più recenti, viene eseguito automaticamente per te. Aurora Serverless v1 dispone di una finestra di manutenzione che puoi visualizzare AWS Management Console nella sezione Manutenzione e backup per il tuo Aurora Serverless v1 Cluster DB. Puoi trovare la data e l'ora in cui potrebbe essere eseguita la manutenzione e se ci sono

interventi di manutenzione in sospeso per il tuo Aurora Serverless v1 Cluster DB, come illustrato nella figura seguente.



È possibile impostare la finestra di manutenzione quando si crea il Aurora Serverless v1 Cluster DB e puoi modificare la finestra in un secondo momento. Per ulteriori informazioni, consulta Impostazione della finestra di manutenzione preferita del cluster database.

Le finestre di manutenzione vengono utilizzate per gli aggiornamenti programmati delle versioni principali. Gli aggiornamenti e le patch delle versioni minori vengono applicati immediatamente durante il ridimensionamento. Il ridimensionamento avviene in base alle impostazioni dell'utente per: `TimeoutAction`

- `ForceApplyCapacityChange`— La modifica viene applicata immediatamente.
- `RollbackCapacityChange`— Aurora aggiorna forzatamente il cluster dopo 3 giorni dal primo tentativo di patch.

Come per qualsiasi modifica forzata senza un punto di ridimensionamento appropriato, ciò potrebbe interrompere il carico di lavoro.

Quando possibile, Aurora Serverless v1 esegue la manutenzione senza interruzioni. Quando è necessaria la manutenzione, il Aurora Serverless v1 Il cluster DB ridimensiona la propria capacità per gestire le operazioni necessarie. Prima della scalabilità, Aurora Serverless v1 cerca un punto di scala. Lo fa per un massimo di tre giorni, se necessario.

Alla fine di ogni giornata quello Aurora Serverless v1 non riesce a trovare un punto di scalabilità, crea un evento di cluster. Questo evento notifica la manutenzione in sospeso e la necessità di eseguire il dimensionamento per eseguire la manutenzione. La notifica include la data in cui Aurora Serverless v1 può forzare la scalabilità del cluster DB.

Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.

Aurora Serverless v1 e failover

Se l'istanza DB per un Aurora Serverless v1 Il cluster DB diventa non disponibile o la zona di disponibilità (AZ) in cui si trova fallisce, Aurora ricrea l'istanza DB in una zona di disponibilità diversa. Tuttavia, Aurora Serverless v1 il cluster non è un cluster Multi-AZ. Questo perché è costituito da una singola istanza database in un'unica zona di disponibiità. Pertanto, questo meccanismo di failover richiede più tempo rispetto a un cluster Aurora con provisioning o Aurora Serverless v2 istanze.
Il Aurora Serverless v1 il tempo di failover non è definito perché dipende dalla domanda e dalla disponibilità di capacità in altri paesi AZs all'interno del dato dato. Regione AWS

Poiché Aurora separa la capacità di calcolo e lo storage, il volume di archiviazione per il cluster è distribuito su più livelli. AZs I dati rimangono disponibili anche se un'interruzione riguarda l'istanza database o la zona di disponibilità associata.

Aurora Serverless v1 e istantanee

Il volume del cluster per un Aurora Serverless v1 il cluster è sempre crittografato. Puoi scegliere la chiave di crittografia, ma non puoi disabilitare la crittografia. Per copiare o condividere un'istantanea di un Aurora Serverless v1 cluster, crittografa l'istantanea usando la tua. AWS KMS key Per ulteriori informazioni, consulta Copia di snapshot del cluster DB. Per ulteriori informazioni sulla crittografia e Amazon Aurora, consulta Creazione di un cluster di database Amazon Aurora

Creare un Aurora Serverless v1 cluster di database

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

La procedura seguente crea un Aurora Serverless v1 cluster senza oggetti o dati dello schema. Se vuoi creare un Aurora Serverless v1 cluster che è un duplicato di un cluster predisposto esistente

o Aurora Serverless v1 cluster, è possibile invece eseguire un'operazione di ripristino o clonazione delle istantanee. Per ottenere queste informazioni, consulta Ripristino da uno snapshot cluster database e Clonazione di un volume per un cluster di database Amazon Aurora. Non è possibile convertire un cluster predisposto esistente in Aurora Serverless v1. Inoltre, non puoi convertire un file esistente Aurora Serverless v1 riconvertire il cluster in un cluster predisposto.

Quando si crea un Aurora Serverless v1 Cluster DB, è possibile impostare la capacità minima e massima per il cluster. Una unità di capacità equivale a una specifica configurazione di memoria e calcolo. Aurora Serverless v1 crea regole di scalabilità per le soglie relative all'utilizzo della CPU, alle connessioni e alla memoria disponibile e si adatta perfettamente a una gamma di unità di capacità in base alle esigenze delle applicazioni. Per ulteriori informazioni, consulta Aurora Serverless v1 architecture.

È possibile impostare i seguenti valori specifici per Aurora Serverless v1 Cluster DB:

- Unità con capacità minima Aurora — Aurora Serverless v1 può ridurre la capacità fino a questa unità di capacità.
- Unità con capacità massima Aurora — Aurora Serverless v1 può aumentare la capacità fino a questa unità di capacità.

È inoltre possibile scegliere le seguenti opzioni di configurazione del dimensionamento facoltative:

- Ridimensionamento forzato della capacità ai valori specificati quando viene raggiunto il timeout: puoi scegliere questa impostazione se lo desideri Aurora Serverless v1 forzare Aurora Serverless v1 da scalare anche se non riesce a trovare un punto di scala prima che scada. Se vuoi Aurora Serverless v1 per annullare le modifiche alla capacità se non riesce a trovare un punto di scala, non scegliere questa impostazione. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.
- Metti in pausa la capacità di elaborazione dopo minuti consecutivi di inattività: puoi scegliere questa impostazione se lo desideri Aurora Serverless v1 per scalare a zero quando non vi è alcuna attività sul cluster DB per un periodo di tempo specificato. Con questa impostazione abilitata, Aurora Serverless v1 Il cluster DB riprende automaticamente l'elaborazione e si adatta alla capacità necessaria per gestire il carico di lavoro quando riprende il traffico del database. Per ulteriori informazioni, consulta Metti in pausa e riprendi per Aurora Serverless v1.

Prima di poter creare un Aurora Serverless v1 Cluster DB, è necessario un AWS account. È inoltre necessario aver completato le attività di configurazione per lavorare con Amazon Aurora. Per

ulteriori informazioni, consulta Configurazione dell'ambiente per Amazon Aurora. È inoltre necessario completare altre fasi preliminari per la creazione di un cluster di database Aurora. Per ulteriori informazioni, vedi Creazione di un cluster database Amazon Aurora.

Aurora Serverless v1 è disponibile solo in alcune versioni di Aurora MySQL Regioni AWS e Aurora PostgreSQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta Aurora Serverless v1.

> **Note**
>
> Il volume del cluster per un Aurora Serverless v1 il cluster è sempre crittografato. Quando crei il tuo Aurora Serverless v1 Cluster DB, non puoi disattivare la crittografia, ma puoi scegliere di utilizzare la tua chiave di crittografia. Con Aurora Serverless v2, è possibile scegliere se crittografare il volume del cluster.

È possibile creare un Aurora Serverless v1 Cluster DB con AWS CLI o l'API RDS.

> **Note**
>
> Se viene visualizzato il seguente messaggio di errore quando si tenta di creare il cluster, l'account necessita di autorizzazioni aggiuntive.
> `Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.`
> Per ulteriori informazioni, consulta Utilizzo di ruoli collegati ai servizi per Amazon Aurora.

Non puoi connetterti direttamente all'istanza DB sul tuo Aurora Serverless v1 Cluster DB. Per connetterti al tuo Aurora Serverless v1 Cluster DB, si utilizza l'endpoint del database. Puoi trovare l'endpoint per il tuo Aurora Serverless v1 Cluster DB nella scheda Connettività e sicurezza relativa al cluster in. AWS Management Console Per ulteriori informazioni, consulta Connessione a un cluster database Amazon Aurora.

AWS CLI

Per creare un nuovo Aurora Serverless v1 Cluster DB con AWS CLI, esegui il create-db-clustercomando e specifica `serverless` l'`--engine-mode`opzione.

Puoi eventualmente specificare l'opzione `--scaling-configuration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni.

I seguenti esempi di comando creano un nuovo cluster di database Serverless impostando l'opzione `--engine-mode` su `serverless`. Gli esempi specificano inoltre i valori per l'opzione `--scaling-configuration`.

Esempio per Aurora MySQL

Il seguente comando creano un nuovo cluster database serverless compatibile con Aurora MySQL. I valori di capacità validi per Aurora MySQL sono 1, 2, 4, 8, 16, 32, 64, 128 e 256.

In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

In Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

Esempio per Aurora PostgreSQL

Il comando seguente crea un nuovo cluster di database serverless compatibile con PostgreSQL 13.9. I valori di capacità validi per Aurora PostgreSQL sono 2, 4, 8, 16, 32, 64, 192 e 384.

In Linux, macOS, oppure Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-postgresql --engine-version 13.9 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

In Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-postgresql --engine-version 13.9 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

API RDS

Per crearne uno nuovo Aurora Serverless v1 Cluster DB con l'API RDS, esegui l'DBClusteroperazione Create e specifica `serverless` il `EngineMode` parametro.

Puoi eventualmente specificare il parametro `ScalingConfiguration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Ripristino di un Aurora Serverless v1 cluster di database

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster non migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Puoi configurare un Aurora Serverless v1 Cluster DB quando si ripristina uno snapshot del cluster DB di cui è stato effettuato il provisioning con l'API AWS CLI o RDS.

Quando si ripristina un'istantanea su un Aurora Serverless v1 Cluster DB, è possibile impostare i seguenti valori specifici:

- Unità con capacità minima Aurora — Aurora Serverless v1 può ridurre la capacità fino a questa unità di capacità.
- Unità con capacità massima Aurora — Aurora Serverless v1 può aumentare la capacità fino a questa unità di capacità.
- Azione di timeout – L'azione da eseguire quando una modifica della capacità scade perché non riesce a trovare un punto di dimensionamento. Aurora Serverless v1 Il cluster database può forzare il cluster database alle nuove impostazioni di capacità impostando l'opzione Forza la capacità di dimensionamento ai valori specificati.... In alternativa, è possibile ripristinare la modifica della capacità per annullarla se non si sceglie l'opzione. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.
- Pause after inactivity (Pausa dopo inattività): periodo di tempo senza traffico di database trascorso il quale la capacità di calcolo viene ridotta a zero. Quando il traffico di database riprende, Aurora aumenta automaticamente la capacità di calcolo e si dimensiona per gestire il traffico.

Per informazioni generali sul ripristino di un cluster database da uno snapshot, consulta Ripristino da uno snapshot cluster database.

AWS CLI

È possibile configurare un Aurora Serverless Cluster DB quando si ripristina uno snapshot del cluster DB di cui è stato eseguito il provisioning con AWS Management Console AWS CLI, the o l'API RDS.

Quando si ripristina un'istantanea su un Aurora Serverless Cluster DB, è possibile impostare i seguenti valori specifici:

- Unità con capacità minima Aurora — Aurora Serverless può ridurre la capacità fino a questa unità di capacità.
- Unità con capacità massima Aurora — Aurora Serverless può aumentare la capacità fino a questa unità di capacità.
- Azione di timeout – L'azione da eseguire quando una modifica della capacità scade perché non riesce a trovare un punto di dimensionamento. Aurora Serverless v1 Il cluster database può forzare il cluster database alle nuove impostazioni di capacità impostando l'opzione Forza la capacità di dimensionamento ai valori specificati.... In alternativa, è possibile ripristinare la modifica della capacità per annullarla se non si sceglie l'opzione. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.

- Pause after inactivity (Pausa dopo inattività): periodo di tempo senza traffico di database trascorso il quale la capacità di calcolo viene ridotta a zero. Quando il traffico di database riprende, Aurora aumenta automaticamente la capacità di calcolo e si dimensiona per gestire il traffico.

> **Note**
>
> La versione dello snapshot del cluster DB deve essere compatibile con Aurora Serverless v1. Per l'elenco delle versioni supportate, vedereAurora Serverless v1.

Per ripristinare un'istantanea su un Aurora Serverless v1 cluster con compatibilità MySQL 5.7, include i seguenti parametri aggiuntivi:

- `--engine` **`aurora-mysql`**
- `--engine-version` **`5.7`**

I `--engine-version` parametri `--engine` and consentono di creare un file compatibile con MySQL 5.7 Aurora Serverless v1 cluster da Aurora compatibile con MySQL 5.6 o Aurora Serverless v1 istantanea. L'esempio seguente ripristina un'istantanea da un cluster compatibile con MySQL 5.6 denominato a un cluster compatibile con MySQL 5.7 *`mydbclustersnapshot`* Aurora Serverless v1 cluster denominato. *`mynewdbcluster`*

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless \
    --engine aurora-mysql \
    --engine-version 5.7
```

In Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql ^
    --engine-version 5.7
```

Puoi eventualmente specificare l'opzione `--scaling-configuration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Nell'esempio seguente, si esegue il ripristino da uno snapshot del cluster DB creato in precedenza denominato *`mydbclustersnapshot`* a un nuovo cluster DB denominato*`mynewdbcluster`*. Si imposta il `--scaling-configuration` in modo che il nuovo Aurora Serverless v1 Il cluster DB può scalare da 8 ACUs a 64 ACUs (unità di capacità Aurora) in base alle esigenze per elaborare il carico di lavoro. Al termine dell'elaborazione e dopo 1000 secondi senza connessioni da supportare, il cluster si arresta fino a quando le richieste di connessione non richiedono il riavvio.

In Linux, macOS, oppure Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

In Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

API RDS

Per configurare un Aurora Serverless v1 Cluster DB quando esegui il ripristino da un cluster DB utilizzando l'API RDS, esegui l'DBClusterFromSnapshotoperazione di ripristino e specifica `serverless` il `EngineMode` parametro.

Puoi eventualmente specificare il parametro `ScalingConfiguration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Modificare un Aurora Serverless v1 cluster di database

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Dopo aver configurato un Aurora Serverless v1 Il cluster DB, è possibile modificare determinate proprietà con AWS Management Console AWS CLI, the o l'API RDS. La maggior parte delle proprietà che puoi modificare sono le stesse di altri tipi di cluster Aurora.

Le modifiche più rilevanti per Aurora Serverless v1 sono le seguenti:

- Modifica della configurazione di dimensionamento
- Aggiornamento della versione principale
- Conversione da Aurora Serverless v1 a fornito

Modifica della configurazione di ridimensionamento di un Aurora Serverless v1 cluster di database

Puoi impostare le capacità minima e massima per il cluster di database. Ogni unità di capacità equivale a una specifica configurazione di memoria e calcolo. Aurora Serverless crea automaticamente regole di ridimensionamento per le soglie relative all'utilizzo della CPU, alle connessioni e alla memoria disponibile. È inoltre possibile impostare se Aurora Serverless mette in pausa il database quando non c'è alcuna attività e poi riprende quando l'attività ricomincia.

Puoi impostare i seguenti valori specifici per la configurazione di dimensionamento:

- Unità con capacità minima Aurora — Aurora Serverless può ridurre la capacità fino a questa unità di capacità.
- Unità con capacità massima Aurora — Aurora Serverless può aumentare la capacità fino a questa unità di capacità.
- Timeout e azione con scalabilità automatica: questa sezione specifica per quanto tempo Aurora Serverless attende di trovare un punto di scala prima del timeout. Specifica anche l'operazione da eseguire quando una modifica della capacità raggiunge il timeout perché non riesce a trovare un punto di dimensionamento. Aurora può forzare la modifica della capacità per impostare la capacità sul valore specificato non appena possibile. Oppure, può eseguire il rollback della modifica della capacità per annullarla. Per ulteriori informazioni, consulta Operazione di timeout per le modifiche di capacità.
- Pausa dopo l'inattività: utilizza l'impostazione opzionale Ridimensiona la capacità a 0 ACUs quando il cluster è inattivo per ridimensionare il database fino a zero capacità di elaborazione mentre è inattivo. Quando il traffico di database riprende, Aurora aumenta automaticamente la capacità di calcolo e si dimensiona per gestire il traffico.

> **Note**
>
> Quando si modifica l'intervallo di capacità per un Aurora Serverless II cluster DB, la modifica avviene immediatamente, indipendentemente dal fatto che si scelga di applicarla immediatamente o durante la successiva finestra di manutenzione programmata.

Console

Puoi modificare la configurazione di dimensionamento di un cluster di database Aurora dalla AWS Management Console.

Per modificare un Aurora Serverless v1 cluster di database

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona Aurora Serverless v1 Cluster DB che si desidera modificare.
4. Per Actions (Operazioni), scegliere Modify cluster (Modifica cluster).
5. Nella sezione Capacity settings (Impostazioni di capacità), modificare la configurazione di dimensionamento.

6. Scegli Continua.
7. Nella pagina Modifica il cluster DB rivedi le modifiche apportate, quindi scegli quando applicarle.
8. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per modificare la configurazione di scalabilità di un Aurora Serverless v1 Cluster DB che utilizza AWS CLI, esegui il modify-db-cluster AWS CLI comando. Specifica l'opzione `--scaling-configuration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

In questo esempio, si modifica la configurazione di scalabilità di un Aurora Serverless v1 Cluster DB denominato*`sample-cluster`*.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

API RDS

È possibile modificare la configurazione di scalabilità di un cluster Aurora DB con l'operazione DBCluster Modify API. Specifica il parametro `ScalingConfiguration` in modo che configuri la capacità minima, quella massima e la pausa automatica quando non sono presenti connessioni. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Aggiornamento della versione principale di un Aurora Serverless v1 cluster di database

> **Important**
>
> AWS ha annunciato la data per end-of-life Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Puoi aggiornare la versione principale per un Aurora Serverless v1 Cluster DB compatibile con PostgreSQL 11 fino a una versione corrispondente compatibile con PostgreSQL 13.

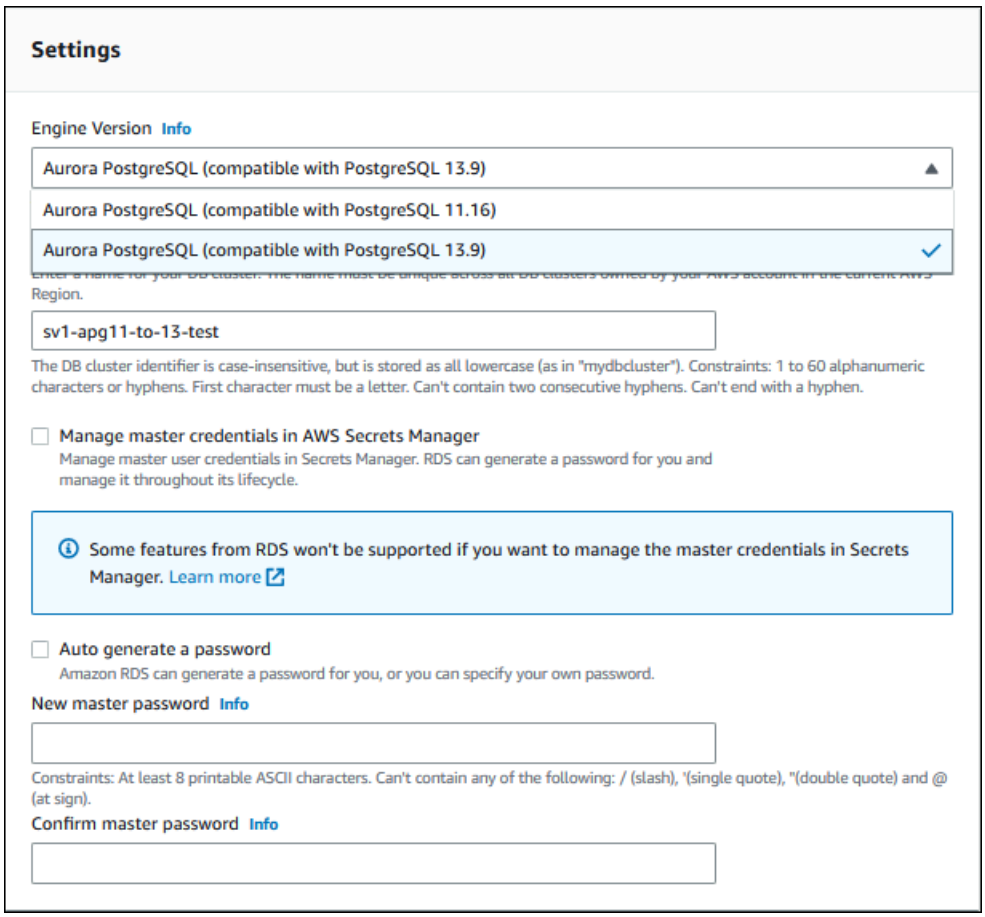
Console

È possibile eseguire un aggiornamento sul posto di un Aurora Serverless v1 Cluster DB che utilizza. AWS Management Console

Per aggiornare un Aurora Serverless v1 cluster di database

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona Aurora Serverless v1 Cluster DB che si desidera aggiornare.
4. Per Actions (Operazioni), scegliere Modify cluster (Modifica cluster).
5. In Versione, scegli 13 come numero di versione di Aurora PostgreSQL.

   Il seguente esempio mostra un aggiornamento locale da Aurora PostgreSQL 11.16 a 13.9.



Se esegui un aggiornamento della versione principale, non modificare tutte le altre proprietà. Per modificare una qualsiasi delle altre proprietà, esegui un'operazione Modifica al termine dell'aggiornamento.

6. Scegli Continua.
7. Nella pagina Modifica il cluster DB rivedi le modifiche apportate, quindi scegli quando applicarle.
8. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per eseguire un aggiornamento sul posto da un sistema compatibile con PostgreSQL 11 Aurora Serverless v1 Da cluster DB a un cluster compatibile con PostgreSQL 13, specifica il parametro `--engine-version` con un numero di versione 13 di Aurora PostgreSQL compatibile con Aurora Serverless v1. Includi anche il `--allow-major-version-upgrade` parametro.

In questo esempio, si modifica la versione principale di un file compatibile con PostgreSQL 11 Aurora Serverless v1 `sample-cluster`Cluster DB denominato. In questo modo viene eseguito un

aggiornamento sul posto a un sistema compatibile con PostgreSQL 13 Aurora Serverless v1 Cluster DB.

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --engine-version 13.serverless_12 \
    --allow-major-version-upgrade
```

In Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --engine-version 13.serverless_12 ^
    --allow-major-version-upgrade
```

API RDS

Per eseguire un aggiornamento sul posto da un sistema compatibile con PostgreSQL 11 Aurora Serverless v1 Da cluster DB a un cluster compatibile con PostgreSQL 13, specifica il parametro `EngineVersion` con un numero di versione 13 di Aurora PostgreSQL compatibile con Aurora Serverless v1. Includi anche il `AllowMajorVersionUpgrade` parametro.

Convertire un Aurora Serverless v1 Da cluster DB a provisioning

È possibile convertire un Aurora Serverless v1 Cluster DB in un cluster DB predisposto. Per eseguire la conversione, utilizza la AWS CLI o l'API Amazon RDS per modificare la classe dell'istanza DB in Provisioned. Utilizza i passaggi seguenti per modificare la classe dell'istanza DB.

L'esempio seguente mostra come utilizzare la AWS CLI per convertire un Aurora Serverless v1 Cluster DB in un cluster predisposto.

AWS CLI

Per convertire un Aurora Serverless v1 Un cluster DB in un cluster fornito, esegui il modify-db-cluster AWS CLI comando.

I parametri seguenti sono obbligatori:

- `--db-cluster-identifier`— Il Aurora Serverless v1 Cluster DB che stai convertendo in provisioned.
- `--engine-mode`: usa il valore `provisioned`.

- `--allow-engine-mode-change`
- `--db-cluster-instance-class`— Scegliete la classe di istanza DB per il cluster DB di cui è stato effettuato il provisioning in base alla capacità del Aurora Serverless v1 Cluster DB.

In questo esempio, converti un Aurora Serverless v1 Cluster DB denominato `sample-cluster` e utilizzo della classe di istanza `db.r5.xlarge` DB.

In Linux, macOS, oppure Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier sample-cluster \
--engine-mode provisioned \
--allow-engine-mode-change \
--db-cluster-instance-class db.r5.xlarge
```

In Windows:

```
aws rds modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--engine-mode provisioned ^
--allow-engine-mode-change ^
--db-cluster-instance-class db.r5.xlarge
```

L'esempio seguente mostra come utilizzare l'API Amazon RDS per convertire un Aurora Serverless v1 Cluster DB in un cluster predisposto.

API RDS

Per convertire un Aurora Serverless v1 Un cluster DB in un cluster fornito, utilizza l'operazione Modify DBCluster API.

I parametri seguenti sono obbligatori:

- `DBClusterIdentifier`— Il Aurora Serverless v1 Cluster DB che stai convertendo in provisioned.
- `EngineMode`: usa il valore `provisioned`.
- `AllowEngineModeChange`
- `DBClusterInstanceClass`— Scegliete la classe di istanza DB per il cluster DB di cui è stato effettuato il provisioning in base alla capacità del Aurora Serverless v1 Cluster DB.

Considerazioni sulla conversione da un Aurora Serverless v1 Un cluster DB a un cluster predisposto

Le seguenti considerazioni si applicano quando Aurora Serverless v1 Il cluster DB viene convertito in un cluster fornito:

- È possibile utilizzare questa conversione come parte dell'aggiornamento del cluster DB da Aurora Serverless v1 in Aurora Serverless v2. Per ulteriori informazioni, vedereAggiornamento da un Aurora Serverless v1 cluster a Aurora Serverless v2.
- Il processo di conversione crea un'istanza DB reader nel cluster DB, promuove l'istanza reader a istanza writer e quindi elimina l'originale Aurora Serverless v1 istanza. Quando si converte il cluster di database, non è possibile eseguire altre modifiche contemporaneamente, ad esempio la modifica della versione del motore di database o del gruppo di parametri del cluster di database. L'operazione di conversione viene applicata immediatamente e non può essere annullata.
- Durante la conversione, viene acquisito uno snapshot del cluster di database di backup in caso di errore. L'identificatore per lo snapshot del cluster di database ha il formato `pre-modify-engine-mode-`*`DB_cluster_identifier`*`-`*`timestamp`*.
- Aurora utilizza la versione secondaria predefinita corrente del motore di database per il cluster di database allocato.
- Se non fornite una classe di istanza DB per il cluster DB convertito, Aurora ne consiglia una basata sulla capacità massima dell'originale Aurora Serverless v1 Cluster DB. La capacità consigliata per le mappature delle classi di istanza è illustrata nella tabella seguente.

Serverless capacità massima (ACUs)	Classe di istanza database allocata
1	db.t3.small
2	db.t3.medium
4	db.t3.large
8	db.r5.large
16	db.r5.xlarge
32	db.r5.2xlarge

Serverless capacità massima (ACUs)	Classe di istanza database allocata
64	db.r5.4xlarge
128	db.r5.8xlarge
192	db.r5.12xlarge
256	db.r5.16xlarge
384	db.r5.24xlarge

> **Note**
>
> A seconda della classe di istanza DB scelta e dell'utilizzo del database, potresti riscontrare costi diversi per un cluster DB fornito rispetto a Aurora Serverless v1.
> Se converti il tuo Aurora Serverless v1 Il cluster DB in una classe di istanza DB espandibile (db.t*) potrebbe comportare costi aggiuntivi per l'utilizzo del cluster DB. Per ulteriori informazioni, consulta Tipi di classi di istanza database.

Dimensionamento Aurora Serverless v1 Capacità del cluster DB manualmente

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

In genere, Aurora Serverless v1 I cluster DB si scalano perfettamente in base al carico di lavoro. Tuttavia, la capacità potrebbe non essere sempre dimensionata in maniera rapida da soddisfare

estremi improvvisi, come ad esempio un aumento esponenziale delle transazioni. In questi casi è possibile avviare manualmente l'operazione di dimensionamento impostando un nuovo valore di capacità. Dopo aver impostato la capacità in modo esplicito, Aurora Serverless v1 ridimensiona automaticamente il cluster DB. Esegue questa operazione in base al periodo di attesa per la riduzione.

È possibile impostare in modo esplicito la capacità di un Aurora Serverless v1 Cluster DB su un valore specifico con AWS Management Console AWS CLI, the o l'API RDS.

Console

Puoi impostare le capacità di un cluster database Aurora dalla AWS Management Console.

Per modificare un Aurora Serverless v1 cluster di database

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Seleziona Aurora Serverless v1 Cluster DB che si desidera modificare.
4. Per Actions (Operazioni), selezionare Set capacity (Imposta capacità).
5. Nella finestra Dimensionamento della capacità del database seleziona quanto segue:

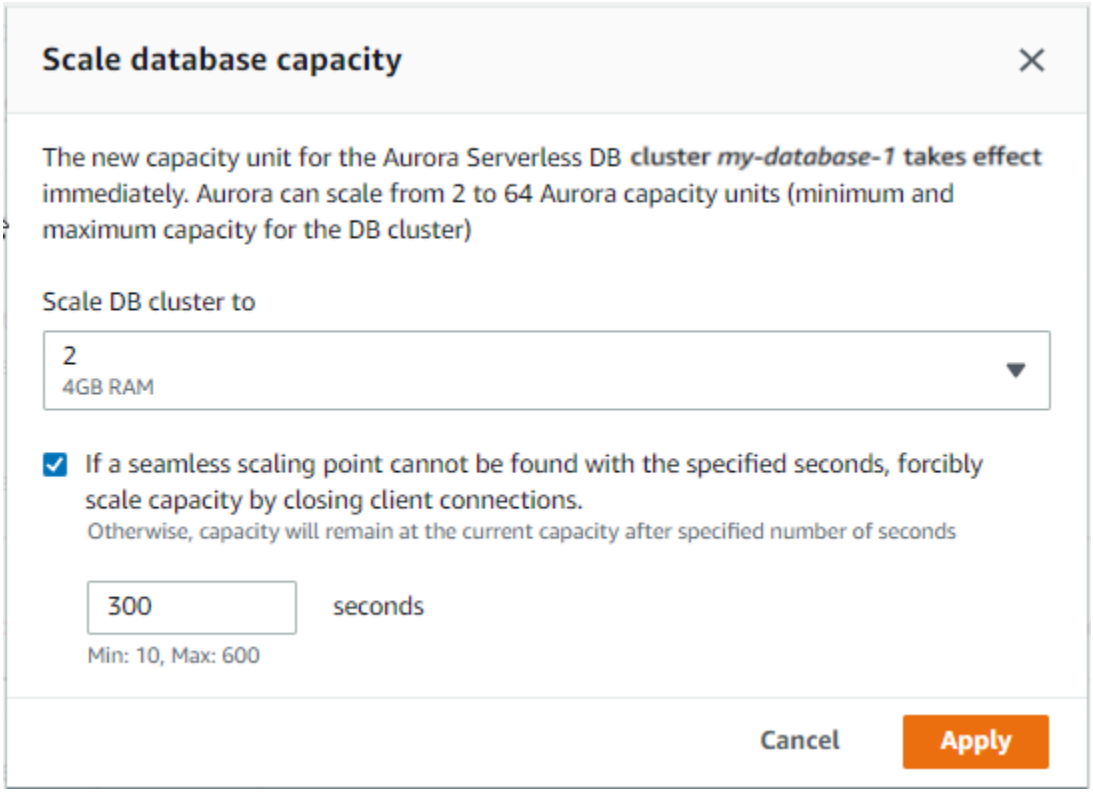
    a. Per il selettore a discesa Scale DB cluster to (Dimensionamento del cluster database a), seleziona la nuova capacità desiderata per il cluster database.

    b. Per la casella di controllo If a seamless scaling point not be found, scegli il comportamento che desideri per Aurora Serverless v1 L'`TimeoutAction`impostazione del cluster DB, come segue:

    - Deselezionate questa opzione se desiderate che la vostra capacità rimanga invariata se Aurora Serverless v1 non riesco a trovare un punto di scala prima del timeout.
    - Seleziona questa opzione se vuoi forzare il Aurora Serverless v1 Il cluster DB modifica la propria capacità anche se non riesce a trovare un punto di scalabilità prima del timeout. Questa opzione può risultare Aurora Serverless v1 interruzione delle connessioni che impediscono di trovare un punto di scala.

    c. Per secondi, inserisci il periodo di tempo che desideri concedere Aurora Serverless v1 Cluster DB per cercare un punto di scalabilità prima del timeout. Puoi specificare un qualsiasi valore compreso tra 10 e 600 secondi (10 minuti). L'impostazione predefinita è 5 minuti (300 secondi). L'esempio seguente impone il Aurora Serverless v1 Il cluster DB deve

essere ridotto a 2 ACUs anche se non riesce a trovare un punto di scalabilità entro cinque minuti.



6. Scegliere Apply (Applica).

Per ulteriori informazioni sui punti di ridimensionamento, `TimeoutAction`, e sui periodi di tempo di recupero, consulta Scalabilità automatica per Aurora Serverless v1.

AWS CLI

Per impostare la capacità di un Aurora Serverless v1 Cluster DB che utilizza AWS CLI, esegui il AWS CLI comando modify-current-db-cluster-capacity e specifica l'`--capacity`opzione. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

In questo esempio, si imposta la capacità di un Aurora Serverless v1 Cluster DB denominato *`sample-cluster`* to*64*.

```
aws rds modify-current-db-cluster-capacity --db-cluster-identifier sample-cluster --
capacity 64
```

API RDS

È possibile impostare la capacità di un cluster Aurora DB con l'operazione ModifyCurrentDBClusterCapacity API. Specifica il parametro `Capacity`. I valori di capacità validi includono quanto segue:

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 e 256.
- Aurora PostgreSQL: 2, 4, 8, 16, 32, 64, 192 e 384.

Visualizzazione Aurora Serverless v1 Cluster database

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster non migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

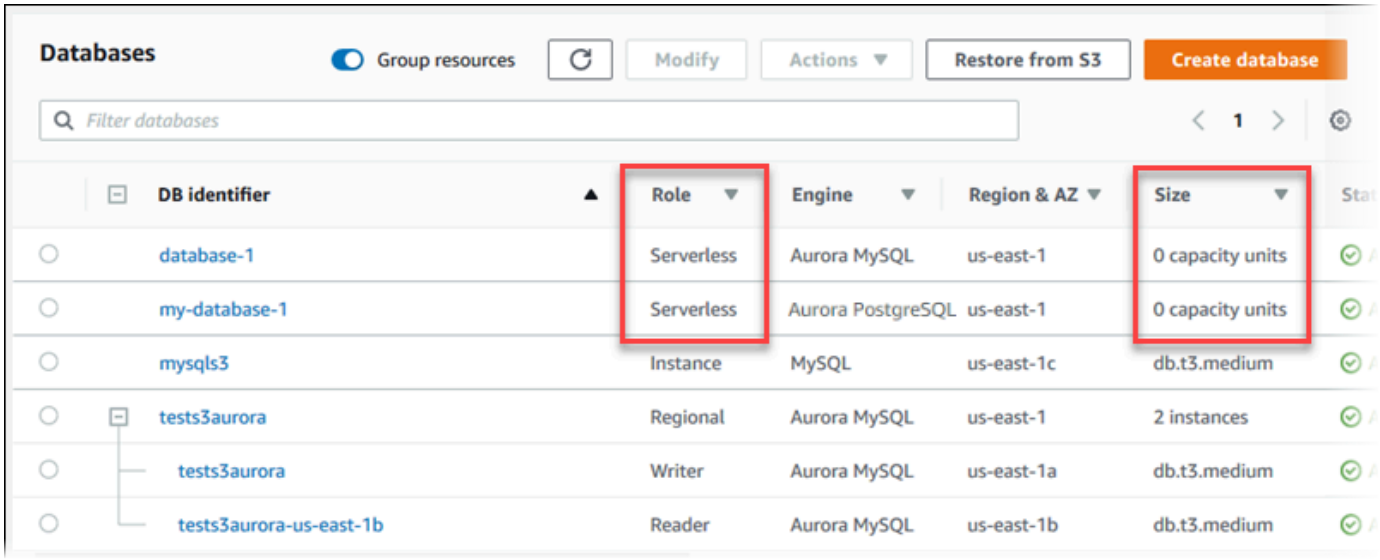
Dopo averne creato uno o più Aurora Serverless v1 Cluster DB, è possibile visualizzare quali cluster DB sono di tipo Serverless e quali di tipo Instance. È inoltre possibile visualizzare il numero attuale di unità di capacità Aurora () ciascuna ACUs Aurora Serverless v1 Il cluster DB utilizza. Ogni ACU è la combinazione di capacità di calcolo (CPU) e memoria (RAM).

Per visualizzare il tuo Aurora Serverless v1 Cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli il file Regione AWS in cui hai creato il Aurora Serverless v1 Cluster DB.

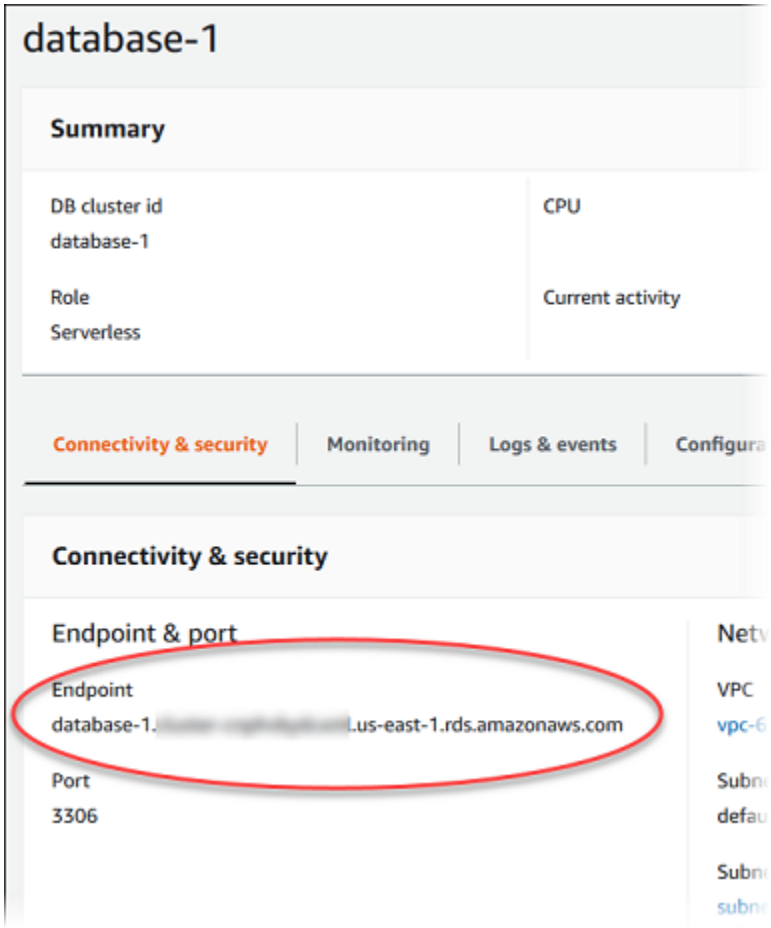
3. Nel riquadro di navigazione, scegliere Databases (Database).

   Per ogni cluster database, il tipo viene indicato su Role (Ruolo). Il Aurora Serverless v1 I cluster DB mostrano Serverless per il tipo. È possibile visualizzare un Aurora Serverless v1 La capacità attuale del cluster DB in Dimensioni.

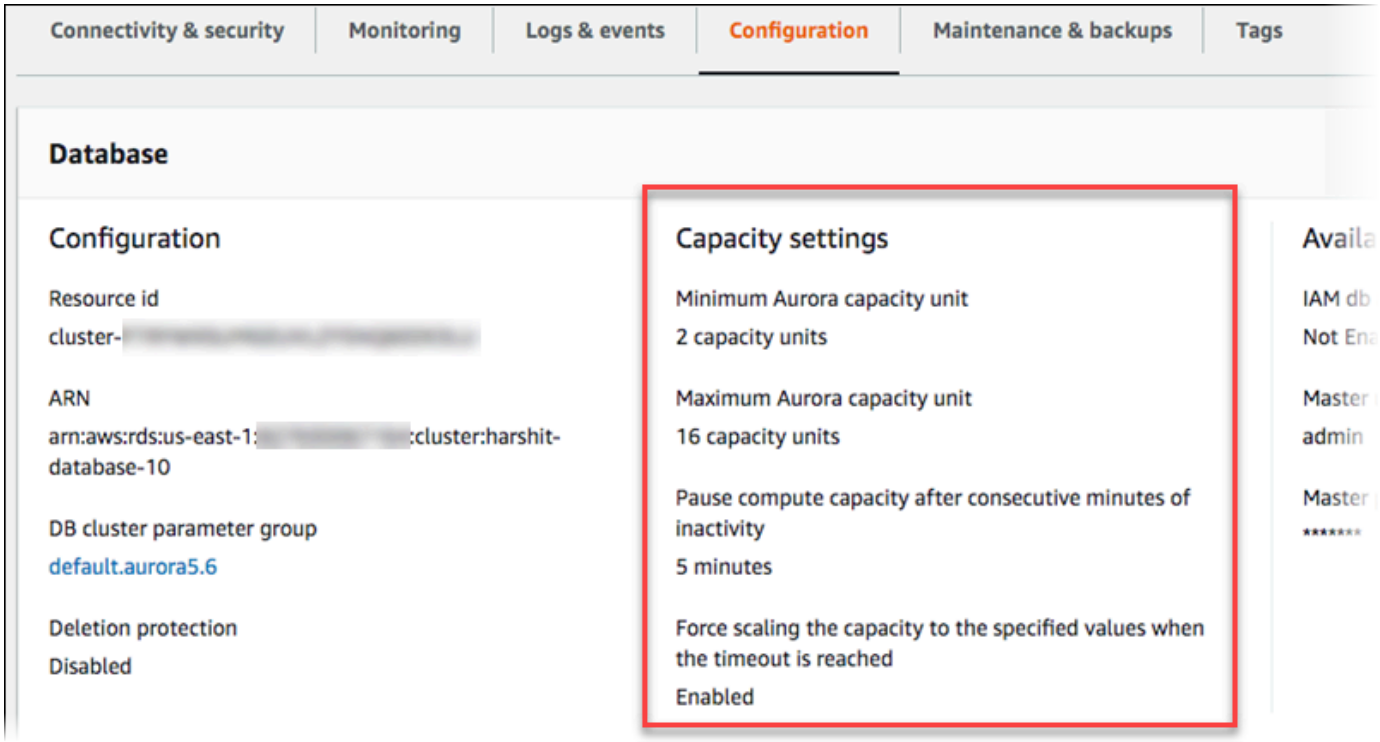


4. Scegli il nome di un Aurora Serverless v1 Cluster DB per visualizzarne i dettagli.

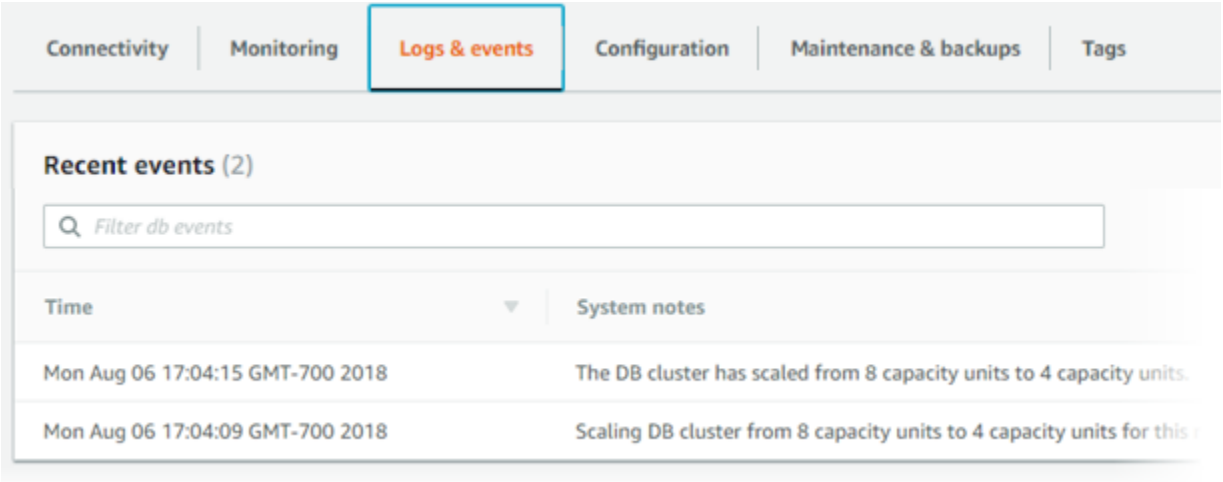
   Nella scheda Connettività e sicurezza, prendi nota dell'endpoint del database. Usa questo endpoint per connetterti al tuo Aurora Serverless v1 Cluster DB.



Seleziona la scheda Configurazione per visualizzare le impostazioni di capacità.



Viene generato un evento di dimensionamento quando il cluster database aumenta o riduce il dimensionamento, viene messo in pausa o ripristinato. Selezionare la scheda Logs & events (Log ed eventi) per visualizzare gli eventi recenti. La seguente immagine mostra esempi di tali eventi.



Monitoraggio della capacità e scalabilità degli eventi per il tuo Aurora Serverless v1 cluster di database

Puoi visualizzare i tuoi Aurora Serverless v1 Cluster DB CloudWatch inserito per monitorare la capacità allocata al cluster DB con la `ServerlessDatabaseCapacity` metrica. Puoi anche monitorare tutte le CloudWatch metriche Aurora standard, come, `CPUUtilization` `DatabaseConnectionsQueries`, e così via.

È possibile fare in modo che Aurora pubblichi alcuni o tutti i log del database su. CloudWatch È possibile selezionare i log da pubblicare abilitando i parametri di configurazione quali `general_log` e `slow_query_log` nel gruppo di parametri del cluster DB associato aAurora Serverless v1 cluster. A differenza dei cluster predisposti, Aurora Serverless v1 i cluster non richiedono di specificare nelle impostazioni del cluster DB su quali tipi di log caricare. CloudWatch Aurora Serverless v1 i cluster caricano automaticamente tutti i log disponibili. Quando si disabilita un parametro di configurazione del registro, la pubblicazione del registro si interrompe. CloudWatch È inoltre possibile eliminare i log in CloudWatch se non sono più necessari.

Per iniziare a usare Amazon CloudWatch per il tuo Aurora Serverless v1 Cluster DB, vediVisualizzazione Aurora Serverless v1 registri con Amazon CloudWatch. Per ulteriori informazioni su come monitorare i cluster Aurora DB, consulta CloudWatch. Monitoraggio degli eventi di registro in Amazon CloudWatch

Per connettersi a un Aurora Serverless v1 Cluster DB, usa l'endpoint del database. Per ulteriori informazioni, consulta Connessione a un cluster database Amazon Aurora.

> **Note**
>
> Non puoi connetterti direttamente a istanze DB specifiche nel tuo Aurora Serverless v1 Cluster DB.

Eliminazione di un Aurora Serverless v1 cluster di database

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno

migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

A seconda di come si crea un Aurora Serverless v1 Cluster DB, la protezione dall'eliminazione potrebbe essere attivata per impostazione predefinita. Non è possibile eliminare immediatamente un Aurora Serverless v1 Cluster DB con protezione da eliminazione abilitata. Per eliminare Aurora Serverless v1 Cluster DB che dispongono di protezione da eliminazione utilizzando la AWS Management Console, è innanzitutto necessario modificare il cluster per rimuovere questa protezione. Per informazioni sull'utilizzo di AWS CLI per questa attività, vedereAWS CLI.

Per disattivare la protezione da eliminazione utilizzando il AWS Management Console

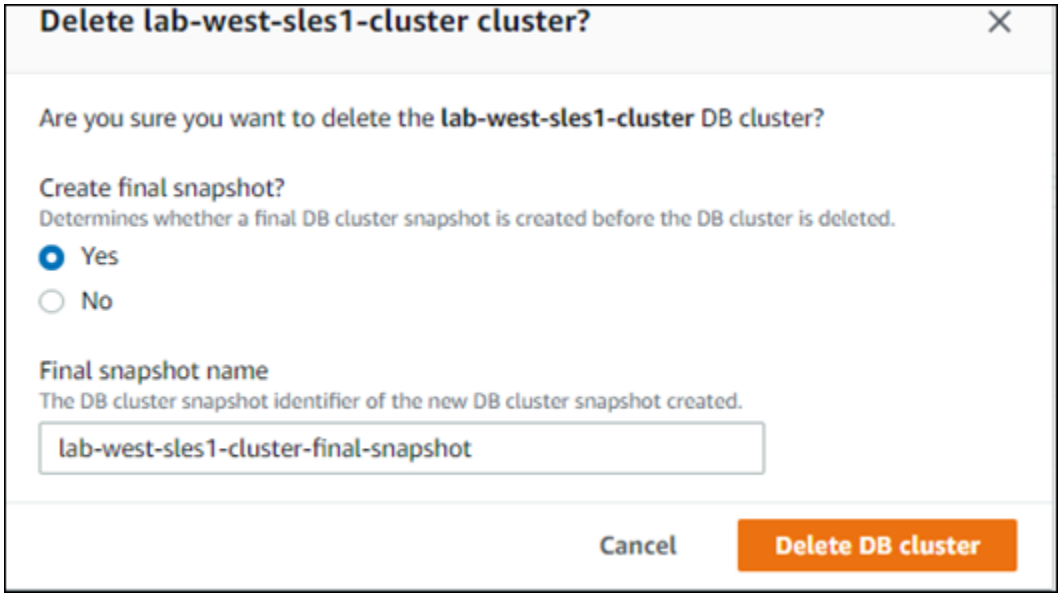
1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, seleziona DB clusters (Cluster database).
3. Scegli il tuo Aurora Serverless v1 Cluster DB dall'elenco.
4. Seleziona Modifica per aprire la configurazione del cluster database. La pagina Modifica cluster DB apre le Impostazioni, le impostazioni di Capacità e altri dettagli di configurazione per Aurora Serverless v1 Cluster DB. La protezione dall'eliminazione è disponibile nella sezione Additional configuration (Configurazione aggiuntiva).
5. Deseleziona l'opzione Enable deletion protection (Abilita protezione eliminazione) nella scheda delle proprietà Additional configuration (Configurazione aggiuntive).
6. Scegli Continue (Continua). Viene visualizzato il Riepilogo delle modifiche .
7. Seleziona Modifica cluster per accettare il riepilogo delle modifiche. Puoi inoltre scegliere Indietro per modificare le modifiche o Annulla per ignorarle.

Una volta che la protezione da eliminazione non è più attiva, puoi eliminare i Aurora Serverless v1 Cluster DB utilizzando AWS Management Console.

Console

Per eliminare un Aurora Serverless v1 cluster di database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nella sezione Risorse seleziona Cluster database.
3. Seleziona Aurora Serverless v1 Cluster DB che desideri eliminare.
4. In Actions (Azioni), scegliere Delete (Elimina). Ti viene richiesto di confermare che desideri eliminare il tuo Aurora Serverless v1 Cluster DB.
5. Si consiglia di mantenere le opzioni preselezionate:
   - Sì per Crea snapshot finale
   - Il tuo Aurora Serverless v1 Nome del cluster DB plus `-final-snapshot` per il nome dello snapshot finale. Tuttavia, puoi modificare il nome della snapshot finale in questo campo.



   Se si sceglie No per creare un'istantanea finale? non è possibile ripristinare il cluster DB utilizzando istantanee o point-in-time ripristino.
6. Seleziona Elimina cluster database.

Aurora Serverless v1 elimina il cluster DB. Se hai scelto di avere un'istantanea finale, vedrai il Aurora Serverless v1 Lo stato del cluster DB cambia in «Backup» prima di essere eliminato e non viene più visualizzato nell'elenco.

AWS CLI

Prima di iniziare, configuralo AWS CLI con il tuo AWS Access Key ID, AWS Secret Access Key e dove Regione AWS Aurora Serverless v1 Il cluster DB è. Per ulteriori informazioni, consulta Nozioni di base sulla configurazione nella Guida per l' AWS Command Line Interface utente.

Non è possibile eliminare un Aurora Serverless v1 Cluster DB fino a dopo aver disabilitato per la prima volta la protezione da eliminazione per i cluster configurati con questa opzione. Se provi a eliminare un cluster con questa opzione di protezione attivata, viene visualizzato il seguente messaggio di errore.

```
An error occurred (InvalidParameterCombination) when calling the DeleteDBCluster
  operation: Cannot delete protected Cluster, please disable deletion protection and
 try again.
```

Puoi cambiare il tuo Aurora Serverless v1 L'impostazione di protezione dall'eliminazione del cluster DB utilizzando il modify-db-cluster AWS CLI comando illustrato di seguito:

```
aws rds modify-db-cluster --db-cluster-identifier your-cluster-name --no-deletion-
protection
```

Con questo comando vengono restituite le proprietà revisionate per il cluster database specificato. Ora puoi eliminare il tuo Aurora Serverless v1 Cluster DB.

Ti consigliamo di creare sempre un'istantanea finale ogni volta che elimini un Aurora Serverless v1 Cluster DB. Il seguente esempio di utilizzo di AWS CLI delete-db-clustermostra come. Specifica il nome del cluster database e un nome per la snapshot.

In Linux, macOS, oppure Unix:

```
aws rds delete-db-cluster --db-cluster-identifier \
  your-cluster-name --no-skip-final-snapshot \
  --final-db-snapshot-identifier name-your-snapshot
```

In Windows:

```
aws rds delete-db-cluster --db-cluster-identifier ^
```

```
your-cluster-name --no-skip-final-snapshot ^
--final-db-snapshot-identifier name-your-snapshot
```

Aurora Serverless v1 e versioni del motore di database Aurora

> **Important**
>
> AWS ha annunciato la end-of-life data per Aurora Serverless v1: 31 marzo 2025. Tutti Aurora Serverless v1 i cluster che non verranno migrati entro il 31 marzo 2025 verranno migrati in Aurora Serverless v2 durante la finestra di manutenzione. Se l'aggiornamento non riesce, Amazon Aurora converte il cluster Serverless v1 in un cluster con provisioning con la versione del motore equivalente durante la finestra di manutenzione. Se applicabile, Amazon Aurora registrerà il cluster di provisioning convertito in Amazon RDS Extended Support. Per ulteriori informazioni, consulta Estensione del supporto di RDS.

Aurora Serverless v1 è disponibile solo in alcune versioni di Aurora MySQL Regioni AWS e Aurora PostgreSQL e Aurora PostgreSQL. Per l'elenco attuale di tale supporto Regioni AWS Aurora Serverless v1 e le versioni specifiche di Aurora MySQL e Aurora PostgreSQL disponibili in ogni regione, vedi. Aurora Serverless v1

Aurora Serverless v1 utilizza il motore di database Aurora associato per identificare le release specifiche supportate per ogni motore di database supportato, come segue:

- Aurora MySQL Serverless
- Aurora PostgreSQL Serverless

Quando le versioni minori dei motori di database diventano disponibili per Aurora Serverless v1, vengono applicati automaticamente nelle varie Regioni AWS sedi Aurora Serverless v1 è disponibile. In altre parole, non è necessario aggiornare il Aurora Serverless v1 DB cluster per ottenere una nuova versione secondaria per il motore DB del cluster quando sarà disponibile per Aurora Serverless v1.

Aurora MySQL Serverless

Aurora Serverless v1 è disponibile solo in alcune versioni di Aurora MySQL Regioni AWS e solo per specifiche. Per l'elenco attuale di tale supporto Regioni AWS Aurora Serverless v1 e le versioni specifiche di Aurora MySQL disponibili in ogni regione, vedi. Aurora Serverless v1 con Aurora MySQL

Per informazioni sui miglioramenti e sulle correzioni di bug per la versione 2 di Aurora MySQL, consulta Aggiornamenti del motore del database per Amazon Aurora MySQL versione 2 nelle Note di rilascio di Aurora MySQL.

Per utilizzare una versione più recente di Aurora MySQL, puoi usare Aurora Serverless v2. Per le Regioni AWS versioni MySQL e Aurora utilizzabili con Aurora Serverless v2, consulta Aurora Serverless v2 con Aurora MySQL. Per informazioni sull'utilizzo su Aurora Serverless v2, consulta Utilizzo Aurora Serverless v2.

Aurora PostgreSQL Serverless

Aurora Serverless v1 è disponibile solo in alcune versioni di Aurora PostgreSQL Regioni AWS e solo per specifiche. Per l'elenco attuale di tale supporto Regioni AWS Aurora Serverless v1 e le versioni specifiche di Aurora PostgreSQL disponibili in ogni regione, vedi. Aurora Serverless v1 con Aurora PostgreSQL

Se desideri utilizzare Aurora PostgreSQL per Aurora Serverless v1 Cluster DB, puoi usare le versioni compatibili con Aurora PostgreSQL 13. Le release secondarie per Aurora edizione compatibile con PostgreSQL includono solo le modifiche compatibili con le versioni precedenti. Il tuo Aurora Serverless v1 Il cluster DB viene aggiornato in modo trasparente quando una release secondaria di Aurora PostgreSQL diventa disponibile per Aurora Serverless v1 nel tuo. Regione AWS

Per utilizzare una versione più recente di Aurora PostgreSQL, puoi usare Aurora Serverless v2. Per le Regioni AWS versioni PostgreSQL e Aurora utilizzabili con Aurora Serverless v2, consulta Aurora Serverless v2 con Aurora PostgreSQL. Per informazioni sull'utilizzo su Aurora Serverless v2, consulta Utilizzo Aurora Serverless v2.

Aggiornamenti automatici delle versioni secondarie per Aurora Serverless v1

Quando le versioni minori dei motori di database diventano disponibili per Aurora Serverless v1, vengono applicati automaticamente nelle varie Regioni AWS sedi Aurora Serverless v1 è disponibile. In altre parole, non è necessario aggiornare il Aurora Serverless v1 DB cluster per ottenere una nuova versione secondaria per il motore DB del cluster quando sarà disponibile per Aurora Serverless v1.

Utilizzo dell'API dati Amazon RDS

Utilizzando RDS Data API (Data API), puoi lavorare con un'interfaccia di servizi Web per il tuo cluster Aurora DB. Data API non richiede una connessione persistente al cluster DB. Fornisce invece un endpoint HTTP sicuro e l'integrazione con AWS SDKs. Puoi usare l'endpoint per eseguire istruzioni SQL senza gestire connessioni.

Gli utenti non devono passare le credenziali con le chiamate a Data API, poiché Data API utilizza le credenziali del database archiviate in. AWS Secrets Manager Per archiviare le credenziali in Secrets Manager, agli utenti devono essere concesse le autorizzazioni appropriate per utilizzare Secrets Manager e anche Data API. Per ulteriori informazioni sull'autorizzazione degli utenti, consulta Autorizzazione dell'accesso all'API dei dati di Amazon RDS.

Puoi anche utilizzare Data API per integrare Amazon Aurora con altre AWS applicazioni come AWS Lambda AWS AppSync, e. AWS Cloud9 Data API offre un modo più sicuro di utilizzo AWS Lambda. Consente di accedere al cluster database senza dover configurare una funzione Lambda per accedere alle risorse in un cloud privato virtuale (VPC). Per ulteriori informazioni, consulta AWS Lambda, AWS AppSync e AWS Cloud9.

Puoi abilitare Data API quando crei il cluster Aurora DB. È inoltre possibile modificare la configurazione in un secondo momento. Per ulteriori informazioni, consulta Abilitazione dell'API dati Amazon RDS.

Dopo aver abilitato Data API, puoi anche utilizzare l'editor di query per eseguire query ad hoc senza configurare uno strumento di query per accedere ad Aurora in un VPC. Per ulteriori informazioni, consulta Utilizzo dell'editor di query Aurora.

Argomenti

- Disponibilità di regioni e versioni per l'Amazon RDS Data API
- Limitazioni per l'Amazon RDS Data API
- Confronto dei comportamenti delle API di Amazon RDS Data Aurora Serverless v2 e dei cluster forniti con i cluster Aurora Serverless v1
- Autorizzazione dell'accesso all'API dei dati di Amazon RDS
- Abilitazione dell'API dati Amazon RDS
- Creazione di un endpoint Amazon VPC per l'Amazon RDS Data API ()AWS PrivateLink

- Chiamata dell'API dati Amazon RDS
- Utilizzo della libreria client Java per RDS Data API
- Elaborazione dei risultati delle query Amazon RDS Data API in formato JSON
- Risoluzione dei problemi relativi all'API dati Amazon RDS
- Registrazione delle chiamate Amazon RDS Data API con AWS CloudTrail
- Monitoraggio delle query RDS Data API con Performance Insights

Disponibilità di regioni e versioni per l'Amazon RDS Data API

Per informazioni sulle regioni e le versioni del motore disponibili per Data API, consulta le seguenti sezioni.

Tipo di cluster	Disponibilità di regioni e versioni
Aurora PostgreSQL con provisioning e Serverless v2	API dati con Aurora PostgreSQL Serverless v2 e provisioning
Aurora con provisioning MySQL e Serverless v2	API dati con Aurora MySQL Serverless v2 e provisioning
Aurora PostgreSQL Serverless v1	API dati con Aurora PostgreSQL Serverless v1
Aurora MySQL Serverless v1	API dati con Aurora MySQL Serverless v1

Se hai bisogno di moduli crittografici convalidati da FIPS 140-2 quando accedi a Data API tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Limitazioni per l'Amazon RDS Data API

RDS Data API presenta le seguenti limitazioni:

- È possibile eseguire query Data API solo su istanze Writer in un cluster DB. Tuttavia, le istanze writer possono accettare sia query di scrittura che di lettura.

- Con i database globali Aurora, puoi abilitare Data API sui cluster DB primari e secondari. Tuttavia, un cluster secondario non dispone di un'istanza writer finché non viene promosso a primario. L'API Data richiede l'accesso all'istanza writer per l'elaborazione delle query, anche per le query di lettura. Di conseguenza, le query di lettura e scrittura inviate al cluster secondario hanno esito negativo poiché manca un'istanza writer. Una volta promosso un cluster secondario e quando è disponibile un'istanza writer, le query Data API su quell'istanza DB hanno esito positivo.
- L'API Data non è supportata nelle classi di istanze T DB.
- Per Aurora Serverless v2 i cluster DB predisposti, RDS Data API non supporta alcuni tipi di dati. Per l'elenco dei tipi supportati, consulta. the section called “Data API rispetto a Aurora Serverless v2 Aurora Serverless v1”
- Per i database Aurora PostgreSQL versione 14 e successive, Data API supporta solo la crittografia delle password. `scram-sha-256`
- Il limite di dimensioni della risposta è 1 MiB. Se la chiamata restituisce più di 1 MiB di dati di risposta, verrà terminata.
- InfattiAurora Serverless v1, il numero massimo di richieste al secondo è 1.000. Per tutti gli altri database supportati, non ci sono limiti.
- Il limite delle dimensioni dell'API dati è 64 KB per riga nel set di risultati restituito dal database. Assicurati che ogni riga di un set di risultati sia pari o inferiore a 64 KB.

Confronto dei comportamenti delle API di Amazon RDS Data Aurora Serverless v2 e dei cluster forniti con i cluster Aurora Serverless v1

I miglioramenti più recenti ad Amazon RDS Data APIs rendono i dati APIs disponibili per i cluster che utilizzano versioni recenti dei motori PostgreSQL o MySQL. Questi cluster possono essere configurati per utilizzare o fornire classi di istanze come o. Aurora Serverless v2 `db.r6g` `db.r6i`

Le seguenti sezioni descrivono le differenze delle API di Amazon RDS tra i cluster DB Aurora Serverless v2 e i cluster DB forniti. Aurora Serverless v1 Aurora Serverless v1 I cluster DB utilizzano la modalità `serverless` motore. I cluster DB predisposti utilizzano la `provisioned` modalità motore. Un cluster Aurora Serverless v2 DB utilizza anche la modalità `provisioned` motore e contiene una o più istanze Aurora Serverless v2 DB con la `db.serverless` classe di istanza.

Numero massimo di richieste al secondo

Aurora Serverless v1

I dati APIs possono soddisfare fino a 1.000 richieste al secondo.

Aurora Serverless v2

I dati APIs possono effettuare un numero illimitato di richieste al secondo.

Abilitazione o disabilitazione dell'API Amazon RDS Data su un database esistente

Aurora Serverless v1

- Con l'API Amazon RDS: utilizza l'`ModifyCluster`operazione e specifica `True` o`False`, a seconda dei casi, il `EnableHttpEndpoint` parametro.
- Con l'opzione AWS CLI— Usa l'`modify-db-cluster`operazione con l'`--no-enable-http-endpoint`opzione `--enable-http-endpoint` o, a seconda dei casi.

Aurora Serverless v2

- Con l'API Amazon RDS: utilizza le `DisableHttpEndpoint` operazioni `EnableHttpEndpoint` e.
- Con le operazioni:use and AWS CLI. `enable-http-endpoint` `disable-http-endpoint`

CloudTrail eventi

Aurora Serverless v1

Gli eventi delle chiamate Data API sono eventi di gestione. Per impostazione predefinita, questi eventi vengono inclusi automaticamente in un percorso. Per ulteriori informazioni, consulta the section called “Esclusione degli eventi Data API da un CloudTrail trail (Aurora Serverless v1solo)”.

Aurora Serverless v2

Gli eventi delle chiamate Data API sono eventi relativi ai dati. Per impostazione predefinita, questi eventi vengono esclusi automaticamente in un percorso. Per ulteriori informazioni, consulta the section called “Inclusione di eventi Data API in un CloudTrail percorso”.

Supporto per più istruzioni

Aurora Serverless v1

- Per Aurora MySQL, le istruzioni multiple non sono supportate.
- Per Aurora PostgreSQL, le istruzioni multiple restituiscono solo la prima risposta alla query.

Aurora Serverless v2

Le istruzioni multiple non sono supportate. Il tentativo di eseguire più istruzioni in una singola chiamata API restituisce i risultati. “`An error occurred (ValidationException) when calling the ExecuteStatement operation: Multistatements aren't supported.`” Per eseguire più istruzioni, effettuate chiamate `ExecuteStatement` API separate o usatele `BatchExecuteStatement` per l'elaborazione in batch.

L'esempio seguente mostra il messaggio di errore risultante da una chiamata API che tenta di eseguire una dichiarazione multipla.

```
 aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table; Select * FROM next_table;

                                "An error occurred (ValidationException) when calling
 the ExecuteStatement operation: Multistatements aren't supported.
```

L'esempio seguente esegue più istruzioni con chiamate API separate. `ExecuteStatement`

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table;"

aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM next_table;"
```

Richieste simultanee per lo stesso ID di transazione

Aurora Serverless v1

Le richieste successive attendono il completamento della richiesta corrente. L'applicazione deve gestire gli errori di timeout se il periodo di attesa è troppo lungo.

Aurora Serverless v2

Quando l'API Data riceve più richieste con lo stesso ID di transazione, restituisce immediatamente questo errore:

`DatabaseErrorException: Transaction is still running a query`

Questo errore si verifica in due situazioni:

- L'applicazione effettua richieste asincrone (come le JavaScript promesse) utilizzando lo stesso ID di transazione.
- Una richiesta precedente con quell'ID di transazione è ancora in fase di elaborazione.

L'esempio seguente mostra tutte le richieste eseguite in parallelo con`promise.all()`.

```
const api_calls = [];
for (let i = 0; i < 10; i++) {
api_calls.push(
    client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    )
);
}
await Promise.all(api_calls);
```

Per risolvere questo errore, attendi il completamento della richiesta corrente prima di inviare un'altra richiesta con lo stesso ID di transazione o rimuovi l'ID della transazione per consentire le richieste parallele.

L'esempio seguente mostra una chiamata API che utilizza l'esecuzione sequenziale con lo stesso ID di transazione.

```
 for (let i = 0; i < 10; i++) {
    await client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    ).promise()
);
}
```

BatchExecuteStatement comportamento

Per ulteriori informazioni su `BatchExecuteStatement`, consulta BatchExecuteStatement.

Aurora Serverless v1

L'oggetto campi generati nel risultato dell'aggiornamento include i valori inseriti.

Aurora Serverless v2

- Per Aurora MySQL, l'oggetto campi generati nel risultato dell'aggiornamento include i valori inseriti.
- Per Aurora PostgreSQL, l'oggetto campi generati è vuoto.

Comportamento ExecuteSQL

Per ulteriori informazioni su`ExecuteSQL`, vedere ExecuteSQL.

Aurora Serverless v1

L'`ExecuteSQL`operazione è obsoleta.

Aurora Serverless v2

L'`ExecuteSQL`operazione non è supportata.

ExecuteStatement comportamento

Per ulteriori informazioni su `ExecuteStatement`, consulta ExecuteStatement.

Aurora Serverless v1

Il `ExecuteStatement` parametro supporta il recupero di colonne di matrici multidimensionali e di tutti i tipi di dati avanzati.

Aurora Serverless v2

Il `ExecuteStatement` parametro non supporta colonne di matrici multidimensionali.
Inoltre, non supporta determinati tipi di dati PostgreSQL, inclusi i tipi geometrici e monetari.
Quando una Data API incontra un tipo di dati non supportato, restituisce questo errore:.
`UnsupportedResultException: The result contains the unsupported data type data_type`

Per risolvere questo problema, trasmetti il tipo di dati non supportato a. `TEXT` L'esempio seguente esegue il cast di un tipo di dati non supportato su. `TEXT`

```
SELECT custom_type::TEXT FROM my_table;--
ORSELECT CAST(custom_type AS TEXT) FROM my_table;
```

Per un elenco dei tipi di dati supportati per ogni motore di database Aurora, consulta Data API operations reference.

Comportamento dei parametri dello schema

Aurora Serverless v1

Il `Schema` parametro non è supportato. Quando includi il `Schema` parametro in una chiamata API, l'API Data ignora il parametro.

Aurora Serverless v2

Il parametro `Schema` è obsoleto. Quando includi il `Schema` parametro in una chiamata API, l'API Data restituisce questo errore:. `ValidationException: The schema parameter isn't supported` L'esempio seguente mostra una chiamata Data API che restituisce l'`ValidationException`errore.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
```

```
--schema "your_schema" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Per risolvere questo problema, rimuovi il `Schema` parametro dalla chiamata API.

L'esempio seguente mostra una chiamata Data API con il `Schema` parametro rimosso.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Autorizzazione dell'accesso all'API dei dati di Amazon RDS

Gli utenti possono richiamare le operazioni di Amazon RDS Data API (Data API) solo se sono autorizzati a farlo. Puoi concedere a un utente l'autorizzazione a utilizzare Data API allegando una policy AWS Identity and Access Management (IAM) che ne definisce i privilegi. È inoltre possibile associare la policy a un ruolo se si utilizzano ruoli IAM. Una policy AWS gestita include `AmazonRDSDataFullAccess` le autorizzazioni per Data API.

La `AmazonRDSDataFullAccess` politica include anche le autorizzazioni da cui l'utente può ottenere il valore di un segreto. AWS Secrets Manager Gli utenti devono utilizzare Secrets Manager per archiviare segreti da utilizzare nelle chiamate a Data API. L'utilizzo dei segreti significa che gli utenti non devono includere le credenziali del database per le risorse a cui destinano nelle chiamate a Data API. Data API chiama in modo trasparente Secrets Manager, che consente (o nega) la richiesta del segreto da parte dell'utente. Per informazioni sulla configurazione dei segreti da utilizzare con Data API, consulta. Archiviazione delle credenziali del database in AWS Secrets Manager

La `AmazonRDSDataFullAccess` policy fornisce l'accesso completo (tramite Data API) alle risorse. Puoi restringere l'ambito definendo le tue policy che specificano l'Amazon Resource Name (ARN) di una risorsa.

Ad esempio, la seguente politica mostra un esempio delle autorizzazioni minime richieste a un utente per accedere all'API Data per il cluster DB identificato dal relativo ARN. La policy include le autorizzazioni necessarie per accedere Secrets Manager e ottenere l'autorizzazione all'istanza DB per l'utente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:prod"
        }
    ]
}
```

Si consiglia di utilizzare un ARN specifico per l'elemento "Risorse" nelle istruzioni delle policy (come illustrato nell'esempio) anziché un carattere jolly (*).

Utilizzo dell'autorizzazione basata su tag

RDS Data API (Data API) e Secrets Manager supportano entrambi l'autorizzazione basata su tag. Itag sono coppie chiave-valore che etichettano una risorsa, ad esempio un cluster RDS, con un valore stringa aggiuntivo, ad esempio:

- `environment:production`
- `environment:development`

È possibile applicare tag alle risorse per l'allocazione dei costi, il supporto delle operazioni, il controllo degli accessi e molti altri motivi. Se non disponi già di tag sulle tue risorse e desideri applicarli, puoi saperne di più alla pagina Applicazione di tag alle risorse Amazon RDS. È possibile utilizzare i tag

nelle istruzioni delle policy per limitare l'accesso ai cluster RDS etichettati con questi tag. Ad esempio, un cluster DB Aurora potrebbe avere tag che identificano l'ambiente come produzione o sviluppo.

Nell'esempio seguente viene illustrato come utilizzare i tag nelle istruzioni delle policy. Questa istruzione richiede che sia il cluster e il segreto passato nella richiesta API dati abbiano un tag `environment:production`.

Ecco come viene applicata la policy: quando un utente effettua una chiamata utilizzando Data API, la richiesta viene inviata al servizio. L'API Data verifica innanzitutto che l'ARN del cluster passato nella richiesta sia taggato con. `environment:production` Quindi chiama Secrets Manager per recuperare il valore del segreto dell'utente nella richiesta. Secrets Manager verifica inoltre che il segreto dell'utente sia contrassegnato con `environment:production`. In tal caso, l'API dati utilizza quindi il valore recuperato per la password DB dell'utente. Infine, se anche questo è corretto, la richiesta dell'API dati viene richiamata correttamente per l'utente.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                 "secretsmanager:GetSecretValue"
               ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
                                       ]
                     }
             }
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                  "rds-data:*"
               ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:*",
            "Condition": {
                    "StringEquals": {
```

```
                    "aws:ResourceTag/environment": [
                                    "production"
                                ]
                    }
                }
            }
        ]
}
```

L'esempio mostra azioni separate per `rds-data` e `secretsmanager` per Data API e Secrets Manager. Tuttavia, è possibile combinare azioni e definire le condizioni dei tag in molti modi diversi per supportare i casi d'uso specifici. Per ulteriori informazioni, consulta Utilizzo delle policy basate su identità (policy IAM) per Secrets Manager.

Nell'elemento "Condizione" delle policy, è possibile scegliere le chiavi tag tra le seguenti:

- `aws:TagKeys`
- `aws:ResourceTag/${TagKey}`

Per ulteriori informazioni sui tag delle risorse e su come utilizzarli`aws:TagKeys`, consulta Controllo dell'accesso alle AWS risorse mediante i tag delle risorse.

> **Note**
>
> Sia Data API che AWS Secrets Manager autorizzano gli utenti. Se non si dispone delle autorizzazioni per tutte le azioni definite in una policy, viene visualizzato un errore `AccessDeniedException`.

Archiviazione delle credenziali del database in AWS Secrets Manager

Quando chiami Amazon RDS Data API (Data API), trasmetti le credenziali per il cluster Aurora DB utilizzando un segreto in Secrets Manager. Per utilizzare questo metodo per passare le credenziali, specifica il nome del segreto o l'Amazon Resource Name (ARN) del segreto.

Per archiviare le credenziali del cluster database in un segreto

1. Utilizzare Secrets Manager per creare un segreto contenente le credenziali per il cluster DB Aurora.

Per le istruzioni, consulta Creazione di un segreto del database nella Guida per l'utente di AWS Secrets Manager .

2. Usa la console Secrets Manager per visualizzare i dettagli del segreto che hai creato o esegui il `aws secretsmanager describe-secret` AWS CLI comando.

   Prendere nota del nome e dell'ARN del segreto, Puoi usarli nelle chiamate all'API Data.

Per ulteriori informazioni relative all'utilizzo di Secrets Manager, consulta la Guida per l'utente di AWS Secrets Manager.

Per informazioni su come Amazon Aurora gestisce la gestione delle identità e degli accessi, consulta Come funziona Amazon Aurora con IAM.

Per informazioni sulla creazione di una policy IAM, consulta Creazione di policy IAM nella Guida per l'utente di IAM. Per informazioni sull'aggiunta di una policy IAM a un utente, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

Abilitazione dell'API dati Amazon RDS

Per utilizzare l'Amazon RDS Data API (Data API), abilitala per il tuo cluster Aurora DB. Puoi abilitare Data API quando crei o modifichi il cluster DB.

> **Note**
>
> La disponibilità dell'API Data per il cluster dipende dalla versione di Aurora, dal motore di database e AWS dalla regione in uso. Per le versioni precedenti di Aurora, Data API funziona solo con Aurora Serverless v1 i cluster. Per le versioni più recenti di Aurora, Data API funziona con cluster che utilizzano sia il provisioning che le istanze. Aurora Serverless v2 Per verificare se il cluster può utilizzare Data API, consulta. Regioni supportate e motori Aurora DB per RDS Data API

Argomenti

- Abilitazione dell'API RDS Data quando crei un database
- Abilitazione dell'API RDS Data su un database esistente

Abilitazione dell'API RDS Data quando crei un database

Durante la creazione di un database che supporta RDS Data API (Data API), puoi abilitare questa funzionalità. Le seguenti procedure descrivono come eseguire questa operazione quando si utilizza la AWS Management Console AWS CLI, la o l'API RDS.

Console

Per abilitare Data API quando crei un cluster DB, seleziona la casella di controllo Abilita RDS Data API nella sezione Connettività della pagina Crea database, come nella schermata seguente.

Per istruzioni su come creare un cluster Aurora DB in grado di utilizzare l'API RDS Data, vedere quanto segue:

- Per i cluster Aurora Serverless v2 e i cluster a cui è stato assegnato il provisioning: Creazione di un cluster database Amazon Aurora
- Per — Aurora Serverless v1 Creare un Aurora Serverless v1 cluster di database

AWS CLI

Per abilitare Data API durante la creazione di un cluster Aurora DB, esegui il create-db-cluster AWS CLI comando con l'`--enable-http-endpoint`opzione.

L'esempio seguente crea un cluster Aurora PostgreSQL DB con Data API abilitata.

PerLinux, o: macOS Unix

```
aws rds create-db-cluster \
    --db-cluster-identifier my_pg_cluster \
    --engine aurora-postgresql \
    --enable-http-endpoint
```

Per Windows:

```
aws rds create-db-cluster ^
    --db-cluster-identifier my_pg_cluster ^
    --engine aurora-postgresql ^
    --enable-http-endpoint
```

API RDS

Per abilitare Data API durante la creazione di un cluster Aurora DB, utilizza l'DBClusteroperazione Create con il valore del `EnableHttpEndpoint` parametro impostato su. `true`

Abilitazione dell'API RDS Data su un database esistente

È possibile modificare un cluster DB che supporta RDS Data API (Data API) per abilitare o disabilitare questa funzionalità.

Argomenti

- Abilitazione o disabilitazione di Data API (Aurora Serverless v2e provisioning)
- Abilitazione o disabilitazione dell'API Data (Aurora Serverless v1solo)

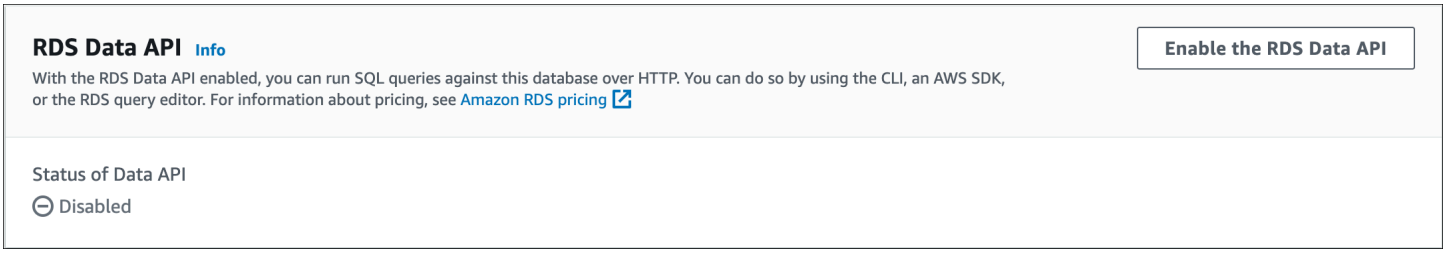
Abilitazione o disabilitazione di Data API (Aurora Serverless v2e provisioning)

Utilizza le seguenti procedure per abilitare o disabilitare Data API sui database forniti Aurora Serverless v2 e su cui è stato eseguito il provisioning. Per abilitare o disabilitare Data API sui Aurora Serverless v1 database, utilizza le procedure inthe section called “Abilitazione o disabilitazione dell'API Data (Aurora Serverless v1solo)”.

Console

È possibile abilitare o disabilitare Data API utilizzando la console RDS per un cluster DB che supporta questa funzionalità. A tale scopo, apri la pagina dei dettagli del cluster del database su cui desideri abilitare o disabilitare Data API e, nella scheda Connettività e sicurezza, vai alla sezione RDS Data API. Questa sezione mostra lo stato di Data API e consente di abilitarla o disabilitarla.

La schermata seguente mostra che l'API RDS Data non è abilitata.



AWS CLI

Per abilitare o disabilitare Data API su un database esistente, esegui il disable-http-endpoint AWS CLI comando enable-http-endpointo e specifica l'ARN del tuo cluster DB.

L'esempio seguente abilita Data API.

Per LinuxmacOS, oUnix:

```
aws rds enable-http-endpoint \
    --resource-arn cluster_arn
```

Per Windows:

```
aws rds enable-http-endpoint ^
    --resource-arn cluster_arn
```

API RDS

Per abilitare o disabilitare Data API su un database esistente, usa le DisableHttpEndpointoperazioni EnableHttpEndpointand.

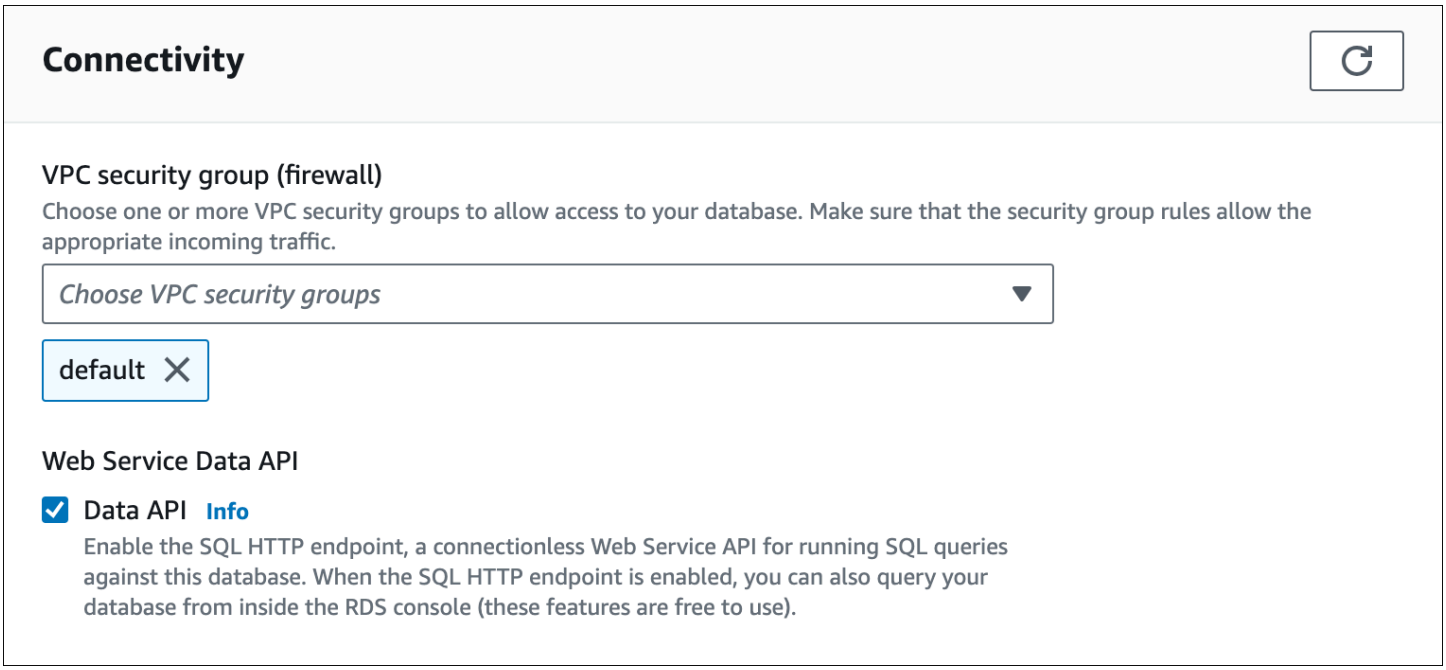
Abilitazione o disabilitazione dell'API Data (Aurora Serverless v1solo)

Utilizza le seguenti procedure per abilitare o disabilitare Data API sui Aurora Serverless v1 database esistenti. Per abilitare o disabilitare Data API sui Aurora Serverless v2 database a cui è stato eseguito il provisioning, utilizza le procedure inthe section called “Abilitazione o disabilitazione dell'API Data”.

Console

Quando modifichi un cluster Aurora Serverless v1 DB, abiliti Data API nella sezione Connettività della console RDS.

La schermata seguente mostra l'API Data abilitata durante la modifica di un cluster Aurora DB.



Per istruzioni su come modificare un cluster Aurora Serverless v1 DB, consultaModificare un Aurora Serverless v1 cluster di database.

AWS CLI

Per abilitare o disabilitare Data API, esegui il modify-db-cluster AWS CLI comando con `--enable-http-endpoint` o`--no-enable-http-endpoint`, a seconda dei casi.

L'esempio seguente abilita Data API on`sample-cluster`.

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --enable-http-endpoint
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --enable-http-endpoint
```

API RDS

Per abilitare Data API, utilizza l'DBClusteroperazione Modifica e imposta il valore di su `true` o`false`, `EnableHttpEndpoint` a seconda dei casi.

Creazione di un endpoint Amazon VPC per l'Amazon RDS Data API ()AWS PrivateLink

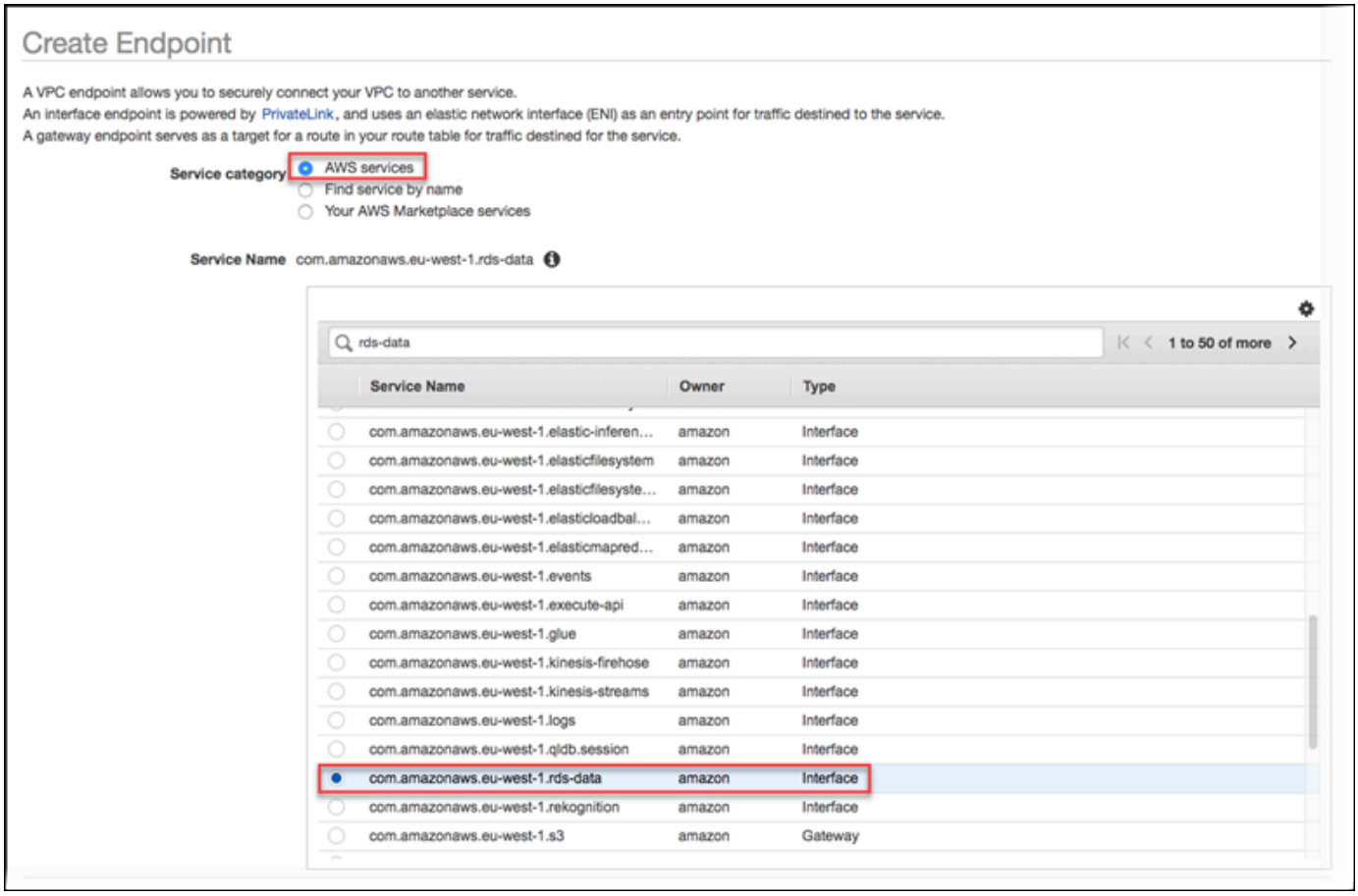
Amazon VPC ti consente di lanciare AWS risorse, come cluster e applicazioni Aurora DB, in un cloud privato virtuale (VPC). AWS PrivateLink fornisce connettività privata tra VPCs e AWS servizi con elevata sicurezza sulla rete Amazon. Utilizzando AWS PrivateLink, puoi creare endpoint Amazon VPC, che ti consentono di connetterti ai servizi su diversi account e basati VPCs su Amazon VPC. Per ulteriori informazioni su AWS PrivateLink, consultare Servizi endpoint VPC (AWS PrivateLink) nella Guida per l'utente di Amazon Virtual Private Cloud.

Puoi chiamare RDS Data API (Data API) con endpoint Amazon VPC. L'uso di un endpoint Amazon VPC mantiene il traffico tra le applicazioni in Amazon VPC e Data API nella AWS rete, senza utilizzare indirizzi IP pubblici. Gli endpoint Amazon VPC consentono di soddisfare i requisiti di conformità e normativi relativi alla limitazione della connettività Internet. Ad esempio, se utilizzi un endpoint Amazon VPC, puoi mantenere il traffico tra un'applicazione in esecuzione su un' EC2 istanza Amazon e l'API Data VPCs che li contiene.

Dopo aver creato l'endpoint Amazon VPC, puoi iniziare a utilizzarlo senza apportare modifiche al codice o alla configurazione nell'applicazione.

Per creare un endpoint Amazon VPC per l'API Data

1. Accedi AWS Management Console e apri la console Amazon VPC all'indirizzo. https://console.aws.amazon.com/vpc/
2. Scegliere Endpoint, quindi Create Endpoint (Crea endpoint).
3. Nella pagina Crea endpoint, per Categoria di servizio, seleziona Servizi AWS . Per Service Name (Nome servizio), scegliere rds-data.



4. Per VPC, scegliere il VPC in cui creare l'endpoint.

   Scegliere il VPC che contiene l'applicazione che effettua chiamate API dati.

5. Per le sottoreti, scegli la sottorete per ogni zona di disponibilità (AZ) utilizzata dal AWS servizio che esegue l'applicazione.

Per creare un endpoint Amazon VPC, specificare l'intervallo di indirizzi IP privati in cui l'endpoint sarà accessibile. A tale scopo, scegliere la sottorete per ogni zona di disponibilità. Questo ha l'effetto di limitare l'endpoint VPC all'intervallo di indirizzi IP privati specifico per ciascuna zona di disponibilità e crea inoltre un endpoint Amazon VPC in ogni zona di disponibilità.

6. Per Enable DNS Name (Abilita nome DNS), seleziona Enable for this endpoint (Abilita per questo endpoint).

   Il DNS privato risolve il nome host DNS dell'API dati standard (`https://rds-data.`*`region`*`.amazonaws.com`) negli indirizzi IP privati associati al nome host DNS specifico dell'endpoint Amazon VPC. Di conseguenza, puoi accedere all'endpoint VPC Data API utilizzando AWS CLI o AWS SDKs senza apportare modifiche al codice o alla configurazione per aggiornare l'URL dell'endpoint di Data API.

7. Per Security group (Gruppo di sicurezza), scegli un gruppo di sicurezza da associare all'endpoint Amazon VPC.

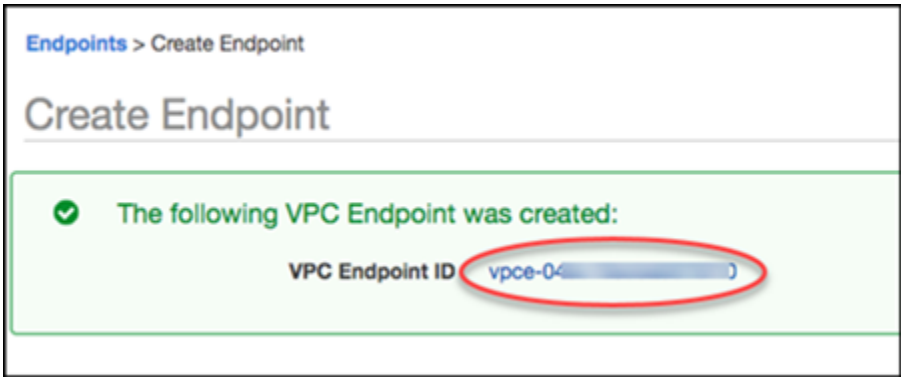
   Scegli il gruppo di sicurezza che consente l'accesso al AWS servizio su cui è in esecuzione l'applicazione. Ad esempio, se un' EC2 istanza Amazon esegue la tua applicazione, scegli il gruppo di sicurezza che consente l'accesso all' EC2 istanza Amazon. Il gruppo di sicurezza consente di controllare il traffico verso l'endpoint Amazon VPC dalle risorse del VPC.

8. Per Policy, scegli Full Access (Accesso completo) per consentire a chiunque all'interno del Amazon VPC di accedere all'API dati tramite questo endpoint. Oppure scegli Custom (Personalizzato) per specificare una policy che limita l'accesso.

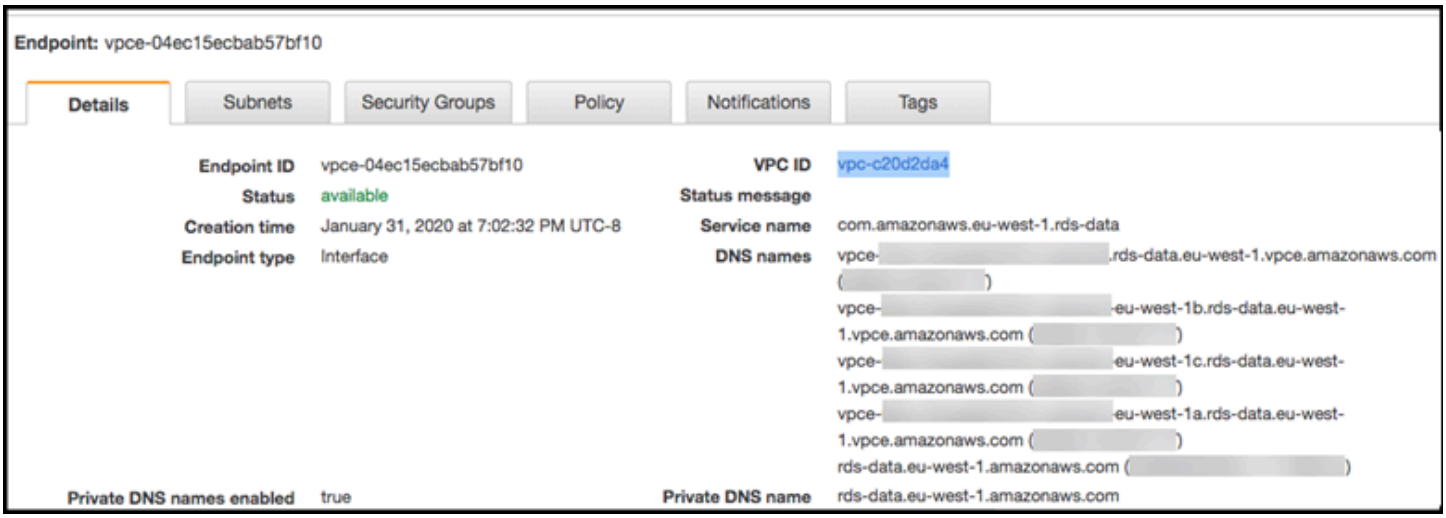
   Se scegli Personalizzato, immetti la policy nello strumento di creazione delle policy.

9. Selezionare Create endpoint (Crea endpoint).

Dopo aver creato l'endpoint, scegli il link in AWS Management Console per visualizzare i dettagli dell'endpoint.



La scheda Details (Dettagli) dell'endpoint mostra i nomi host DNS generati durante la creazione dell'endpoint Amazon VPC.



È possibile utilizzare l'endpoint standard (`rds-data.`*`region`*`.amazonaws.com`) o uno degli endpoint specifici di VPC per chiamare l'API dati all'interno di Amazon VPC. L'endpoint API dati standard esegue automaticamente l'instradamento all'endpoint Amazon VPC. Questo routing si verifica perché il nome host DNS privato è stato abilitato al momento della creazione dell'endpoint Amazon VPC.

Quando utilizzi un endpoint Amazon VPC in una chiamata Data API, tutto il traffico tra l'applicazione e l'API Data rimane nell'Amazon VPCs che lo contiene. Puoi utilizzare un endpoint Amazon VPC per qualsiasi tipo di chiamata API dati. Per informazioni sulla chiamata a Data API, consulta. Chiamata dell'API dati Amazon RDS

Chiamata dell'API dati Amazon RDS

Con l'Amazon RDS Data API (Data API) abilitata sul tuo cluster Aurora DB, puoi eseguire istruzioni SQL sul cluster Aurora DB utilizzando Data API o. AWS CLI Data API supporta i linguaggi di programmazione supportati da. AWS SDKs Per ulteriori informazioni, consulta Strumenti su cui costruire AWS.

Argomenti

- Riferimento alle operazioni dell'API Amazon RDS Data
- Chiamata dell'API dati Amazon RDS con AWS CLI
- Chiamata dell'API Amazon RDS Data da un'applicazione Python
- Chiamata dell'API Amazon RDS Data da un'applicazione Java

- Controllo del comportamento di timeout dell'API Data

Riferimento alle operazioni dell'API Amazon RDS Data

L'API Amazon RDS Data fornisce le seguenti operazioni per eseguire istruzioni SQL.

Operazione API dati	AWS CLI comando	Descrizione
ExecuteStatement	aws rds-data execute-statement	Esegue un'istruzione SQL in un database.
BatchExecuteStatement	aws rds-data batch-execute-statement	Esegue un'istruzione SQL batch su un array di dati per operazioni di aggiornamento in blocco e di inserimento. Puoi eseguire un'istruzione DML (Data Manipolation Language) con una matrice di set di parametri. Un'istruzione SQL batch può fornire un miglioramento significativo delle prestazioni su singole operazioni di inserimento e aggiornamento.

Puoi utilizzare entrambe le operazioni per eseguire singole istruzioni SQL o per eseguire transazioni. Per le transazioni, Data API fornisce le seguenti operazioni.

Operazione API dati	AWS CLI comando	Descrizione
BeginTransaction	aws rds-data begin-transaction	Inizia una transazione SQL.
CommitTransaction	aws rds-data commit-transaction	Termina una transazione SQL ed esegue il commit delle modifiche.
RollbackTransaction	aws rds-data rollback-transaction	Esegue un rollback di una transazione.

Le operazioni per l'esecuzione di istruzioni SQL e il supporto delle transazioni hanno i seguenti parametri e AWS CLI opzioni comuni di Data API. Alcune operazioni supportano altri parametri o opzioni.

Parametro operazione API dati	AWS CLI opzione di comando	Richiesto	Descrizione
`resourceArn`	`--resource-arn`	Sì	L'Amazon Resource Name (ARN) del cluster Aurora DB. Il cluster deve avere lo stesso Account AWS ruolo o utente IAM che richiama l'API Data. Per accedere a un cluster in un account diverso, assumi un ruolo in quell'account.
`secretArn`	`--secret-arn`	Sì	Il nome o l'ARN del secreto che abilita l'accesso al cluster database.

RDS Data API supporta i seguenti tipi di dati per Aurora MySQL:

- `TINYINT(1)`, `BOOLEAN`, `BOOL`
- `TINYINT`
- `SMALLINT` [`SIGNED` | `UNSIGNED`]
- `MEDIUMINT` [`SIGNED` | `UNSIGNED`]
- `INT` [`SIGNED` | `UNSIGNED`]
- `BIGINT` [`SIGNED` | `UNSIGNED`]
- `FLOAT`
- `DOUBLE`
- `VARCHAR`, `CHAR`, `TEXT`, `ENUM`
- `VARBINARY`, `BINARY`, `BLOB`
- `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`
- `DECIMAL`
- `JSON`

- BIT, BIT(N)

RDS Data API supporta i seguenti tipi scalari Aurora PostgreSQL:

- BOOL
- BYTEA
- DATE
- CIDR
- DECIMAL, NUMERIC
- ENUM
- FLOAT8, DOUBLE PRECISION
- INET
- INT, INT4, SERIAL
- INT2, SMALLINT, SMALLSERIAL
- INT8, BIGINT, BIGSERIAL
- JSONB, JSON
- REAL, FLOAT
- TEXT, CHAR(N), VARCHAR, NAME
- TIME
- TIMESTAMP
- UUID
- VECTOR

RDS Data API supporta i seguenti tipi di array Aurora PostgreSQL:

- BOOL[], BIT[]
- DATE[]
- DECIMAL[], NUMERIC[]
- FLOAT8[], DOUBLE PRECISION[]
- INT[], INT4[]
- INT2[]
- INT8[], BIGINT[]

- JSON[]
- REAL[], FLOAT[]
- TEXT[], CHAR(N)[], VARCHAR[], NAME[]
- TIME[]
- TIMESTAMP[]
- UUID[]

È possibile utilizzare i parametri nelle chiamate Data API verso ExecuteStatement eBatchExecuteStatement, quando si eseguono i comandi e. AWS CLI execute-statement batch-execute-statement Per utilizzare un parametro, specifica una coppia nome-valore nel tipo di dati SqlParameter. Specifica il valore con il tipo di dati Field. La tabella seguente associa i tipi di dati Java Database Connectivity (JDBC) ai tipi di dati specificati nelle chiamate API dati.

Tipo di dati JDBC	Tipo di dati API dati
INTEGER, TINYINT, SMALLINT, BIGINT	LONG (o STRING)
FLOAT, REAL, DOUBLE	DOUBLE
DECIMAL	STRING
BOOLEAN, BIT	BOOLEAN
BLOB, BINARY, LONGVARBINARY, VARBINARY	BLOB
CLOB	STRING
Altri tipi (inclusi i tipi correlati a data e ora)	STRING

> **Note**
>
> Puoi specificare il tipo di dati LONG o STRING nella chiamata all'API data per i valori LONG restituiti dal database. Ti consigliamo di farlo per evitare di perdere la precisione per numeri estremamente grandi, cosa che può succedere quando lavori con JavaScript.

Alcuni tipi, come `DECIMAL` e`TIME`, richiedono un suggerimento affinché Data API passi `String` i valori al database come tipo corretto. Per utilizzare un suggerimento, includere i valori per `typeHint` nel tipo di dati `SqlParameter`. i seguito sono riportati i valori possibili per `typeHint`:

- `DATE` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `DATE` al database. Il formato accettato è `YYYY-MM-DD`.
- `DECIMAL` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `DECIMAL` al database.
- `JSON` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `JSON` al database.
- `TIME` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `TIME` al database. Il formato accettato è `HH:MM:SS[.FFF]`.
- `TIMESTAMP` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `TIMESTAMP` al database. Il formato accettato è `YYYY-MM-DD HH:MM:SS[.FFF]`.
- `UUID` – Il valore del parametro `String` corrispondente viene inviato come oggetto di tipo `UUID` al database.

> **Note**
>
> Attualmente, Data API non supporta array di Universal Unique Identifiers (). UUIDs

> **Note**
>
> Per Amazon Aurora PostgreSQL, Data API restituisce sempre il tipo di dati Aurora PostgreSQL nel fuso orario UTC. `TIMESTAMPTZ`

Chiamata dell'API dati Amazon RDS con AWS CLI

Puoi chiamare RDS Data API (Data API) utilizzando. AWS CLI

I seguenti esempi utilizzano l'API AWS CLI for Data. Per ulteriori informazioni, consulta la Documentazione di riferimento AWS CLI per l'API dati.

In ogni esempio, sostituisci l'Amazon Resource Name (ARN) per il cluster DB con l'ARN per il tuo cluster Aurora DB. Inoltre, sostituisci l'ARN segreto con l'ARN del segreto in Secrets Manager che consente l'accesso al cluster database.

> **Note**
>
> AWS CLI Può formattare le risposte in JSON.

Argomenti

- Avvio di una transazione SQL
- Esecuzione di un'istruzione SQL
- Esecuzione di un'istruzione SQL batch su un array di dati
- Esecuzione del commit di una transazione SQL
- Rollback di una transazione SQL

Avvio di una transazione SQL

Puoi avviare una transazione SQL utilizzando il comando CLI `aws rds-data begin-transaction`. La chiamata restituisce un identificatore di transazione.

> **Important**
>
> In Data API, una transazione scade se non ci sono chiamate che utilizzano il relativo ID entro tre minuti. Se una transazione scade prima che venga confermata, Data API la ripristina automaticamente.
> Le istruzioni DDL (Data Definition Language) MySQL all'interno di una transazione causano un commit implicito. Si consiglia di eseguire ogni istruzione MySQL DDL in un comando `execute-statement` separato con l'opzione. `--continue-after-timeout`

Oltre alle opzioni comuni, specifica l'opzione `--database`, che fornisce il nome del database.

Ad esempio, il comando CLI seguente avvia una transazione SQL.

PerLinux, o: macOS Unix

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Per Windows:

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Di seguito è riportato un esempio della risposta.

```
{
    "transactionId": "ABC1234567890xyz"
}
```

Esecuzione di un'istruzione SQL

Puoi eseguire un'istruzione SQL utilizzando il comando CLI `aws rds-data execute-statement`.

Puoi eseguire un'istruzione SQL in una transazione specificando l'identificatore di transazione con l'opzione `--transaction-id`. Puoi avviare una transazione utilizzando il comando CLI `aws rds-data begin-transaction`. Puoi terminare ed eseguire il commit di una transazione utilizzando il comando CLI `aws rds-data commit-transaction`.

> **Important**
>
> Se non specifichi l'opzione `--transaction-id`, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

Oltre alle opzioni comuni, specifica le seguenti opzioni:

- `--sql` (obbligatoria) – Un'istruzione SQL da eseguire sul cluster database.
- `--transaction-id` (facoltativa) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione in cui desideri includere l'istruzione SQL.

- `--parameters` (facoltativa) – I parametri per l'istruzione SQL.
- `--include-result-metadata | --no-include-result-metadata` (opzionale) – Un valore che indica se includere metadati nei risultati. Il valore di default è `--no-include-result-metadata`.
- `--database` (opzionale) – Il nome del database.

  L'opzione `--database` potrebbe non funzionare quando si esegue un'istruzione SQL dopo l'esecuzione di `--sql "use database_name;"` nella richiesta precedente. Si consiglia di utilizzare l'opzione `--database` invece di eseguire le istruzioni `--sql "use database_name;"`.
- `--continue-after-timeout | --no-continue-after-timeout`(opzionale): un valore che indica se continuare a eseguire l'istruzione dopo la chiamata supera l'intervallo di timeout dell'API Data di 45 secondi. Il valore di default è `--no-continue-after-timeout`.

  Per istruzioni DDL (Data Definition Language), è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata per evitare errori e la possibilità di strutture dati danneggiate.
- `--format-records-as "JSON"|"NONE"` - Un valore facoltativo che specifica se formattare il set di risultati come stringa JSON. Il valore predefinito è `"NONE"`. Per informazioni sull'utilizzo dell'elaborazione di set di risultati JSON, consulta Elaborazione dei risultati delle query Amazon RDS Data API in formato JSON.

Il cluster database restituisce una risposta per la chiamata.

> **Note**
>
> Il limite di dimensioni della risposta è 1 MiB. Se la chiamata restituisce più di 1 MiB di dati di risposta, verrà terminata.
> InfattiAurora Serverless v1, il numero massimo di richieste al secondo è 1.000. Per tutti gli altri database supportati, non ci sono limiti.

Ad esempio, il comando CLI seguente esegue una singola istruzione SQL e omette i metadati nei risultati (il valore predefinito).

Per LinuxmacOS, oUnix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "select * from mytable"
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "select * from mytable"
```

Di seguito è riportato un esempio della risposta.

```
{
    "numberOfRecordsUpdated": 0,
    "records": [
        [
            {
                "longValue": 1
            },
            {
                "stringValue": "ValueOne"
            }
        ],
        [
            {
                "longValue": 2
            },
            {
                "stringValue": "ValueTwo"
            }
        ],
        [
            {
                "longValue": 3
            },
            {
                "stringValue": "ValueThree"
            }
```

```
        ]
    ]
}
```

Il comando CLI seguente esegue una singola istruzione SQL in una transazione specificando l'opzione `--transaction-id`.

Per LinuxmacOS, oUnix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Di seguito è riportato un esempio della risposta.

```
{
    "numberOfRecordsUpdated": 1
}
```

Il comando CLI seguente esegue una singola istruzione SQL con parametri.

Per LinuxmacOS, oUnix:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Di seguito è riportato un esempio della risposta.

```
{
    "numberOfRecordsUpdated": 1
}
```

Il comando CLI seguente esegue un'istruzione SQL DDL (Data Definition Language). L'istruzione DDL rinomina la colonna `job` nella colonna `role`.

> **Important**
>
> Per istruzioni DDL, è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata. Quando un'istruzione DDL termina prima che l'esecuzione sia terminata, può causare errori e verosimilmente strutture dati danneggiate. Per continuare a eseguire un'istruzione dopo che una chiamata supera l'intervallo di timeout dell'RDS Data API di 45 secondi, specifica l'opzione. `--continue-after-timeout`

PerLinux, o: macOS Unix

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Per Windows:

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Di seguito è riportato un esempio della risposta.

```
{
    "generatedFields": [],
    "numberOfRecordsUpdated": 0
}
```

> **Note**
>
> I dati `generatedFields` non sono supportati da Aurora PostgreSQL. Per ottenere i valori dei campi generati, utilizza la clausola `RETURNING`. Per ulteriori informazioni, consulta Returning Data From Modified Rows nella documentazione PostgreSQL.

Esecuzione di un'istruzione SQL batch su un array di dati

Puoi eseguire un'istruzione SQL batch su un'array di dati utilizzando il comando CLI `aws rds-data batch-execute-statement`. Puoi utilizzare questo comando per eseguire un'operazione di importazione in blocco o di aggiornamento.

Puoi eseguire un'istruzione SQL in una transazione specificando l'identificatore di transazione con l'opzione `--transaction-id`. Puoi avviare una transazione utilizzando il comando CLI `aws rds-data begin-transaction`. Puoi terminare ed eseguire il commit di una transazione utilizzando il comando CLI `aws rds-data commit-transaction`.

> **Important**
>
> Se non specifichi l'opzione `--transaction-id`, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

Oltre alle opzioni comuni, specifica le seguenti opzioni:

- `--sql` (obbligatoria) – Un'istruzione SQL da eseguire sul cluster database.

> **Tip**
>
> Perché le istruzioni siano compatibili con MySQL, non includere un punto e virgola alla fine del parametro `--sql`. Un punto e virgola finale potrebbe causare un errore di sintassi.

- `--transaction-id` (facoltativa) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione in cui desideri includere l'istruzione SQL.
- `--parameter-set` (facoltativa) – Il set di parametri per l'operazione batch.
- `--database` (opzionale) – Il nome del database.

Il cluster database restituisce una risposta alla chiamata.

> **Note**
>
> Non è previsto alcun limite massimo al numero di set di parametri. Tuttavia, la dimensione massima della richiesta HTTP inviata tramite Data API è di 4 MiB. Se la richiesta supera questo limite, Data API restituisce un errore e non elabora la richiesta. Questo limite di 4 MiB include la dimensione delle intestazioni HTTP e la notazione JSON nella richiesta. Pertanto, il numero di set di parametri che è possibile includere dipende da una combinazione di fattori, ad esempio la dimensione dell'istruzione SQL e la dimensione di ogni set di parametri.
> Il limite di dimensioni della risposta è 1 MiB. Se la chiamata restituisce più di 1 MiB di dati di risposta, verrà terminata.
> InfattiAurora Serverless v1, il numero massimo di richieste al secondo è 1.000. Per tutti gli altri database supportati, non ci sono limiti.

Ad esempio, il seguente comando CLI esegue un'istruzione SQL batch su un'array di dati con un set di parametri.

Per LinuxmacOS, oUnix:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" \
```

```
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

Per Windows:

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" ^
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

> **Note**
>
> Non includere interruzioni di riga nell'opzione `--parameter-sets`.

Esecuzione del commit di una transazione SQL

Utilizzando il comando CLI `aws rds-data commit-transaction`, puoi terminare una transazione SQL avviata con `aws rds-data begin-transaction` ed eseguire il commit delle modifiche.

Oltre alle opzioni di comando, specifica l'opzione seguente:

- `--transaction-id` (obbligatorio) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione che desideri terminare e di cui eseguire il commit.

Ad esempio, il comando CLI seguente termina una transazione SQL ed esegue il commit delle modifiche.

Per LinuxmacOS, oUnix:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Per Windows:

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Di seguito è riportato un esempio della risposta.

```
{
    "transactionStatus": "Transaction Committed"
}
```

Rollback di una transazione SQL

Utilizzando il comando CLI `aws rds-data rollback-transaction`, puoi eseguire il rollback di una transazione SQL avviata con `aws rds-data begin-transaction`. Il rollback di una transazione annulla le relative modifiche.

> **Important**
>
> Se l'ID transazione è scaduto, il rollback della transazione è stato eseguito automaticamente. In questo caso, un comando `aws rds-data rollback-transaction` che specifica l'ID transazione scaduto restituisce un errore.

Oltre alle opzioni di comando, specifica l'opzione seguente:

- `--transaction-id` (obbligatorio) – L'identificatore di una transazione che è stata avviata utilizzando il comando CLI `begin-transaction`. Specifica l'ID transazione della transazione di cui si desidera eseguire il rollback.

Ad esempio, il AWS CLI comando seguente ripristina una transazione SQL.

Per LinuxmacOS, oUnix:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Per Windows:

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Di seguito è riportato un esempio della risposta.

```
{
    "transactionStatus": "Rollback Complete"
    }
```

Chiamata dell'API Amazon RDS Data da un'applicazione Python

Puoi chiamare Amazon RDS Data API (Data API) da un'applicazione Python.

I seguenti esempi utilizzano l' AWS SDK per Python (Boto). Per ulteriori informazioni su Boto, consulta la documentazione di SDK AWS for Python (Boto 3).

In ogni esempio, sostituisci l'Amazon Resource Name (ARN) del cluster DB con l'ARN per il tuo cluster Aurora DB. Inoltre, sostituisci l'ARN segreto con l'ARN del segreto in Secrets Manager che consente l'accesso al cluster database.

Argomenti

- Esecuzione di una query SQL
- Esecuzione di un'istruzione SQL DML
- Esecuzione di una transazione SQL

Esecuzione di una query SQL

Puoi eseguire un'istruzione `SELECT` e recuperare i risultati da un'applicazione Python.

L'esempio seguente esegue una query SQL.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

response1 = rdsData.execute_statement(
            resourceArn = cluster_arn,
            secretArn = secret_arn,
            database = 'mydb',
            sql = 'select * from employees limit 3')

print (response1['records'])
[
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'ROSALEZ'
        },
        {
            'stringValue': 'ALEJANDRO'
        },
        {
            'stringValue': '2016-02-15 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'DOE'
        },
        {
            'stringValue': 'JANE'
```

```
        },
        {
            'stringValue': '2014-05-09 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'STILES'
        },
        {
            'stringValue': 'JOHN'
        },
        {
            'stringValue': '2017-09-20 04:34:33.0'
        }
    ]
]
```

Esecuzione di un'istruzione SQL DML

Puoi eseguire un'istruzione DML (Data Manipulation Language) per inserire, aggiornare o eliminare dati nel database. Puoi anche utilizzare parametri in istruzioni DML.

> **Important**
>
> Se una chiamata non fa parte di una transazione perché non include il parametro `transactionID`, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente esegue un'istruzione SQL insert e utilizza i parametri.

```
import boto3

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

rdsData = boto3.client('rds-data')
```

```
param1 = {'name':'firstname', 'value':{'stringValue': 'JACKSON'}}
param2 = {'name':'lastname', 'value':{'stringValue': 'MATEO'}}
paramSet = [param1, param2]

response2 = rdsData.execute_statement(resourceArn=cluster_arn,
                                      secretArn=secret_arn,
                                      database='mydb',
                                      sql='insert into employees(first_name, last_name)
 VALUES(:firstname, :lastname)',
                                      parameters = paramSet)

print (response2["numberOfRecordsUpdated"])
```

Esecuzione di una transazione SQL

Puoi avviare una transazione SQL, eseguire una o più istruzioni SQL, quindi eseguire il commit delle modifiche con un'applicazione Python.

> **Important**
>
> Una transazione scade se non ci sono chiamate che utilizzano il suo ID transazione in un periodo di tre minuti. Se una transazione scade prima che venga eseguito il commit della stessa, viene automaticamente sottoposta a rollback.
> Se non specifichi un ID transazione, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente esegue una transazione SQL che inserisce una riga in una tabella.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

tr = rdsData.begin_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb')
```

```
response3 = rdsData.execute_statement(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb',
     sql = 'insert into employees(first_name, last_name) values('XIULAN', 'WANG')',
     transactionId = tr['transactionId'])

cr = rdsData.commit_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     transactionId = tr['transactionId'])

cr['transactionStatus']
'Transaction Committed'

response3['numberOfRecordsUpdated']
1
```

> **Note**
>
> Se esegui un'istruzione DDL (Data Definition Language), è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata. Quando un'istruzione DDL termina prima che l'esecuzione sia terminata, può causare errori e verosimilmente strutture dati danneggiate. Per continuare a eseguire un'istruzione dopo che una chiamata supera l'intervallo di timeout dell'API RDS Data di 45 secondi, imposta il parametro su. `continueAfterTimeout` `true`

Chiamata dell'API Amazon RDS Data da un'applicazione Java

Puoi chiamare Amazon RDS Data API (Data API) da un'applicazione Java.

Gli esempi seguenti utilizzano l' AWS SDK for Java. Per ulteriori informazioni, consulta la Guida per gli sviluppatori di AWS SDK per Java.

In ogni esempio, sostituisci l'Amazon Resource Name (ARN) del cluster DB con l'ARN per il tuo cluster Aurora DB. Inoltre, sostituisci l'ARN segreto con l'ARN del segreto in Secrets Manager che consente l'accesso al cluster database.

Argomenti

- Esecuzione di una query SQL

- Esecuzione di una transazione SQL
- Esecuzione di un'operazione SQL batch

Esecuzione di una query SQL

Puoi eseguire un'istruzione SELECT e recuperare i risultati con un'applicazione Java.

L'esempio seguente esegue una query SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementResult;
import com.amazonaws.services.rdsdata.model.Field;

import java.util.List;

public class FetchResultsExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    ExecuteStatementRequest request = new ExecuteStatementRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb")
            .withSql("select * from mytable");

    ExecuteStatementResult result = rdsData.executeStatement(request);

    for (List<Field> fields: result.getRecords()) {
      String stringValue = fields.get(0).getStringValue();
      long numberValue = fields.get(1).getLongValue();

      System.out.println(String.format("Fetched row: string = %s, number = %d",
 stringValue, numberValue));
```

```
    }
  }
}
```

Esecuzione di una transazione SQL

Puoi avviare una transazione SQL, eseguire una o più istruzioni SQL, quindi eseguire il commit delle modifiche con un'applicazione Java.

> **Important**
>
> Una transazione scade se non ci sono chiamate che utilizzano il suo ID transazione in un periodo di tre minuti. Se una transazione scade prima che venga eseguito il commit della stessa, viene automaticamente sottoposta a rollback.
> Se non specifichi un ID transazione, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente esegue una transazione SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BeginTransactionRequest;
import com.amazonaws.services.rdsdata.model.BeginTransactionResult;
import com.amazonaws.services.rdsdata.model.CommitTransactionRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;

public class TransactionExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BeginTransactionRequest beginTransactionRequest = new BeginTransactionRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb");
```

```
    BeginTransactionResult beginTransactionResult =
 rdsData.beginTransaction(beginTransactionRequest);
    String transactionId = beginTransactionResult.getTransactionId();

    ExecuteStatementRequest executeStatementRequest = new ExecuteStatementRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table VALUES ('hello world!')");
    rdsData.executeStatement(executeStatementRequest);

    CommitTransactionRequest commitTransactionRequest = new CommitTransactionRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN);
    rdsData.commitTransaction(commitTransactionRequest);
  }
}
```

> **Note**
>
> Se esegui un'istruzione DDL (Data Definition Language), è consigliabile continuare a eseguire l'istruzione dopo il timeout della chiamata. Quando un'istruzione DDL termina prima che l'esecuzione sia terminata, può causare errori e verosimilmente strutture dati danneggiate. Per continuare a eseguire un'istruzione dopo che una chiamata supera l'intervallo di timeout dell'API RDS Data di 45 secondi, imposta il parametro su. `continueAfterTimeout` `true`

Esecuzione di un'operazione SQL batch

Puoi eseguire operazioni di inserimento a blocchi e di aggiornamento su un'array di dati con un'applicazione Java. Puoi eseguire un'istruzione DML con un array di set di parametri.

> **Important**
>
> Se non specifichi un ID transazione, viene eseguito automaticamente il commit delle modifiche risultanti dalla chiamata.

L'esempio seguente illustra un'operazione di inserimento in batch.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BatchExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.Field;
import com.amazonaws.services.rdsdata.model.SqlParameter;

import java.util.Arrays;

public class BatchExecuteExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
      AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BatchExecuteStatementRequest request = new BatchExecuteStatementRequest()
            .withDatabase("test")
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table2 VALUES (:string, :number)")
            .withParameterSets(Arrays.asList(
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("Hello")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(1L))
                    ),
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("World")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(2L))
                    )
            ));

    rdsData.batchExecuteStatement(request);
  }
}
```

Controllo del comportamento di timeout dell'API Data

Tutte le chiamate all'API dati sono sincrone. Supponiamo di eseguire un'operazione Data API che esegue un'istruzione SQL come `INSERT` o. `CREATE TABLE` Se la chiamata Data API viene restituita correttamente, l'elaborazione SQL viene completata al ritorno della chiamata.

Per impostazione predefinita, l'API Data annulla un'operazione e restituisce un errore di timeout se l'operazione non termina l'elaborazione entro 45 secondi. In tal caso, i dati non vengono inseriti, la tabella non viene creata e così via.

Puoi utilizzare l'API Data per eseguire operazioni di lunga durata che non possono essere completate entro 45 secondi. Se prevedi che un'operazione come un'operazione in blocco `INSERT` o un'operazione DDL su una tabella di grandi dimensioni richieda più di 45 secondi, puoi specificare il `continueAfterTimeout` parametro per l'operazione. `ExecuteStatement` L'applicazione continua a ricevere l'errore di timeout. Tuttavia, l'operazione continua a essere eseguita e non viene annullata. Per vedere un esempio, consulta Esecuzione di una transazione SQL.

Se l' AWS SDK per il tuo linguaggio di programmazione ha un proprio periodo di timeout per le chiamate API o le connessioni socket HTTP, assicurati che tutti questi periodi di timeout siano superiori a 45 secondi. Per alcuni SDKs, il periodo di timeout è inferiore a 45 secondi per impostazione predefinita. Ti consigliamo di impostare qualsiasi periodo di timeout specifico dell'SDK o del client su almeno un minuto. In questo modo si evita la possibilità che l'applicazione riceva un errore di timeout mentre l'operazione Data API viene ancora completata correttamente. In questo modo, puoi essere sicuro se riprovare l'operazione o meno.

Ad esempio, supponiamo che l'SDK restituisca un errore di timeout all'applicazione, ma che l'operazione Data API venga comunque completata entro l'intervallo di timeout dell'API Data. In tal caso, ritentare l'operazione potrebbe inserire dati duplicati o altrimenti produrre risultati errati. L'SDK potrebbe ritentare l'operazione automaticamente, causando dati errati senza alcuna azione da parte dell'applicazione.

L'intervallo di timeout è particolarmente importante per Java 2 SDK. In quell'SDK, il timeout delle chiamate API e il timeout del socket HTTP sono entrambi di 30 secondi per impostazione predefinita. Ecco un esempio di impostazione di tali timeout su un valore più alto:

```
public RdsDataClient createRdsDataClient() {
    return RdsDataClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
```

```
        .httpClientBuilder(createHttpClientBuilder())
        .credentialsProvider(defaultCredentialsProvider()) // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return ApacheHttpClient.builder() // Change this to your desired HttpClient
        .socketTimeout(Duration.ofSeconds(60));
}
```

Ecco un esempio equivalente che utilizza il client di dati asincrono:

```
public static RdsDataAsyncClient createRdsDataAsyncClient() {
    return RdsDataAsyncClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .credentialsProvider(defaultCredentialsProvider())  // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return NettyNioAsyncHttpClient.builder() // Change this to your desired
 AsyncHttpClient
        .readTimeout(Duration.ofSeconds(60));
}
```

Utilizzo della libreria client Java per RDS Data API

È possibile scaricare e utilizzare una libreria client Java per RDS Data API (Data API). Questa libreria client Java offre un modo alternativo di utilizzare Data API. Utilizzando questa libreria, puoi mappare le classi lato client alle richieste e alle risposte di Data API. Questo supporto per la mappatura può facilitare l'integrazione con alcuni tipi Java specifici, ad esempio `Date`, `Time` e `BigDecimal`.

Download della libreria client Java per l'API dati

La libreria client Java Data API è open source GitHub nella seguente posizione:

https://github.com/awslabs/rds-data-api-client-libreria-java

Puoi creare manualmente la libreria dai file di origine, ma la best practice consiste nell'utilizzare la libreria utilizzando la gestione delle dipendenze di Apache Maven. Aggiungi la seguente dipendenza al file Maven POM.

Per la versione 2.x, che è compatibile con AWS SDK 2.x, usa quanto segue:

```
<dependency>
   <groupId>software.amazon.rdsdata</groupId>
   <artifactId>rds-data-api-client-library-java</artifactId>
   <version>2.0.0</version>
</dependency>
```

Per la versione 1.x, che è compatibile con AWS SDK 1.x, utilizzate quanto segue:

```
<dependency>
    <groupId>software.amazon.rdsdata</groupId>
    <artifactId>rds-data-api-client-library-java</artifactId>
    <version>1.0.8</version>
</dependency>
```

Esempi di librerie client Java

Di seguito puoi trovare alcuni esempi comuni di utilizzo della libreria client Java dell'API dati. Questi esempi presuppongono che sia disponibile una tabella `accounts` con due colonne: `accountId` e `name`. Hai anche il seguente oggetto di trasferimento dati (DTO).

```
public class Account {
    int accountId;
```

```
    String name;
    // getters and setters omitted
}
```

La libreria client consente di passare DTOs come parametri di input. L'esempio seguente mostra come i clienti DTOs vengono mappati ai set di parametri di input.

```
var account1 = new Account(1, "John");
var account2 = new Account(2, "Mary");
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParamSets(account1, account2)
         .execute();
```

In alcuni casi, è più facile lavorare con valori semplici come parametri di input. Puoi farlo con la seguente sintassi.

```
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParameter("accountId", 3)
         .withParameter("name", "Zhang")
         .execute();
```

Di seguito è riportato un altro esempio che funziona con valori semplici come parametri di input.

```

client.forSql("INSERT INTO accounts(accountId, name) VALUES(?, ?)", 4, "Carlos")
         .execute();

```

La libreria client fornisce la mappatura automatica del DTOs momento in cui viene restituito un risultato. Gli esempi seguenti mostrano come il risultato viene mappato al tuo. DTOs

```
List<Account> result = client.forSql("SELECT * FROM accounts")
          .execute()
          .mapToList(Account.class);

Account result = client.forSql("SELECT * FROM accounts WHERE account_id = 1")
          .execute()
          .mapToSingle(Account.class);
```

In molti casi, il set di risultati del database contiene solo un singolo valore. Al fine di semplificare il recupero di tali risultati, la libreria client offre le seguenti API:

```
int numberOfAccounts = client.forSql("SELECT COUNT(*) FROM accounts")
          .execute()
          .singleValue(Integer.class);
```

> **Note**
>
> La funzione `mapToList` converte un set di risultati SQL in un elenco di oggetti definito dall'utente. Non è supportato l'utilizzo dell'istruzione `.withFormatRecordsAs(RecordsFormatType.JSON)` in una chiamata `ExecuteStatement` per la libreria client Java, perché ha lo stesso scopo. Per ulteriori informazioni, consulta Elaborazione dei risultati delle query Amazon RDS Data API in formato JSON.

Elaborazione dei risultati delle query Amazon RDS Data API in formato JSON

Quando invochi l'operazione `ExecuteStatement`, puoi scegliere di restituire i risultati della query come stringa in formato JSON. In questo modo puoi utilizzare le funzionalità di parsificazione JSON del linguaggio di programmazione per interpretare e riformattare il set di risultati. Ciò può aiutare a evitare di scrivere codice aggiuntivo per scorrere il set di risultati e interpretare ogni valore di colonna.

Per richiedere il set di risultati in formato JSON, passi il parametro opzionale`formatRecordsAs` con il valore di `JSON`. Il set di risultati in formato JSON viene restituito nel campo `formattedRecords` della struttura `ExecuteStatementResponse`.

L'operazione `BatchExecuteStatement` non restituisce un set di risultati. Pertanto, l'opzione JSON non si applica a tale operazione.

Per personalizzare le chiavi nella struttura hash JSON, definisci gli alias di colonna nel set di risultati. Puoi farlo utilizzando la clausola `AS` nell'elenco delle colonne della query SQL.

Puoi utilizzare la funzionalità JSON per semplificare la lettura del set di risultati e mappare il suo contenuto su framework specifici del linguaggio. Poiché il volume del set di risultati con codifica ASCII è maggiore della rappresentazione di default, puoi scegliere la rappresentazione di default per le query che restituiscono un numero elevato di righe o contenuti di colonne di grandi dimensioni che consumano più memoria di quella disponibile per l'applicazione.

Argomenti

- Recupero dei risultati delle query in formato JSON
- Mappatura dei tipi di dati
- Risoluzione dei problemi
- Esempi

Recupero dei risultati delle query in formato JSON

Per ricevere il set di risultati come stringa JSON, includetelo `.withFormatRecordsAs(RecordsFormatType.JSON)` nella `ExecuteStatement` chiamata. Il valore restituito è ritornato come stringa JSON nel campo `formattedRecords`. In questo caso, il valore di `columnMetadata` è `null`. Le etichette di colonna sono le chiavi dell'oggetto che rappresenta ogni riga. Questi nomi di colonna vengono ripetuti per ogni riga del set dei risultati. I valori delle colonne sono stringhe con virgolette, valori numerici o valori speciali che rappresentano `true`, `false` o `null`. I metadati delle colonne come i vincoli di lunghezza e il tipo specifico di numeri e stringhe non vengono conservati nella risposta JSON.

Se ometti la chiamata `.withFormatRecordsAs()` o specifichi un parametro di `NONE`, il set di risultati viene restituito in formato binario utilizzando i campi `Records` e `columnMetadata`.

Mappatura dei tipi di dati

I valori SQL nel set di risultati sono mappati su un set più piccolo di tipi JSON. I valori sono rappresentati in JSON come stringhe, numeri e alcune costanti speciali come `true`, `false` e `null`. Puoi convertire questi valori in variabili nell'applicazione, utilizzando una tipizzazione forte o debole secondo quanto previsto per il linguaggio di programmazione.

Tipo di dati JDBC	Tipo di dati JSON
`INTEGER`, `TINYINT`, `SMALLINT`, `BIGINT`	Numero di default. Stringa se l'opzione `LongReturnType` è impostata su `STRING`.
`FLOAT`, `REAL`, `DOUBLE`	Numero
`DECIMAL`	Stringa di default. Numero se l'opzione `DecimalReturnType` è impostata su `DOUBLE_OR_LONG` .

Tipo di dati JDBC	Tipo di dati JSON
`STRING`	Stringa
`BOOLEAN`, `BIT`	Booleano
`BLOB`, `BINARY`, `VARBINARY` , `LONGVARBINARY`	Stringa nella codifica base64.
`CLOB`	Stringa
`ARRAY`	Array
`NULL`	`null`
Altri tipi (inclusi i tipi correlati a data e ora)	Stringa

Risoluzione dei problemi

La risposta JSON è limitata a 10 megabyte. Se la risposta è superiore a questo limite, il programma riceve un errore `BadRequestException`. In questo caso, è possibile risolvere l'errore utilizzando una delle seguenti tecniche:

- Ridurre il numero di righe nel set dei risultati. A tale scopo, aggiungi una clausola `LIMIT`. Puoi dividere un set di risultati di grandi dimensioni in blocchi più piccoli inviando diverse query con clausole `LIMIT` e `OFFSET`.

  Se il set di risultati include righe filtrate in base alla logica dell'applicazione, puoi rimuovere tali righe dal set di risultati aggiungendo ulteriori condizioni nella clausola `WHERE`.
- Ridurre il numero di colonne nel set dei risultati. A tale scopo, rimuovi gli elementi dall'elenco di selezione della query.
- Abbrevia le etichette delle colonne utilizzando gli alias di colonna nella query. Il nome di ogni colonna viene ripetuto nella stringa JSON per ogni riga del set di risultati. Pertanto, il risultato di una query con nomi di colonna lunghi e molte righe potrebbe superare il limite di dimensione. In particolare, utilizza alias di colonna per espressioni complicate al fine di evitare di ripetere l'intera espressione nella stringa JSON.
- Sebbene con SQL sia possibile utilizzare alias di colonna per generare un set di risultati con più di una colonna con lo stesso nome, i nomi di chiave duplicati non sono consentiti in JSON. La Data

API di RDS restituisce un errore se si richiede il set di risultati in formato JSON e più di una colonna ha lo stesso nome. Assicurati quindi che tutte le etichette delle colonne abbiano nomi univoci.

Esempi

I seguenti esempi Java mostrano come invocare `ExecuteStatement` richiedendo la risposta sotto forma di stringa in formato JSON e quindi come interpretare il set di risultati. Sostituite i valori appropriati per i parametri *`databaseNamesecretStoreArn`*, e*`clusterArn`*.

Nell'esempio Java seguente viene illustrata una query che restituisce un valore numerico decimale nel set di risultati. Le invocazioni di `assertThat` verificano che i campi della risposta presentino le proprietà attese in base alle regole per i set di risultati in formato JSON.

Questo esempio funziona con lo schema e i dati di esempio seguenti:

```
create table test_simplified_json (a float);
insert into test_simplified_json values(10.0);
```

```
public void JSON_result_set_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request);
}
```

Il valore del campo `formattedRecords` del programma precedente è:

```
[{"a":10.0}]
```

I campi `Records` e `ColumnMetadata` nella risposta sono entrambi nulli, a causa della presenza del set di risultati JSON.

Nell'esempio Java seguente viene illustrata una query che restituisce un valore numerico decimale nel set di risultati. L'esempio invoca `getFormattedRecords` per restituire solo la stringa in

formato JSON e ignorare gli altri campi di risposta che sono vuoti o nulli. L'esempio deserializza il risultato in una struttura che rappresenta un elenco di record. Ogni record presenta campi i cui nomi corrispondono agli alias di colonna del set di risultati. Questa tecnica semplifica il codice che analizza il set di risultati. L'applicazione non deve eseguire il ciclo tra le righe e le colonne del set di risultati e convertire ogni valore nel tipo appropriato.

Questo esempio funziona con lo schema e i dati di esempio seguenti:

```
create table test_simplified_json (a int);
insert into test_simplified_json values(17);
```

```
public void JSON_deserialization_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request)
      .getFormattedRecords();

/* Turn the result set into a Java object, a list of records.
   Each record has a field 'a' corresponding to the column
   labelled 'a' in the result set. */
    private static class Record { public int a; }
    var recordsList = new ObjectMapper().readValue(
        response, new TypeReference<List<Record>>() {
        });
}
```

Il valore del campo `formattedRecords` del programma precedente è:

```
[{"a":17}]
```

Per recuperare la colonna a della riga dei risultati 0, l'applicazione fa riferimento a `recordsList.get(0).a`.

Al contrario, l'esempio Java seguente mostra il tipo di codice necessario per costruire una struttura dati che contiene il set di risultati quando non si utilizza il formato JSON. In questo caso, ogni riga

del set di risultati contiene campi con informazioni su un singolo utente. La creazione di una struttura dati per rappresentare il set di risultati richiede il ciclo tra le righe. Per ogni riga, il codice recupera il valore di ciascun campo, esegue una conversione di tipo appropriata e assegna il risultato al campo corrispondente dell'oggetto che rappresenta la riga. Quindi il codice aggiunge l'oggetto che rappresenta ogni utente alla struttura dati che rappresenta l'intero set di risultati. Se la query viene modificata per riordinare, aggiungere o rimuovere campi nel set di risultati, anche il codice dell'applicazione dovrebbe essere modificato.

```
/* Verbose result-parsing code that doesn't use the JSON result set format */
for (var row: response.getRecords()) {
    var user = User.builder()
      .userId(row.get(0).getLongValue())
      .firstName(row.get(1).getStringValue())
      .lastName(row.get(2).getStringValue())
      .dob(Instant.parse(row.get(3).getStringValue()))
      .build();
    result.add(user);
  }
```

I seguenti valori di esempio mostrano i valori del campo `formattedRecords` per set di risultati con diversi numeri di colonne, alias di colonna e tipi di dati di colonna.

Se il set di risultati include più righe, ogni riga viene rappresentata come un oggetto che è un elemento di una matrice. Ogni colonna del set di risultati diventa una chiave nell'oggetto. Le chiavi sono ripetute per ogni riga del set dei risultati. Pertanto, per i set di risultati costituiti da molte righe e colonne, potrebbe essere necessario definire alias di colonna brevi per evitare di superare il limite di lunghezza dell'intera risposta.

Questo esempio funziona con lo schema e i dati di esempio seguenti:

```
create table sample_names (id int, name varchar(128));
insert into sample_names values (0, "Jane"), (1, "Mohan"), (2, "Maria"), (3, "Bruce"),
 (4, "Jasmine");
```

```
[{"id":0,"name":"Jane"},{"id":1,"name":"Mohan"},
{"id":2,"name":"Maria"},{"id":3,"name":"Bruce"},{"id":4,"name":"Jasmine"}]
```

Se una colonna del set di risultati è definita come espressione, il testo dell'espressione diventa la chiave JSON. Pertanto, in genere è conveniente definire un alias di colonna descrittivo per ogni

espressione nell'elenco di selezione della query. Ad esempio, la seguente query include espressioni come chiamate di funzione e operazioni aritmetiche nell'elenco di selezione.

```
select count(*), max(id), 4+7 from sample_names;
```

Tali espressioni vengono passate al set di risultati JSON come chiavi.

```
[{"count(*)":5,"max(id)":4,"4+7":11}]
```

L'aggiunta di colonne AS con etichette descrittive rende le chiavi più semplici da interpretare nel set di risultati JSON.

```
select count(*) as rows, max(id) as largest_id, 4+7 as addition_result from
 sample_names;
```

Con la query SQL revisionata, le etichette di colonna definite dalle clausole AS vengono utilizzate come nomi chiave.

```
[{"rows":5,"largest_id":4,"addition_result":11}]
```

Il valore di ogni coppia chiave-valore nella stringa JSON può essere una stringa con virgolette. La stringa potrebbe contenere caratteri unicode. Se la stringa contiene sequenze di escape o i caratteri " o \, tali caratteri sono preceduti da caratteri backslash di escape. I seguenti esempi di stringhe JSON dimostrano queste possibilità. Ad esempio, il risultato `string_with_escape_sequences` contiene i caratteri speciali backspace, nuova riga, ritorno a capo, tabulazione, form feed e \.

```
[{"quoted_string":"hello"}]
[{"unicode_string":"####"}]
[{"string_with_escape_sequences":"\b \n \r \t \f \\ '"}]
```

Il valore di ogni coppia chiave-valore nella stringa JSON può rappresentare anche un numero. Il numero potrebbe essere un numero intero, un valore a virgola mobile, un valore negativo o un valore rappresentato come notazione esponenziale. I seguenti esempi di stringhe JSON dimostrano queste possibilità.

```
[{"integer_value":17}]
[{"float_value":10.0}]
[{"negative_value":-9223372036854775808,"positive_value":9223372036854775807}]
```

```
[{"very_small_floating_point_value":4.9E-324,"very_large_floating_point_value":1.79769313486231
```

I valori booleani e null sono rappresentati con le parole chiave speciali senza virgolette `true`,`false` e `null`. I seguenti esempi di stringhe JSON dimostrano queste possibilità.

```
[{"boolean_value_1":true,"boolean_value_2":false}]
[{"unknown_value":null}]
```

Se selezioni un valore di tipo BLOB, il risultato viene rappresentato nella stringa JSON come valore codificato in base64. Per riconvertire il valore nella sua rappresentazione originale puoi utilizzare la funzione di decodifica appropriata nella lingua dell'applicazione. Ad esempio, in Java invochi la funzione `Base64.getDecoder().decode()`. Il seguente output di esempio mostra il risultato della selezione di un valore BLOB di `hello world` e della restituzione del set di risultati come stringa in formato JSON.

```
[{"blob_column":"aGVsbG8gd29ybGQ="}]
```

L'esempio Python seguente mostra come accedere ai valori dal risultato di una chiamata alla funzione Python `execute_statement`. Il set di risultati è un valore stringa nel campo `response['formattedRecords']`. Il codice trasforma la stringa JSON in una struttura dati invocando la funzione `json.loads`. Quindi ogni riga del set di risultati è un elemento elenco all'interno della struttura dati e all'interno di ogni riga puoi fare riferimento a ciascun campo del set di risultati usandone il nome.

```
import json

result = json.loads(response['formattedRecords'])
print (result[0]["id"])
```

L' JavaScript esempio seguente mostra come accedere ai valori del risultato di una chiamata alla JavaScript `executeStatement` funzione. Il set di risultati è un valore stringa nel campo `response.formattedRecords`. Il codice trasforma la stringa JSON in una struttura dati invocando la funzione `JSON.parse`. Quindi ogni riga del set di risultati è un elemento vettore all'interno della struttura dati e all'interno di ogni riga puoi fare riferimento a ciascun campo del set di risultati usandone il nome.

```
<script>
    const result = JSON.parse(response.formattedRecords);
```

```
    document.getElementById("display").innerHTML = result[0].id;
</script>
```

Risoluzione dei problemi relativi all'API dati Amazon RDS

Utilizza le seguenti sezioni, intitolate con messaggi di errore comuni, per risolvere i problemi riscontrati con l'Amazon RDS Data API (Data API).

Argomenti

- <transaction_ID>La transazione non è stata trovata
- Il pacchetto per la query è troppo grande
- La risposta del database è andata oltre il limite delle dimensioni
- HttpEndpointnon è abilitato per il cluster <cluster_ID>
- DatabaseErrorException: Transaction sta ancora eseguendo una query
- Eccezione di risultato non supportata
- Le istruzioni multiple non sono supportate
- Il parametro Schema non è supportato

<transaction_ID>La transazione non è stata trovata

In questo caso, l'ID transazione specificato in una chiamata API dati non è stato trovato. Al messaggio di errore viene aggiunta la causa di questo problema, ovvero una delle seguenti:

- La transazione potrebbe essere scaduta.

  Assicurati che ogni chiamata della transazione venga eseguita entro tre minuti dalla precedente.

  È anche possibile che l'ID di transazione specificato non sia stato creato da una BeginTransactionchiamata. Assicurati che la chiamata abbia un ID transazione valido.
- Una chiamata precedente ha comportato l'interruzione della transazione.

  La transazione era già terminata dalla chiamata `CommitTransaction` o `RollbackTransaction`.
- La transazione è stata interrotta a causa di un errore di una chiamata precedente.

  Verifica se le chiamate precedenti hanno generato eccezioni.

Per informazioni sull'esecuzione di transazioni, consulta Chiamata dell'API dati Amazon RDS.

Il pacchetto per la query è troppo grande

In questo caso, il set di risultati restituito per una riga era troppo grande. Il limite delle dimensioni dell'API dati è 64 KB per riga nel set di risultati restituito dal database.

Per risolvere questo problema, verifica che ogni riga in un set di risultati sia corrispondente o inferiore a 64 KB.

La risposta del database è andata oltre il limite delle dimensioni

In questo caso, le dimensioni del set di risultati restituito dal database erano troppo grandi. Il limite dell'API dati è 1 MiB nel set di risultati restituito dal database.

Per risolvere questo problema, assicurati che le chiamate a Data API restituiscano 1 MiB di dati o meno. Se devi restituire più di 1 MiB, è possibile utilizzare chiamate `ExecuteStatement` molteplici con la clausola `LIMIT` nella query.

Per ulteriori informazioni sulla clausola `LIMIT`, consulta SELECT Syntax nella documentazione MySQL.

HttpEndpointnon è abilitato per il cluster <cluster_ID>

Controlla le seguenti potenziali cause di questo problema:

- Il cluster Aurora DB non supporta Data API. Per informazioni sui tipi di cluster DB supportati da RDS Data API, consulta. the section called “Disponibilità di regioni e versioni per l'Amazon RDS Data API”
- L'API dei dati non è abilitata per il cluster Aurora DB. Per utilizzare Data API con un cluster Aurora DB, l'API Data deve essere abilitata per il cluster DB. Per informazioni sull'abilitazione dell'API Data, consultaAbilitazione dell'API dati Amazon RDS.
- Il cluster DB è stato rinominato dopo che Data API è stata abilitata. In tal caso, disattiva Data API per quel cluster e riattivala.
- L'ARN specificato non corrisponde esattamente all'ARN del cluster. Verifica che l'ARN restituito da un'altra origine o costruito dalla logica del programma corrisponda esattamente all'ARN del cluster. Ad esempio, assicurati che l'ARN utilizzato presenti il corretto formato maiuscolo/minuscolo per tutti i caratteri alfabetici.

DatabaseErrorException: Transaction sta ancora eseguendo una query

Se l'applicazione invia una richiesta con un ID di transazione e tale transazione sta attualmente elaborando un'altra richiesta, Data API restituisce immediatamente questo errore all'applicazione. Questa condizione potrebbe verificarsi se l'applicazione effettua richieste asincrone, utilizzando un meccanismo come le «promesse» in Javascript.

Per risolvere questo problema, attendi il completamento della richiesta precedente, quindi riprova. Puoi continuare a riprovare finché l'errore non si verifica più o l'applicazione non riceve un tipo di errore diverso.

Questa condizione può verificarsi con Data API for Aurora Serverless v2 e con le istanze a cui è stato eseguito il provisioning. In Data API forAurora Serverless v1, le richieste successive per lo stesso ID di transazione attendono automaticamente il completamento della richiesta precedente. Tuttavia, questo comportamento precedente potrebbe potenzialmente comportare dei timeout dovuti al fatto che la richiesta precedente richiedeva troppo tempo. Se stai eseguendo il porting di un'applicazione Data API precedente che effettua richieste simultanee, modifica la logica di gestione delle eccezioni per tenere conto di questo nuovo tipo di errore.

Eccezione di risultato non supportata

L'API Data non supporta tutti i tipi di dati. Questo errore si verifica quando si esegue una query che restituisce un tipo di dati non supportato.

Per risolvere questo problema, trasmetti il tipo di dati non supportato a. `TEXT` Per esempio:

```
SELECT custom_type::TEXT FROM my_table;
-- OR
SELECT CAST(custom_type AS TEXT) FROM my_table;
```

Le istruzioni multiple non sono supportate

Le istruzioni multiple non sono supportate nell'API Data per Aurora Serverless v2 e nei cluster con provisioning. Il tentativo di eseguire più istruzioni in una singola chiamata API genera questo errore.

Per eseguire più istruzioni, utilizzate chiamate `ExecuteStatement` API separate o utilizzate l'`BatchExecuteStatement`API per l'elaborazione in batch.

Il parametro Schema non è supportato

Aurora Serverless v1 ignora silenziosamente il parametro dello schema. Tuttavia, Aurora Serverless v2 e i cluster provisioned rifiutano esplicitamente le chiamate API che includono il parametro schema.

Per risolvere questo problema, rimuovi il parametro schema da tutte le chiamate all'API Data quando usi Aurora Serverless v2 o cluster con provisioning.

Registrazione delle chiamate Amazon RDS Data API con AWS CloudTrail

RDS Data API (Data API) è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o AWS servizio in Data API. CloudTrail acquisisce tutte le chiamate API per Data API come eventi, incluse le chiamate dalla console Amazon RDS e dalle chiamate di codice alle operazioni Data API. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Data API. Utilizzando i dati raccolti da CloudTrail, puoi determinare molte informazioni. Queste informazioni includono la richiesta che è stata fatta all'API dati, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata fatta e ulteriori dettagli.

Per ulteriori informazioni CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

Utilizzo delle informazioni dell'API dati in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività supportata (eventi di gestione) in Data API, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi di gestione recenti nel tuo AWS account. Per ulteriori informazioni, consulta Lavorare con la cronologia degli CloudTrail eventi nella Guida AWS CloudTrail per l'utente.

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Data API, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutte le AWS regioni. Il trail registra gli eventi di tutte le AWS regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS CloudTrail :

- Panoramica della creazione di un percorso
- CloudTrail servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte le operazioni Data API vengono registrate CloudTrail e documentate nel riferimento dell'API del servizio dati Amazon RDS. Ad esempio, le chiamate alle `ExecuteStatement` operazioni`BatchExecuteStatement`, `BeginTransaction``CommitTransaction`, e generano voci nei file di registro. CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Inclusione ed esclusione degli eventi Data API da un AWS CloudTrail percorso

La maggior parte degli utenti di Data API si affida agli eventi di un AWS CloudTrail trail per registrare le operazioni di Data API. I dati degli eventi non rivelano il nome del database, il nome dello schema o le istruzioni SQL nelle richieste all'API Data. Tuttavia, sapere quale utente ha effettuato un tipo di chiamata su uno specifico cluster DB in un determinato momento può aiutare a rilevare modelli di accesso anomali.

Inclusione degli eventi Data API in un percorso AWS CloudTrail

Per i database Aurora PostgreSQL Serverless v2 e con provisioning, le seguenti operazioni Data API vengono registrate come eventi di dati. AWS CloudTrail Gli eventi relativi ai dati sono operazioni API data-plane ad alto volume che non vengono registrate per impostazione predefinita. CloudTrail Per gli eventi di dati sono previsti costi aggiuntivi. Per informazioni sui CloudTrail prezzi, consulta Prezzi.AWS CloudTrail

- BatchExecuteStatement
- BeginTransaction
- CommitTransaction
- ExecuteStatement
- RollbackTransaction

Puoi utilizzare la CloudTrail console o AWS CLI le operazioni CloudTrail API per registrare queste operazioni Data API. Nella CloudTrail console, scegli RDS Data API - DB Cluster per il tipo di evento Data. Per ulteriori informazioni, consulta Registrazione degli eventi relativi ai dati con la AWS Management Console nella Guida per l'AWS CloudTrail utente.

Utilizzando AWS CLI, esegui il `aws cloudtrail put-event-selectors` comando per registrare queste operazioni dell'API Data per il tuo percorso. Per registrare tutti gli eventi Data API sui cluster DB, specifica `AWS::RDS::DBCluster` il tipo di risorsa. L'esempio seguente registra tutti gli eventi Data API sui cluster DB. Per ulteriori informazioni, consulta Registrazione degli eventi relativi ai dati con la AWS Command Line Interface nella Guida per l'utente.AWS CloudTrail

```
aws cloudtrail put-event-selectors --trail-name trail_name --advanced-event-selectors \
'{
   "Name": "RDS Data API Selector",
   "FieldSelectors": [
      {
         "Field": "eventCategory",
         "Equals": [
            "Data"
         ]
      },
      {
         "Field": "resources.type",
         "Equals": [
            "AWS::RDS::DBCluster"
         ]
      }
   ]
}'
```

È possibile configurare selettori di eventi avanzati per filtrare ulteriormente i campi `readOnly` `eventName`, `eresources.ARN`. Per ulteriori informazioni su questi campi, consultare AdvancedFieldSelector.

Esclusione degli eventi Data API da un AWS CloudTrail trail (Aurora Serverless v1solo)

InfattiAurora Serverless v1, gli eventi Data API sono eventi di gestione. Per impostazione predefinita, tutti gli eventi Data API sono inclusi in un AWS CloudTrail percorso. Tuttavia, poiché Data API può generare un gran numero di eventi, potresti voler escludere questi eventi dal tuo CloudTrail percorso. L'impostazione Exclude Amazon RDS Data API events esclude tutti gli eventi Data API dal trail. Non puoi escludere eventi Data API specifici.

Per escludere gli eventi API dati da un percorso, procedi nel seguente modo:

- Nella CloudTrail console, scegli l'impostazione Exclude Amazon RDS Data API events quando crei un trail o aggiorni un trail.
- Nell' CloudTrail API, usa l'PutEventSelectorsoperazione. Se utilizzi selettori di eventi avanzati, puoi escludere gli eventi Data API impostando il `eventSource` campo diverso `dardsdata.amazonaws.com`. Se utilizzi selettori di eventi di base, puoi escludere gli eventi Data API impostando il valore dell'`ExcludeManagementEventSources`attributo su. `rdsdata.amazonaws.com` Per ulteriori informazioni, consulta Registrazione degli eventi con la AWS Command Line Interface nella Guida per l'AWS CloudTrail utente.

> **Warning**
>
> L'esclusione degli eventi Data API da un CloudTrail registro può oscurare le azioni di Data API. Presta attenzione quando concedi alle entità principali l'autorizzazione `cloudtrail:PutEventSelectors` necessaria per eseguire questa operazione.

È possibile disattivare questa esclusione in qualsiasi momento modificando l'impostazione della console o i selettori di eventi per un percorso. Il percorso inizierà quindi a registrare gli eventi API dati. Tuttavia, non è possibile ripristinare gli eventi API dati che si sono verificati mentre era in atto l'esclusione.

Quando escludi gli eventi Data API utilizzando la console o l'API, anche l'operazione CloudTrail `PutEventSelectors` API risultante viene registrata nei tuoi CloudTrail log. Se gli eventi Data API non vengono visualizzati nei CloudTrail log, cerca un `PutEventSelectors` evento con l'`ExcludeManagementEventSources`attributo impostato su. `rdsdata.amazonaws.com`

Per ulteriori informazioni, consulta Registrazione di eventi di gestione per i percorsi nella Guida per l'utente di AWS CloudTrail .

Informazioni sulle voci dei file di registro dell'API dei dati

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

Aurora PostgreSQL Serverless v2 e provisioning

L'esempio seguente mostra una voce di CloudTrail registro che dimostra il `ExecuteStatement` funzionamento per Aurora PostgreSQL Serverless v2 e i database con provisioning. Per questi database, tutti gli eventi Data API sono eventi di dati in cui l'origine dell'evento è rdsdataapi.amazonaws.com e il tipo di evento è Rds Data Service.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdataapi.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
```

```
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "Rds Data Service",
    "recipientAccountId": "123456789012"
}
```

Aurora Serverless v1

L'esempio seguente mostra come viene visualizzata la voce di registro dell'esempio precedente per. CloudTrail Aurora Serverless v1 InfattiAurora Serverless v1, tutti gli eventi sono eventi di gestione la cui origine è rdsdata.amazonaws.com e il tipo di evento è. AwsApiCall

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdata.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
```

```
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Monitoraggio delle query RDS Data API con Performance Insights

Se il cluster Aurora è in esecuzione Aurora Serverless v2 o dispone di istanze, puoi utilizzare Performance Insights with RDS Data API.

Per ulteriori informazioni su come utilizzare Performance Insights con Aurora, consulta. Monitoraggio del carico DB con Performance Insights su Amazon Aurora

Come vengono rappresentate le query RDS Data API in Performance Insights

Con Data API, il cluster Aurora elabora le query in base alle chiamate Data API inviate dall'applicazione. Data API esegue anche alcune istruzioni SQL come parte del proprio funzionamento interno, come l'annullamento di query che superano la soglia di timeout. Entrambi i tipi di operazioni SQL sono mostrati nelle statistiche e nei grafici di Performance Insights.

- Per le query Data API inviate a un cluster Aurora, il campo Host nel dashboard PI è contrassegnato come RDS Data API. Per Aurora PostgreSQL, il campo application_name ha il valore. `rds-data-api` Cerca queste etichette quando analizzi il carico del database utilizzando Top hosts o Top Applications come dimensione.
- Tutte le query interne eseguite da Data API per gestire aspetti del database, come il pool di connessioni e i timeout delle query, sono annotate con un prefisso RDS Data API. Esempio: `/* RDS Data API */ select * from my_table;` cerca questi prefissi quando si analizza il caricamento del database tramite Top SQL come dimensione. Le istruzioni sono annotate con un commento SQL di. `/* RDS Data API */`

Utilizzo dell'editor di query Aurora

L'editor di query Aurora consente di eseguire istruzioni SQL sul cluster Aurora DB tramite. AWS Management ConsoleÈ possibile eseguire query SQL, istruzioni di manipolazione dei dati (DML) e istruzioni di definizione dei dati (DDL). L'utilizzo dell'interfaccia della console consente di eseguire la manutenzione del database, produrre report e condurre esperimenti SQL. Puoi evitare di configurare la configurazione di rete per connetterti al tuo cluster DB da un sistema client separato, come un'istanza EC2 o un computer portatile.

L'editor di query richiede un cluster Aurora DB con RDS Data API (Data API) abilitata. Per informazioni sui cluster DB che supportano Data API e su come abilitarla, consulta. Utilizzo dell'API dati Amazon RDS L'SQL che puoi eseguire è soggetto alle limitazioni dell'API Data. Per ulteriori informazioni, consulta the section called “Limitazioni”.

Disponibilità dell'editor di query

L'editor di query è disponibile per i cluster Aurora DB che utilizzano le versioni dei motori Aurora MySQL e Aurora PostgreSQL che supportano Data API e dove Data API è disponibile. Regioni AWS Per ulteriori informazioni, consulta Regioni supportate e motori Aurora DB per RDS Data API.

Autorizzazione di accesso all'editor della query

Un utente deve essere autorizzato a eseguire query nell'editor della query. È possibile autorizzare un utente a eseguire query nell'editor di query aggiungendo la policy, una `AmazonRDSDataFullAccess` policy predefinita (IAM), a quell'utente. AWS Identity and Access Management

> **Note**
>
> Assicurati di utilizzare lo stesso nome utente e la stessa password quando crei l'utente IAM che hai usato per l'utente del database, ad esempio il nome utente e la password amministrativi. Per ulteriori informazioni, consulta Creazione di un utente IAM nell' Account AWS nella Guida per l'utente di AWS Identity and Access Management .

Puoi anche creare una policy IAM che concede l'accesso all'editor della query. Dopo aver creato la policy, aggiungila a ciascun utente che richiede l'accesso all'editor della query.

La seguente policy fornisce le autorizzazioni minime richieste per un utente per accedere all'editor della query.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "QueryEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutResourcePolicy",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:TagResource"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "QueryEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "tag:GetResources",
                "secretsmanager:CreateSecret",
                "secretsmanager:ListSecrets",
                "dbqms:CreateFavoriteQuery",
                "dbqms:DescribeFavoriteQueries",
                "dbqms:UpdateFavoriteQuery",
                "dbqms:DeleteFavoriteQueries",
                "dbqms:GetQueryString",
                "dbqms:CreateQueryHistory",
                "dbqms:UpdateQueryHistory",
                "dbqms:DeleteQueryHistory",
                "dbqms:DescribeQueryHistory",
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Per informazioni sulla creazione di una policy IAM, consulta Creazione di policy IAM nella Guida per l'utente di AWS Identity and Access Management .

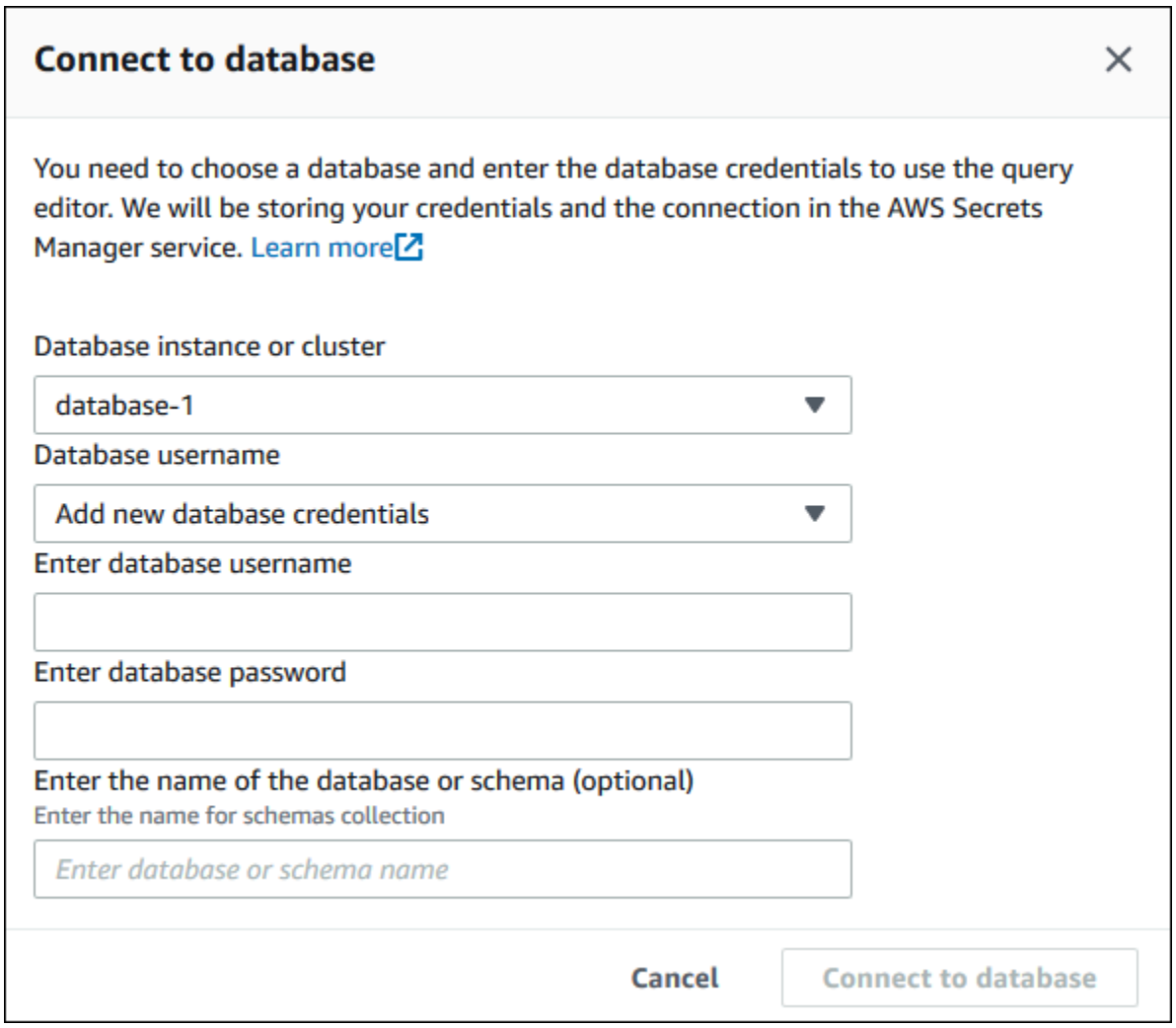
Per informazioni sull'aggiunta di una policy IAM a un utente, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di AWS Identity and Access Management .

Esecuzione di query nell'editor della query

È possibile eseguire istruzioni SQL su un cluster Aurora DB nell'editor di query. L'SQL che è possibile eseguire è soggetto alle limitazioni dell'API Data. Per ulteriori informazioni, consulta the section called “Limitazioni”.

Per eseguire una query nell'editor della query

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'angolo superiore destro di AWS Management Console, scegli il cluster Regione AWS in cui hai creato i cluster Aurora DB su cui desideri interrogare.
3. Nel riquadro di navigazione, scegli Databases (Database).
4. Scegli il cluster Aurora DB su cui eseguire le query SQL.
5. In Actions (Operazioni), scegliere Query. Se è la prima volta che si esegue la connessione al database, viene visualizzata la pagina Connect to database (Connetti al database).



6. Immetti le seguenti informazioni:

   a. Per Istanza o cluster di database, scegli il cluster Aurora DB su cui desideri eseguire le query SQL.

   b. In Database username (Nome utente del database), scegliere il nome utente dell'utente del database a cui connettersi o selezionare Add new database credentials (Aggiungi nuove credenziali del database. Se si sceglie Add new database credentials (Aggiungi nuove credenziali del database), immettere il nome utente per le nuove credenziali del database in Enter database username (Immetti il nome utente database).

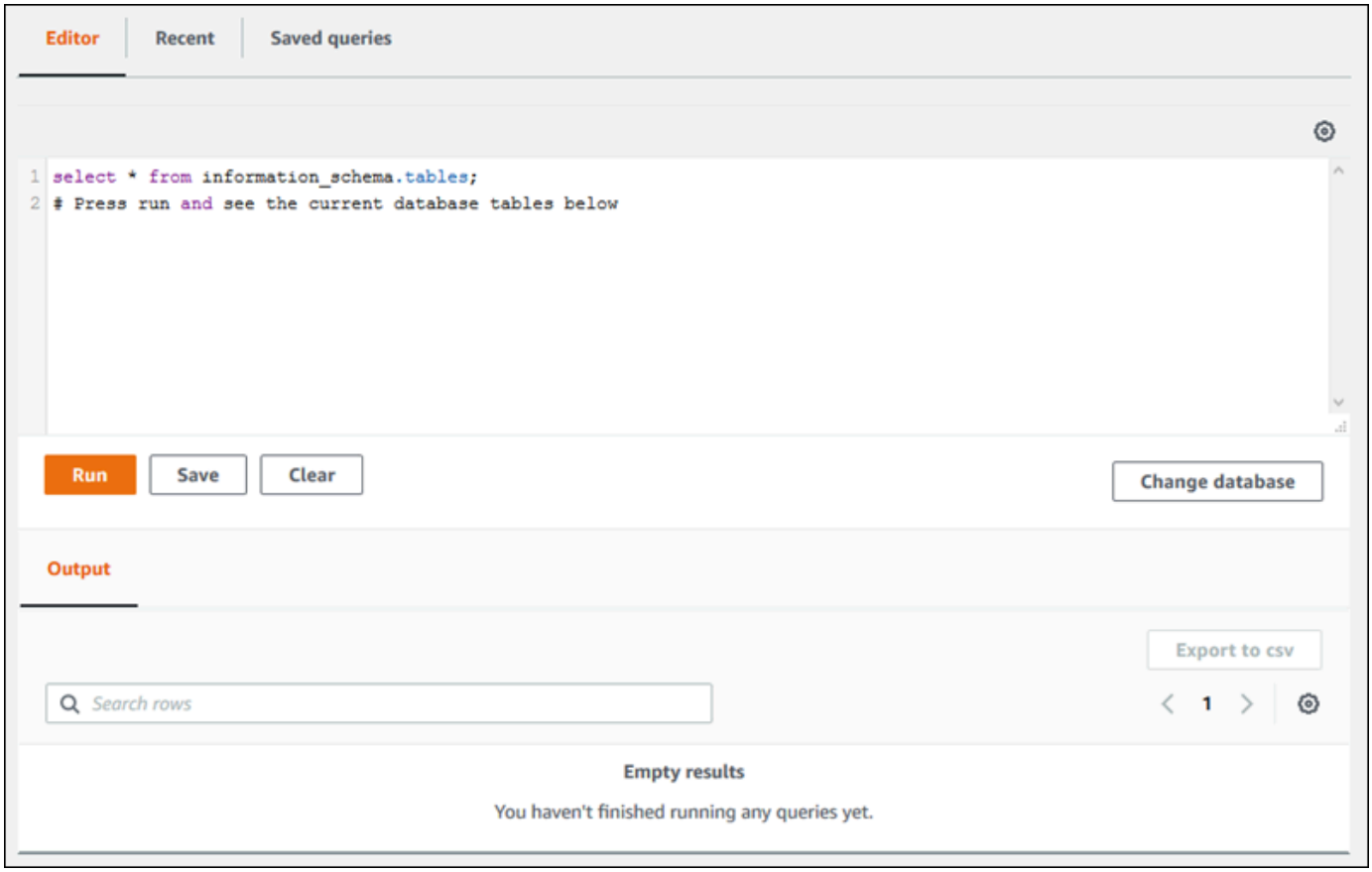
   c. In Enter database password (Immetti la password database), immettere la password per l'utente del database scelto.

   d. Nell'ultima casella, immettere il nome del database o dello schema da utilizzare per il cluster database Aurora.

   e. Scegliere Connect to database (Connetti al database).

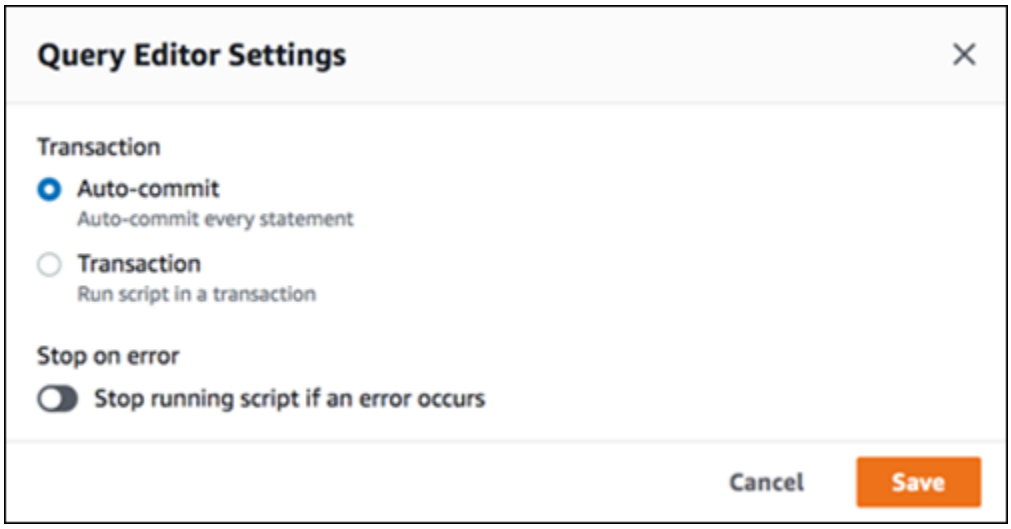
> **Note**
>
> Se la connessione ha esito positivo, le informazioni relative a connessione e autenticazione vengono archiviate in AWS Secrets Manager. Non è necessario inserire nuovamente le informazioni di connessione.

7. Nell'editor della query, immettere la query SQL che si desidera eseguire sul database.



Ogni istruzione SQL è sottoposta automaticamente a commit o è possibile eseguire istruzioni SQL in uno script come parte di una transazione. Per controllare questo comportamento, scegliere l'icona a forma di ingranaggio sopra la finestra di query.

Viene visualizzata la finestra Query Editor Settings (Impostazioni editor della query).



Se si sceglie Auto-commit (Commit automatico), viene eseguito automaticamente il commit di ogni istruzione SQL. Se si sceglie Transazione, è possibile eseguire un gruppo di istruzioni in uno script. Alla fine dello script viene eseguito automaticamente il commit delle istruzioni a meno che prima non sia stato eseguito esplicitamente il commit o il rollback. Inoltre, è possibile scegliere di interrompere uno script in esecuzione se si verifica un errore abilitando Stop on error (Interrompi in caso di errore).

> **Note**
>
> In un gruppo di istruzioni, le istruzioni DDL (Data Definition Language) possono causare il commit di istruzioni DML (Data Manipulation Language) precedenti. È possibile anche includere istruzioni `COMMIT` e `ROLLBACK` in un gruppo di istruzioni in uno script.

Dopo aver effettuato le scelte nella finestra Query Editor Settings (Impostazioni editor della query), scegliere Save (Salva).

8. Scegliere Run (Esegui) o premere Ctrl+Invio. L'editor della query visualizza i risultati della query.

   Dopo aver eseguito la query, salvarla in Saved queries (Query salvate) scegliendo Save As (Salva con nome).

   Esportare i risultati della query in formato foglio di calcolo scegliendo Export to csv (Esporta in formato .csv).

Puoi trovare, modificare ed eseguire nuovamente query precedenti. A tale scopo, seleziona la scheda Recent (Recente) o Saved queries (Query salvate), scegli il testo della query, quindi seleziona Run (Esegui).

Per cambiare il database, scegli Change database (Cambia database).

Riferimento al Database Query Metadata Service (DBQMS) API

Il Database Query Metadata Service (`dbqms`) è un servizio solo interno. Fornisce le tue query recenti e salvate per l'editor di query su più AWS servizi, incluso AmazonRDS. AWS Management Console

Azioni supportate DBQMS

- CreateFavoriteQuery
- CreateQueryHistory
- CreateTab
- DeleteFavoriteQueries
- DeleteQueryHistory
- DeleteTab

- DescribeFavoriteQueries
- DescribeQueryHistory
- DescribeTabs
- GetQueryString
- UpdateFavoriteQuery
- UpdateQueryHistory
- UpdateTab

CreateFavoriteQuery

Salva una nuova query preferita. Ogni utente può creare fino a 1000 query salvate. Questo limite è soggetto a modifiche in futuro.

CreateQueryHistory

Salva una nuova voce della cronologia delle query.

CreateTab

Salva una nuova scheda della query. Ogni utente può creare fino a 10 schede di query.

DeleteFavoriteQueries

Elimina una o più query salvate.

DeleteQueryHistory

Elimina le voci della cronologia delle query.

DeleteTab

Elimina le voci della scheda di query.

DescribeFavoriteQueries

Elenca le query salvate create da un utente in un determinato account.

DescribeQueryHistory

Elenca le voci della cronologia delle query.

DescribeTabs

Elenca le schede delle query create da un utente in un determinato account.

GetQueryString

Recupera il testo completo della query da un ID di query.

UpdateFavoriteQuery

Aggiorna la stringa di query, la descrizione, il nome o la data di scadenza.

UpdateQueryHistory

Aggiorna lo stato della cronologia delle query.

UpdateTab

Aggiorna il nome della scheda di query e la stringa di query.

Utilizzo dell'apprendimento automatico di Amazon Aurora

Utilizzando l'apprendimento automatico di Amazon Aurora, puoi integrare il tuo cluster Aurora DB con uno dei seguenti servizi di AWS machine learning, a seconda delle tue esigenze. Ciascuno di essi supporta casi d'uso specifici di machine learning.

Amazon Bedrock

Amazon Bedrock è un servizio completamente gestito che rende disponibili i principali modelli di base delle aziende di intelligenza artificiale API tramite strumenti per sviluppatori per aiutare a creare e scalare applicazioni di intelligenza artificiale generativa. Con Amazon Bedrock, paghi per eseguire inferenze su qualsiasi modello di fondazione di terze parti. I prezzi si basano sul volume dei token di input e di output, nonché sull'eventuale acquisto o meno della velocità di trasmissione effettiva assegnata per il modello. Per ulteriori informazioni, consulta Che cos'è Amazon Bedrock? nella Guida per l'utente di Amazon Bedrock

Amazon Comprehend

Amazon Comprehend è un servizio gestito di elaborazione del linguaggio naturale (NLP) utilizzato per estrarre informazioni dai documenti. Amazon Comprehend consente di dedurre il sentiment in base al contenuto dei documenti, analizzando entità, frasi chiave, lingua e altre caratteristiche. Per ulteriori informazioni, consultare Che cos'è Amazon Comprehend? nella Guida per gli sviluppatori di Amazon Comprehend.

SageMaker AI

Amazon SageMaker AI è un servizio di machine learning completamente gestito. I data scientist e gli sviluppatori utilizzano Amazon SageMaker AI per creare, addestrare e testare modelli di machine learning per una varietà di attività di inferenza, come il rilevamento delle frodi e la raccomandazione di prodotti. Quando un modello di machine learning è pronto per l'uso in produzione, può essere distribuito nell'ambiente ospitato da Amazon SageMaker AI. Per ulteriori informazioni, consulta What Is Amazon SageMaker AI? nella Amazon SageMaker AI Developer Guide.

L'uso di Amazon Comprehend con il cluster Aurora DB richiede meno configurazioni preliminari rispetto all'utilizzo dell'IA. SageMaker Se non conosci l'apprendimento AWS automatico, ti consigliamo di iniziare esplorando Amazon Comprehend.

Argomenti

- Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL
- Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora Postgre SQL

Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora My SQL

Utilizzando l'apprendimento automatico di Amazon Aurora con il tuo cluster Aurora My SQL DB, puoi usare Amazon Bedrock, Amazon Comprehend o SageMaker Amazon AI, a seconda delle tue esigenze. Ciascuno di essi supporta diversi casi d'uso dell'apprendimento automatico.

Indice

- Requisiti per l'utilizzo dell'apprendimento automatico di Aurora con Aurora My SQL
- Disponibilità di regioni e versioni
- Funzionalità e limitazioni supportate dell'apprendimento automatico di Aurora con Aurora My SQL
- Configurazione del cluster Aurora My SQL DB per utilizzare l'apprendimento automatico Aurora
  - Configurazione del cluster Aurora My SQL DB per utilizzare Amazon Bedrock
  - Configurazione del cluster Aurora My SQL DB per utilizzare Amazon Comprehend
  - Configurazione del cluster Aurora My SQL DB per l'utilizzo dell'IA SageMaker
    - Configurazione del cluster Aurora My SQL DB per l'utilizzo di Amazon S3 SageMaker for AI (opzionale)
  - Concessione agli utenti del database dell'accesso a machine learning di Aurora
    - Concessione dell'accesso alle funzioni di Amazon Bedrock
    - Concessione dell'accesso alle funzioni Amazon Comprehend
    - Concessione dell'accesso alle funzioni AI SageMaker
- Utilizzo di Amazon Bedrock con il cluster Aurora My DB SQL
- Utilizzo di Amazon Comprehend con il cluster Aurora My DB SQL
- Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora My DB SQL
  - Requisito del set di caratteri per le funzioni SageMaker AI che restituiscono stringhe
  - Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)
- Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora My SQL
  - Modello e prompt

  - Cache delle query
  - Ottimizzazione batch per le chiamate di funzione Aurora Machine Learning
- Monitoraggio del machine learning di Aurora

Requisiti per l'utilizzo dell'apprendimento automatico di Aurora con Aurora My SQL

AWS i servizi di apprendimento automatico sono servizi gestiti che vengono configurati ed eseguiti nei propri ambienti di produzione. L'apprendimento automatico Aurora supporta l'integrazione con Amazon Bedrock, Amazon Comprehend e AI. SageMaker Prima di provare a configurare il cluster Aurora My SQL DB per utilizzare l'apprendimento automatico di Aurora, assicurati di comprendere i seguenti requisiti e prerequisiti.

- I servizi di machine learning devono essere eseguiti nello Regione AWS stesso cluster Aurora My SQL DB. Non è possibile utilizzare i servizi di machine learning di un cluster Aurora My SQL DB in un'altra regione.
- Se il cluster Aurora My SQL DB si trova in un cloud pubblico virtuale diverso (VPC) dal servizio Amazon Bedrock, Amazon Comprehend o SageMaker AI, il gruppo Security deve consentire le connessioni in uscita al servizio VPC di machine learning Aurora di destinazione. Per ulteriori informazioni, consulta Controlla il traffico verso AWS le tue risorse utilizzando i gruppi di sicurezza nella Amazon VPC User Guide.
- Puoi aggiornare un cluster Aurora che esegue una versione precedente di Aurora My SQL a una versione superiore supportata se desideri utilizzare l'apprendimento automatico Aurora con quel cluster. Per ulteriori informazioni, consulta Aggiornamenti del motore di database per Amazon Aurora My SQL.
- Il cluster Aurora My SQL DB deve utilizzare un gruppo di parametri del cluster DB personalizzato. Al termine del processo di configurazione per ogni servizio di machine learning Aurora che desideri utilizzare, aggiungi l'Amazon Resource Name (ARN) del IAM ruolo associato che è stato creato per il servizio. Ti consigliamo di creare SQL in anticipo un gruppo di parametri del cluster DB personalizzato per Aurora My DB e di configurare il cluster Aurora My SQL DB per utilizzarlo in modo che sia pronto per essere modificato al termine del processo di configurazione.
- Per SageMaker l'IA:
  - I componenti di machine learning che si desidera utilizzare per le inferenze devono essere configurati e pronti per l'uso. Durante il processo di configurazione per il tuo cluster Aurora My SQL DB, assicurati ARN di avere a disposizione l'endpoint SageMaker AI. I data scientist del tuo

team sono probabilmente i più adatti a gestire il lavoro con l' SageMaker intelligenza artificiale per preparare i modelli e gestire le altre attività simili. Per iniziare a usare Amazon SageMaker AI, consulta Get Started with Amazon SageMaker AI. Per ulteriori informazioni su inferenze ed endpoint, consulta Real-time inference (Inferenza in tempo reale).

- Per utilizzare l' SageMaker intelligenza artificiale con i tuoi dati di addestramento, devi configurare un bucket Amazon S3 come parte della tua configurazione Aurora My SQL per Aurora Machine Learning. A tale scopo, segui la stessa procedura generale utilizzata per la configurazione dell'integrazione AI. SageMaker Per un riepilogo di questo processo di configurazione facoltativo, consultare Configurazione del cluster Aurora My SQL DB per l'utilizzo di Amazon S3 SageMaker for AI (opzionale).

- Per i database globali di Aurora, configuri i servizi di machine learning di Aurora che desideri utilizzare in tutto ciò Regioni AWS che compone il tuo database globale Aurora. Ad esempio, se desideri utilizzare l'apprendimento automatico di Aurora con l' SageMaker intelligenza artificiale per il tuo database globale Aurora, esegui le seguenti operazioni per ogni cluster Aurora My DB in ogni: SQL Regione AWS
  - Configura i servizi Amazon SageMaker AI con gli stessi modelli di formazione e gli stessi endpoint per l' SageMaker intelligenza artificiale. Anche questi devono utilizzare gli stessi nomi.
  - Crea i IAM ruoli come descritto inConfigurazione del cluster Aurora My SQL DB per utilizzare l'apprendimento automatico Aurora.
  - Aggiungi il IAM ruolo al gruppo ARN di parametri del cluster DB personalizzato per ogni cluster Aurora My SQL DB in ogni. Regione AWS

  Queste attività richiedono che l'apprendimento automatico di Aurora sia disponibile per la tua versione di Aurora My SQL in tutti gli elementi Regioni AWS che compongono il tuo database globale di Aurora.

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia tra le versioni specifiche di ciascun motore di database Aurora e tra Regioni AWS.

- Per informazioni sulla disponibilità della versione e della regione per Amazon Comprehend e Amazon SageMaker AI con Aurora My, consulta. SQL Apprendimento automatico Aurora con Aurora My SQL
- Amazon Bedrock è supportato solo su Aurora SQL My versione 3.06 e successive.

Per informazioni sulla disponibilità regionale per Amazon Bedrock, consulta Model support by Regione AWS nella Amazon Bedrock User Guide.

Funzionalità e limitazioni supportate dell'apprendimento automatico di Aurora con Aurora My SQL

Quando si utilizza Aurora My con SQL Aurora Machine Learning, si applicano le seguenti limitazioni:

- L'estensione Aurora per l'apprendimento automatico non supporta le interfacce vettoriali.
- Le integrazioni di machine learning di Aurora non sono supportate se utilizzate in un trigger.
- Le funzioni di machine learning di Aurora non sono compatibili con la replica di registrazione binaria (binlog).
  - L'impostazione `--binlog-format=STATEMENT` genera un'eccezione per le chiamate alle funzioni Aurora machine learning.
  - Le funzioni di machine Learning di Aurora sono non deterministiche e le funzioni archiviate non deterministiche non sono compatibili con il formato binlog.

  Per ulteriori informazioni, consulta Binary Logging Formats nella documentazione personale. SQL
- Le funzioni archiviate che chiamano tabelle con colonne sempre generate non sono supportate. Questo vale per qualsiasi funzione SQL memorizzata di Aurora My. Per ulteriori informazioni su questo tipo di colonna, consulta CREATETABLEe Colonne generate nella SQL documentazione personale.
- Le funzioni di Amazon Bedrock non sono `RETURNS JSON` supportate. Puoi utilizzare `CONVERT` o convertire da `CAST TEXT` a, `JSON` se necessario.
- Amazon Bedrock non supporta le richieste in batch.
- Aurora My SQL supporta qualsiasi endpoint SageMaker AI che legge e scrive il formato value () CSV separato da virgole, tramite un di. `ContentType text/csv` Questo formato è accettato dai seguenti algoritmi di intelligenza artificiale integrati: SageMaker
  - Linear Learner
  - Random Cut Forest
  - XGBoost

  Per ulteriori informazioni su questi algoritmi, consulta Choose an Algorithm nella Amazon SageMaker AI Developer Guide.

Configurazione del cluster Aurora My SQL DB per utilizzare l'apprendimento automatico Aurora

Nei seguenti argomenti sono illustrate le procedure di configurazione separate per ciascuno di questi servizi di machine learning di Aurora.

Argomenti

- Configurazione del cluster Aurora My SQL DB per utilizzare Amazon Bedrock
- Configurazione del cluster Aurora My SQL DB per utilizzare Amazon Comprehend
- Configurazione del cluster Aurora My SQL DB per l'utilizzo dell'IA SageMaker
  - Configurazione del cluster Aurora My SQL DB per l'utilizzo di Amazon S3 SageMaker for AI (opzionale)
- Concessione agli utenti del database dell'accesso a machine learning di Aurora
  - Concessione dell'accesso alle funzioni di Amazon Bedrock
  - Concessione dell'accesso alle funzioni Amazon Comprehend
  - Concessione dell'accesso alle funzioni AI SageMaker

Configurazione del cluster Aurora My SQL DB per utilizzare Amazon Bedrock

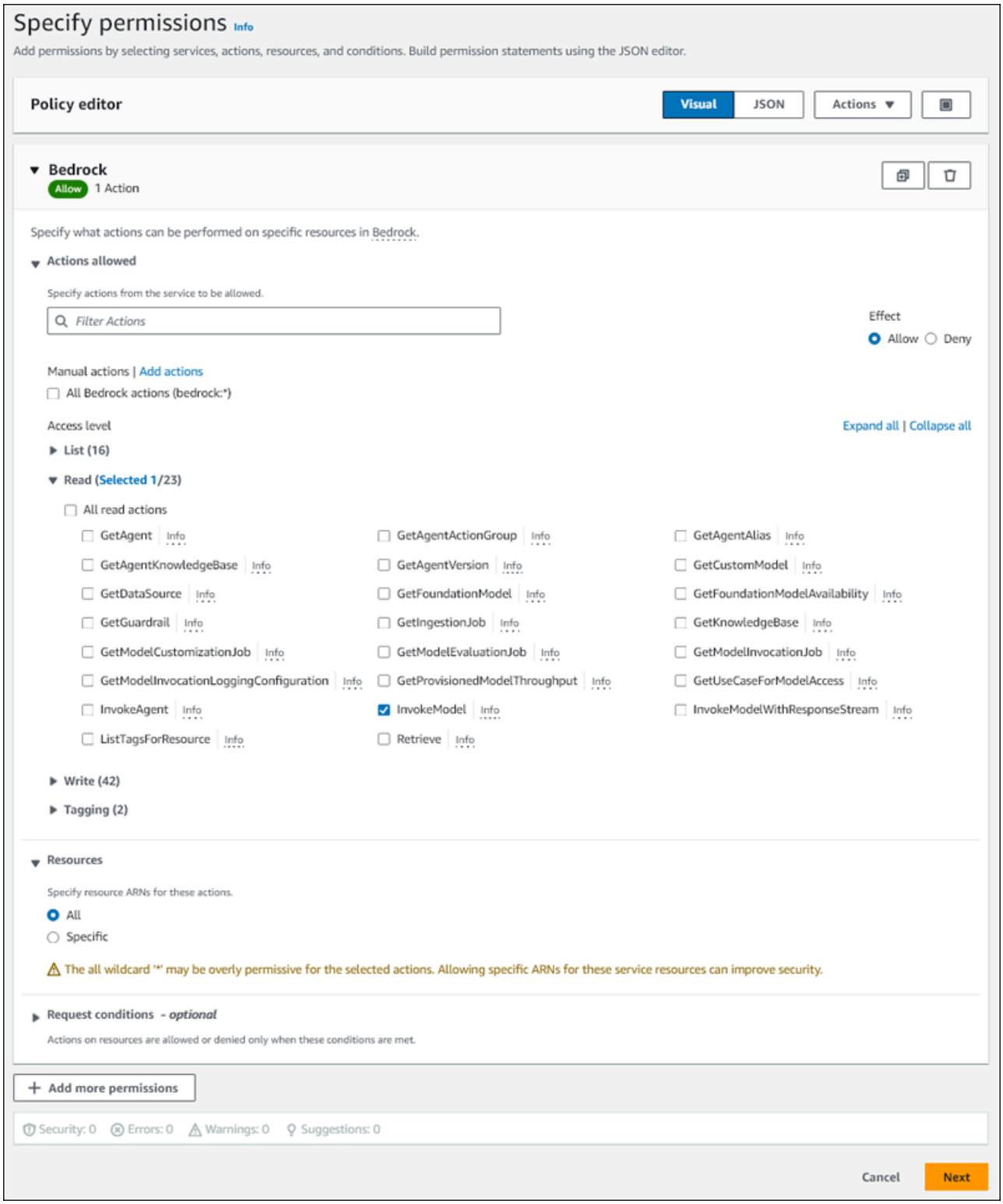
L'apprendimento automatico Aurora si basa su AWS Identity and Access Management (IAM) ruoli e policy per consentire al cluster Aurora My SQL DB di accedere e utilizzare i servizi Amazon Bedrock. Le seguenti procedure creano una politica e un ruolo di IAM autorizzazione in modo che il cluster DB possa integrarsi con Amazon Bedrock.

Per creare la policy IAM

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegli Politiche nel riquadro di navigazione.
3. Scegliere Create a policy (Crea una policy).
4. Nella pagina Specificare le autorizzazioni, per Seleziona un servizio, scegli Bedrock.

   Vengono visualizzate le autorizzazioni di Amazon Bedrock.
5. Espandi Leggi, quindi seleziona. InvokeModel
6. Per Risorse, seleziona Tutto.

La pagina Specificare le autorizzazioni dovrebbe essere simile alla figura seguente.



7. Scegli Next (Successivo).

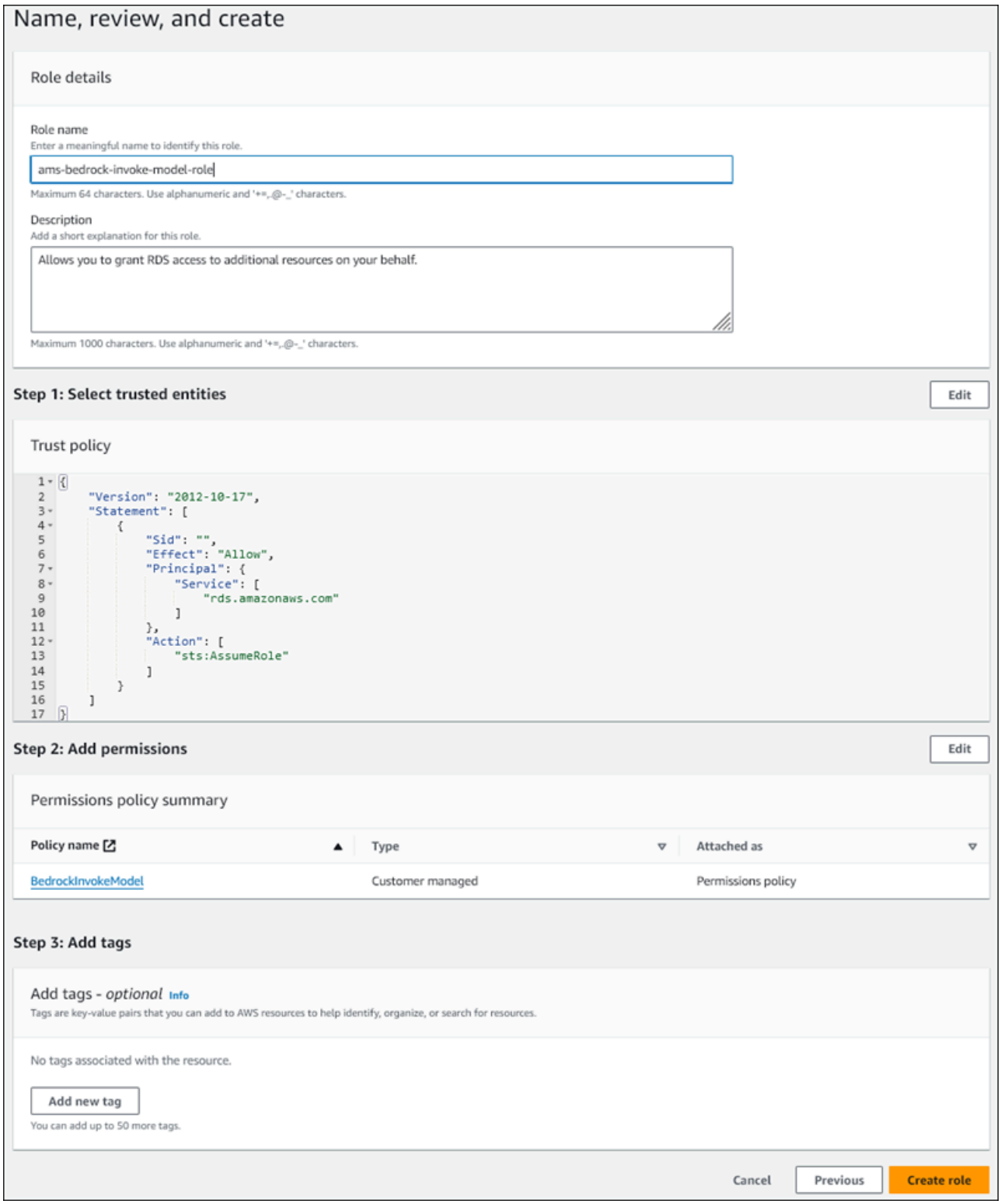
8. Nella pagina Rivedi e crea, inserisci un nome per la tua politica, ad esempio**`BedrockInvokeModel`**.
9. Rivedi la tua politica, quindi scegli Crea politica.

Successivamente crei il IAM ruolo che utilizza la politica di autorizzazione di Amazon Bedrock.

Per creare il ruolo IAM

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel riquadro di navigazione scegliere Roles (Ruoli).
3. Scegliere Crea ruolo.
4. Nella pagina Seleziona un'entità affidabile, in Caso d'uso, scegli RDS.
5. Seleziona RDS- Aggiungi ruolo al database, quindi scegli Avanti.
6. Nella pagina Aggiungi autorizzazioni, per Politiche di autorizzazione, seleziona la IAM politica che hai creato, quindi scegli Avanti.
7. Nella pagina Nome, rivedi e crea, inserisci un nome per il tuo ruolo, ad esempio. **`ams-bedrock-invoke-model-role`**

   Il ruolo dovrebbe essere simile alla figura seguente.



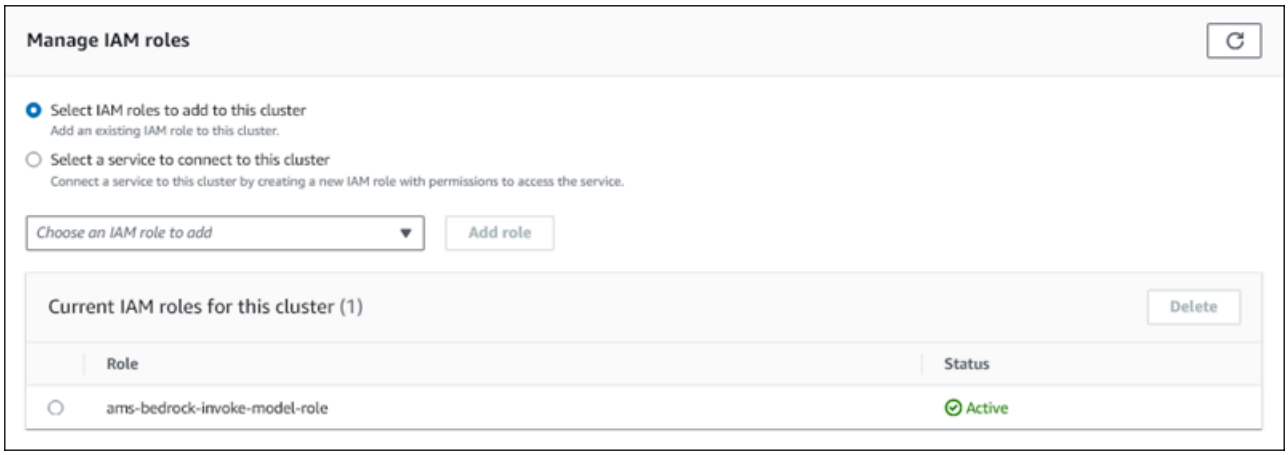
8. Rivedi il tuo ruolo, quindi scegli Crea ruolo.

Successivamente associ il IAM ruolo Amazon Bedrock al tuo cluster DB.

Per associare il IAM ruolo al tuo cluster DB

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione seleziona Database.
3. Scegli il cluster Aurora My SQL DB che desideri connettere ai servizi Amazon Bedrock.
4. Sceglere la scheda Connectivity & security (Connettività e sicurezza).
5. Nella sezione Gestisci IAM ruoli, scegli Seleziona IAM per aggiungere a questo cluster.
6. Scegli quello IAM che hai creato, quindi scegli Aggiungi ruolo.

   Il IAM ruolo è associato al cluster DB, prima con lo stato In sospeso, quindi Attivo. Una volta completato il processo, puoi trovare il ruolo nell'elenco IAMRuoli correnti per questo cluster.



È necessario aggiungere questo IAM ruolo al `aws_default_bedrock_role` parametro del gruppo di parametri del cluster DB personalizzato associato al cluster Aurora My SQL DB. ARN Se il cluster Aurora My SQL DB non utilizza un gruppo di parametri del cluster DB personalizzato, è necessario crearne uno da utilizzare con il cluster Aurora My SQL DB per completare l'integrazione. Per ulteriori informazioni, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

Per configurare il parametro del cluster DB

1. Nella RDS console Amazon, apri la scheda Configurazione del tuo cluster Aurora My SQL DB.
2. Individua il gruppo di parametri del cluster DB configurato per il tuo cluster. Scegli il link per aprire il gruppo di parametri del cluster DB personalizzato, quindi scegli Modifica.
3. Individua il parametro `aws_default_bedrock_role` nel gruppo di parametri cluster database personalizzato.

4. Nel campo Valore, inserisci ARN il IAM ruolo.
5. Scegli Salva modifiche per salvare l'impostazione.
6. Riavvia l'istanza principale del cluster Aurora SQL My DB in modo che questa impostazione dei parametri abbia effetto.

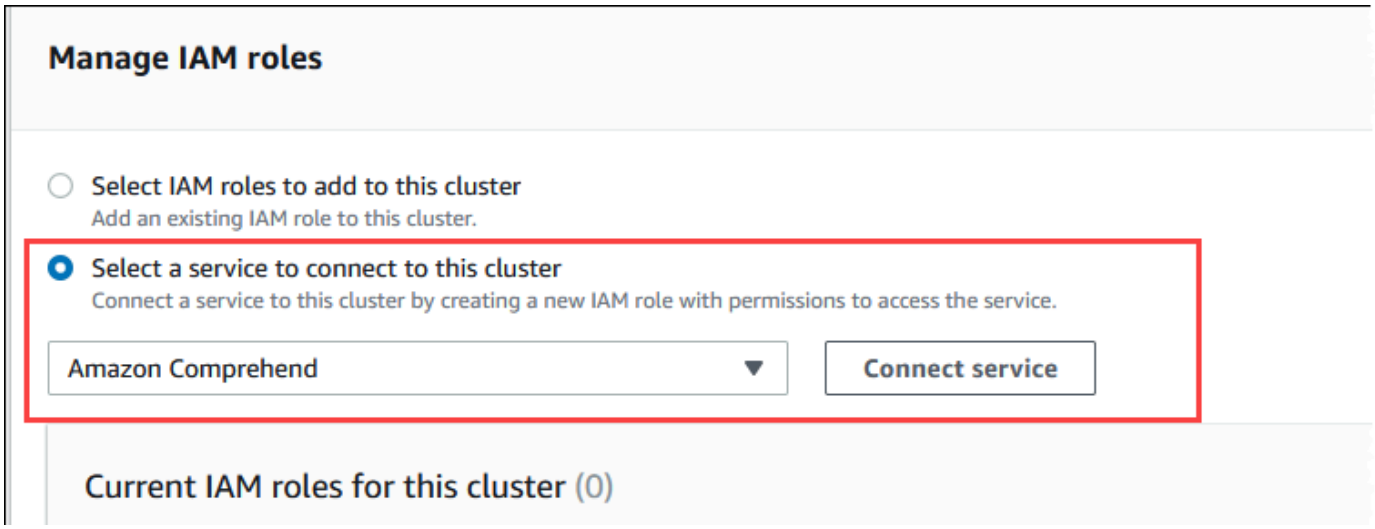
L'IAMintegrazione per Amazon Bedrock è completa. Continua a configurare il tuo cluster Aurora My SQL DB per l'utilizzo con Amazon Bedrock entro. Concessione agli utenti del database dell'accesso a machine learning di Aurora

Configurazione del cluster Aurora My SQL DB per utilizzare Amazon Comprehend

L'apprendimento automatico Aurora si basa su AWS Identity and Access Management ruoli e policy per consentire al cluster Aurora My SQL DB di accedere e utilizzare i servizi Amazon Comprehend. La procedura seguente crea automaticamente un IAM ruolo e una policy per il cluster in modo che possa utilizzare Amazon Comprehend.

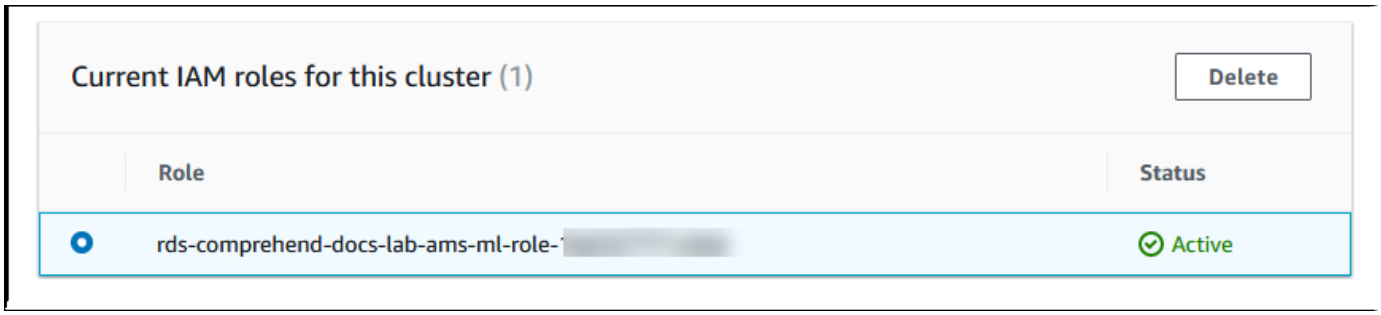
Per configurare il cluster Aurora My SQL DB per utilizzare Amazon Comprehend

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione seleziona Database.
3. Scegli il cluster Aurora My SQL DB che desideri connettere ai servizi Amazon Comprehend.
4. Sceglere la scheda Connectivity & security (Connettività e sicurezza).
5. Nella sezione Gestisci IAM ruoli, scegli Seleziona un servizio per connetterti a questo cluster.
6. Scegli Amazon Comprehend dal menu, quindi scegli il servizio Connect.



7. La finestra di dialogo Connetti il cluster ad Amazon Comprehend non richiede informazioni aggiuntive. Tuttavia, è possibile che venga visualizzato un messaggio che segnala che l'integrazione tra Aurora e Amazon Comprehend è al momento è in fase di anteprima. Assicurati di leggere il messaggio prima di continuare. Puoi scegliere Annulla se preferisci non procedere.
8. Scegli Connetti un servizio per completare il processo di integrazione.

   Aurora crea il ruolo. IAM Crea inoltre la policy che consente al cluster Aurora My SQL DB di utilizzare i servizi Amazon Comprehend e allega la policy al ruolo. Una volta completato il processo, puoi trovare il ruolo nell'elenco IAMRuoli correnti per questo cluster, come mostrato nell'immagine seguente.



   È necessario aggiungere questo IAM ruolo al `aws_default_comprehend_role` parametro del gruppo di parametri del cluster DB personalizzato associato al cluster Aurora My SQL DB. ARN Se il cluster Aurora My SQL DB non utilizza un gruppo di parametri del cluster DB personalizzato, è necessario crearne uno da utilizzare con il cluster Aurora My SQL DB per completare l'integrazione. Per ulteriori informazioni, consulta Gruppi di parametri del cluster DB per i cluster DB di Amazon Aurora.

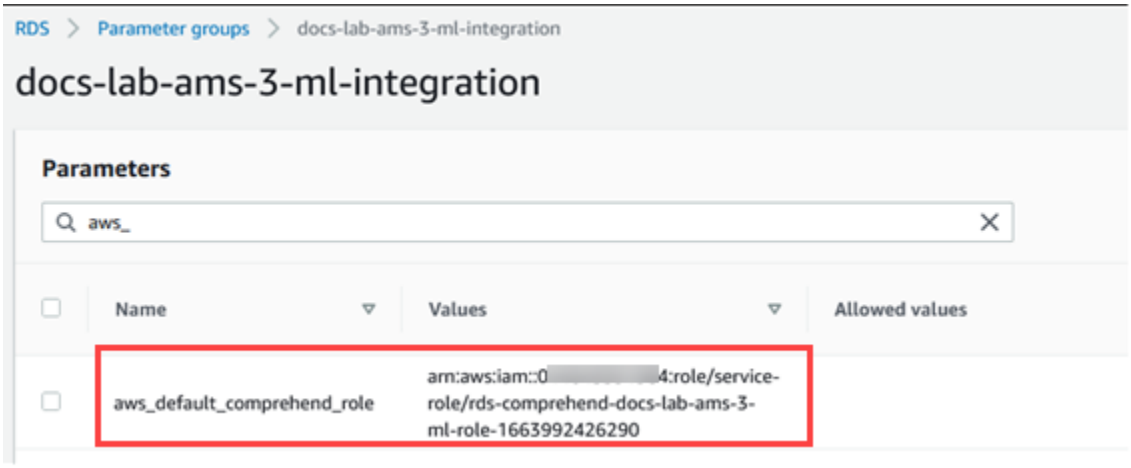
   Dopo aver creato il gruppo di parametri del cluster DB personalizzato e averlo associato al cluster Aurora SQL My DB, puoi continuare a seguire questi passaggi.

   Se il cluster utilizza un gruppo di parametri cluster database personalizzato, procedi come descritto di seguito.

   a. Nella RDS console Amazon, apri la scheda Configurazione del tuo cluster Aurora My SQL DB.
   b. Individua il gruppo di parametri del cluster DB configurato per il tuo cluster. Scegli il link per aprire il gruppo di parametri del cluster DB personalizzato, quindi scegli Modifica.
   c. Individua il parametro `aws_default_comprehend_role` nel gruppo di parametri cluster database personalizzato.

d. Nel campo Valore, inserisci ARN il IAM ruolo.

e. Scegli Salva modifiche per salvare l'impostazione. Nell'immagine seguente, è disponibile un esempio.

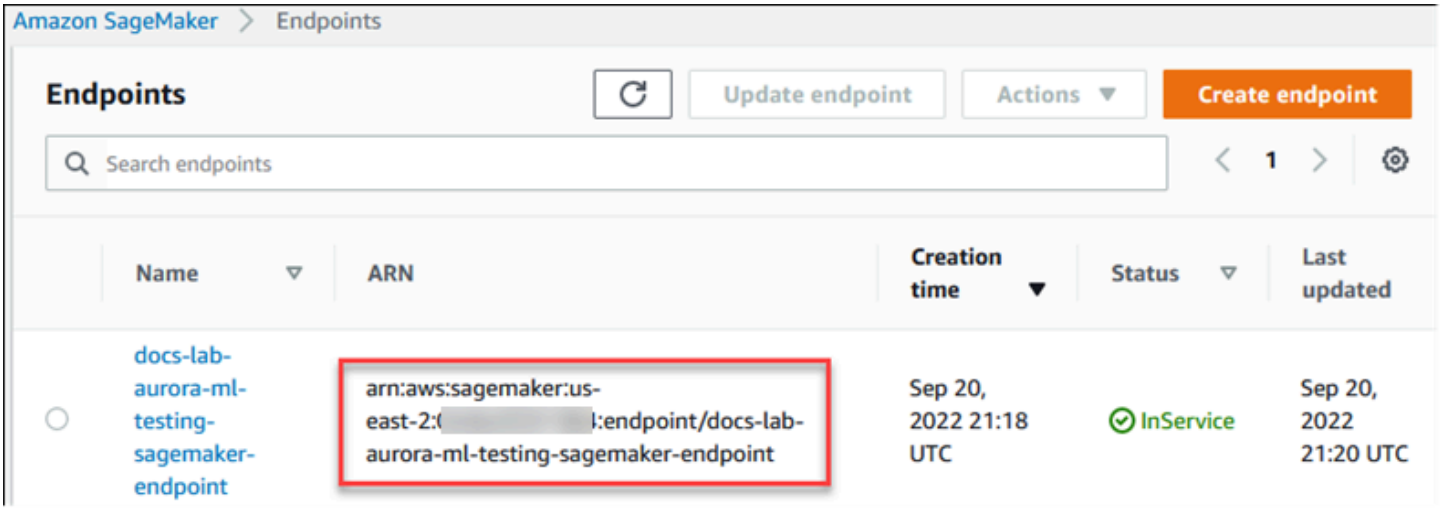


Riavvia l'istanza principale del cluster Aurora SQL My DB in modo che questa impostazione dei parametri abbia effetto.

L'IAMintegrazione per Amazon Comprehend è completa. Continua a configurare il tuo cluster Aurora My SQL DB per funzionare con Amazon Comprehend concedendo l'accesso agli utenti del database appropriati.

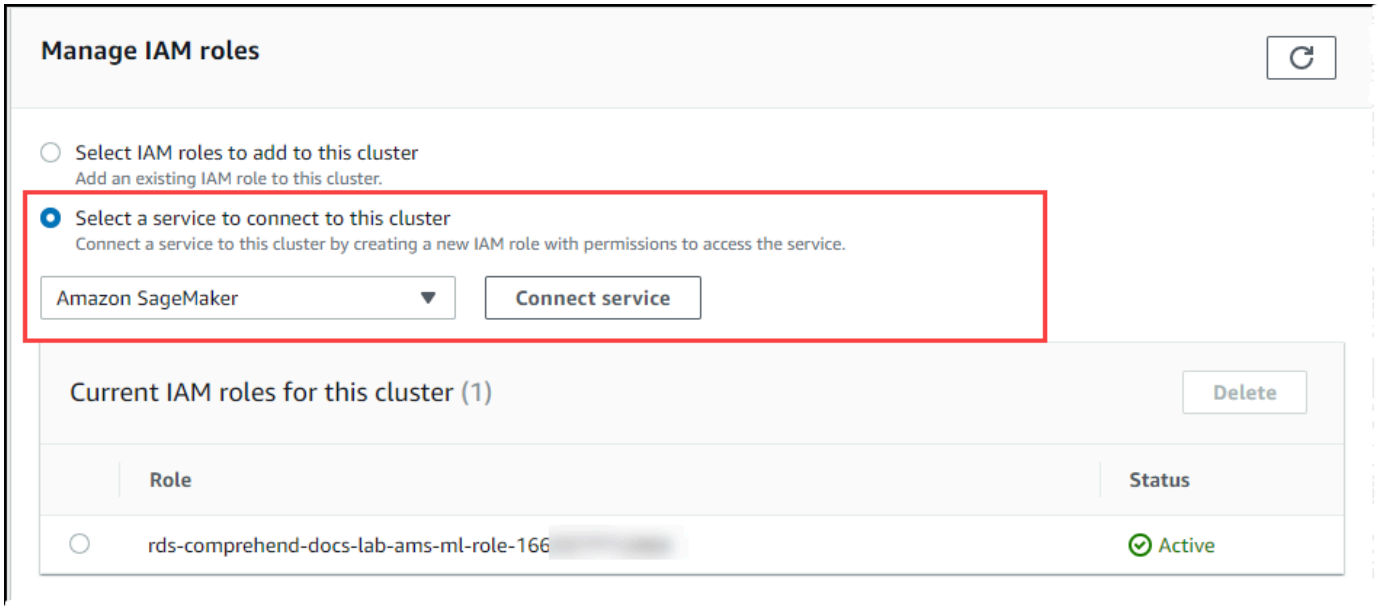
Configurazione del cluster Aurora My SQL DB per l'utilizzo dell'IA SageMaker

La procedura seguente crea automaticamente il IAM ruolo e la policy per il cluster Aurora My SQL DB in modo che possa utilizzare l' SageMaker IA. Prima di provare a seguire questa procedura, assicurati di avere a disposizione l'endpoint SageMaker AI in modo da poterlo inserire quando necessario. In genere, i data scientist del tuo team si occuperanno di produrre un endpoint che puoi usare dal tuo cluster Aurora SQL My DB. Puoi trovare tali endpoint nella console AISageMaker . Nel riquadro di navigazione, apri il menu Inferenza e scegli Endpoint. Nell'immagine seguente, è disponibile un esempio.



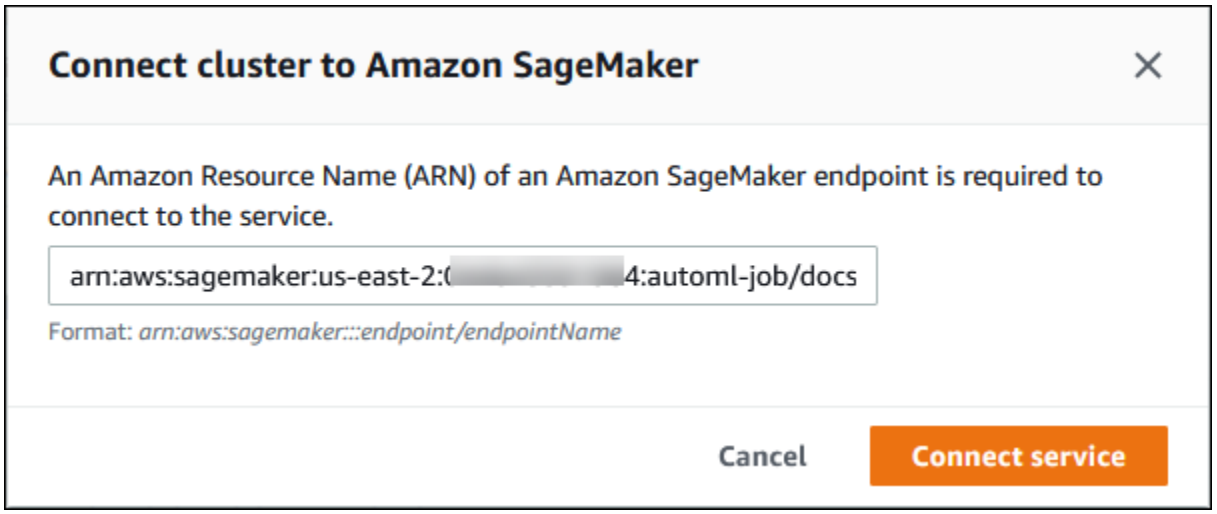
Per configurare il cluster Aurora My SQL DB per utilizzare l'IA SageMaker

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Databases dal menu di RDS navigazione Amazon, quindi scegli il cluster Aurora My SQL DB che desideri connettere ai servizi SageMaker AI.
3. Sceglere la scheda Connectivity & security (Connettività e sicurezza).
4. Scorri fino alla sezione Gestisci IAM ruoli, quindi scegli Seleziona un servizio per connetterti a questo cluster. Scegli SageMaker AI dal selettore.



5. Scegliere Connect service (Connetti servizio).

6. Nella finestra di dialogo Connect cluster to SageMaker AI, inserisci ARN l'endpoint SageMaker AI.



7. Aurora crea il ruolo. IAM Crea inoltre la policy che consente al cluster Aurora My SQL DB di utilizzare i servizi di SageMaker intelligenza artificiale e allega la policy al ruolo. Una volta completato il processo, puoi trovare il ruolo nell'elenco IAMRuoli correnti per questo cluster.
8. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
9. Scegli Ruoli dalla sezione Gestione degli accessi del menu di navigazione AWS Identity and Access Management .
10. Trova il ruolo tra quelli elencati. Il suo nome utilizza il seguente schema.

```
rds-sagemaker-your-cluster-name-role-auto-generated-digits
```

11. Apri la pagina di riepilogo del ruolo e individua ilARN. Prendi nota ARN o copialo utilizzando il widget di copia.
12. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
13. Scegli il tuo cluster Aurora My SQL DB, quindi scegli la scheda Configurazione.
14. Individua il gruppo di parametri cluster database e scegli il collegamento per aprire il gruppo di parametri cluster database personalizzato. Trova il `aws_default_sagemaker_role` parametro e inserisci il ARN IAM ruolo nel campo Valore e salva l'impostazione.
15. Riavvia l'istanza principale del cluster Aurora SQL My DB in modo che questa impostazione dei parametri abbia effetto.

La IAM configurazione è ora completa. Continua a configurare il tuo cluster Aurora My SQL DB per funzionare con l' SageMaker intelligenza artificiale concedendo l'accesso agli utenti del database appropriati.

Se desideri utilizzare i tuoi modelli di SageMaker intelligenza artificiale per la formazione anziché utilizzare componenti SageMaker AI predefiniti, devi anche aggiungere il bucket Amazon S3 al tuo cluster Aurora SQL My DB, come indicato di seguito. Configurazione del cluster Aurora My SQL DB per l'utilizzo di Amazon S3 SageMaker for AI (opzionale)

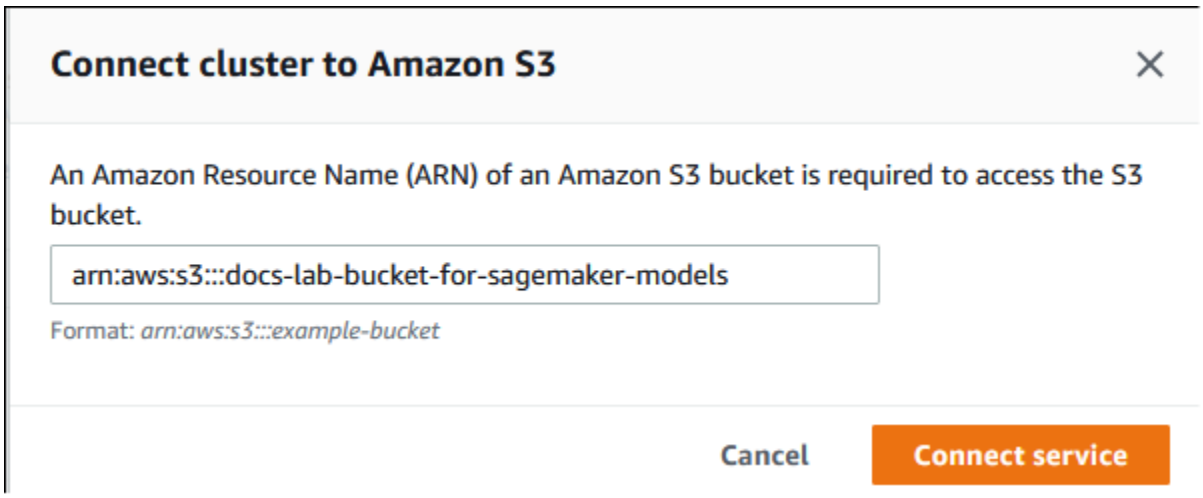
Configurazione del cluster Aurora My SQL DB per l'utilizzo di Amazon S3 SageMaker for AI (opzionale)

Per utilizzare l' SageMaker intelligenza artificiale con i tuoi modelli anziché utilizzare i componenti predefiniti forniti dall' SageMaker IA, devi configurare un bucket Amazon S3 per l'utilizzo del cluster Aurora My DBSQL. Per ulteriori informazioni sulla creazione di un bucket Amazon S3, consultaCreazione di un bucketnellaGuida all'utente di Amazon Simple Storage Service.

Per configurare il cluster Aurora My SQL DB per l'utilizzo di un bucket Amazon S3 per l'intelligenza artificiale SageMaker

1. Accedi a AWS Management Console e apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
2. Scegli Databases dal menu di RDS navigazione Amazon, quindi scegli il cluster Aurora My SQL DB che desideri connettere ai servizi SageMaker AI.
3. Sceglere la scheda Connectivity & security (Connettività e sicurezza).
4. Scorri fino alla sezione Gestisci IAM ruoli, quindi scegli Seleziona un servizio per connetterti a questo cluster. Scegli Amazon S3 dal selettore.

5. Scegliere Connect service (Connetti servizio).
6. Nella finestra di dialogo Connect cluster to Amazon S3, inserisci il ARN bucket Amazon S3, come mostrato nell'immagine seguente.



7. Scegli Connetti un servizio per completare questo processo.

Per ulteriori informazioni sull'uso dei bucket Amazon S3 con l' SageMaker intelligenza artificiale, consulta Specificare un bucket Amazon S3 per caricare set di dati di addestramento e archiviare i dati di output nella Amazon AI Developer Guide. SageMaker Per ulteriori informazioni su come lavorare con l' SageMaker intelligenza artificiale, consulta la sezione Get Started with Amazon SageMaker AI Notebook Instances nella Amazon SageMaker AI Developer Guide.

Concessione agli utenti del database dell'accesso a machine learning di Aurora

Agli utenti del database deve essere concessa l'autorizzazione per richiamare le funzioni di machine learning di Aurora. La modalità di concessione dell'autorizzazione dipende dalla versione SQL di My utilizzata per il cluster Aurora My SQL DB, come descritto di seguito. Il modo in cui eseguire SQL questa operazione dipende dalla versione di My utilizzata dal cluster Aurora My SQL DB.

- Per Aurora My SQL versione 3 (compatibile con My SQL 8.0), agli utenti del database deve essere assegnato il ruolo di database appropriato. Per ulteriori informazioni, consulta Using Roles in the My SQL 8.0 Reference Manual.
- Per Aurora My SQL versione 2 (compatibile con My SQL 5.7), agli utenti del database vengono concessi privilegi. Per ulteriori informazioni, consulta Controllo degli accessi e gestione degli account nel manuale di riferimento My SQL 5.7.

La tabella seguente mostra i ruoli e i privilegi necessari agli utenti del database per utilizzare le funzioni di machine learning.

Aurora La mia SQL versione 3 (ruolo)	Aurora La mia SQL versione 2 (privilegio)
AWS_BEDROCK_ACCESS	–
AWS_COMPREHEND_ACCESS	INVOKE COMPREHEND
AWS_SAGEMAKER_ACCESS	INVOKE SAGEMAKER

Concessione dell'accesso alle funzioni di Amazon Bedrock

Per consentire agli utenti del database di accedere alle funzioni di Amazon Bedrock, utilizza la seguente SQL dichiarazione:

```
GRANT AWS_BEDROCK_ACCESS TO user@domain-or-ip-address;
```

Agli utenti del database devono inoltre essere concesse `EXECUTE` le autorizzazioni per le funzioni che crei per lavorare con Amazon Bedrock:

```
GRANT EXECUTE ON FUNCTION database_name.function_name TO user@domain-or-ip-address;
```

Infine, gli utenti del database devono avere i ruoli impostati su: `AWS_BEDROCK_ACCESS`

```
SET ROLE AWS_BEDROCK_ACCESS;
```

Le funzioni di Amazon Bedrock sono ora disponibili per l'uso.

Concessione dell'accesso alle funzioni Amazon Comprehend

Per consentire agli utenti del database di accedere alle funzioni di Amazon Comprehend, usa l'istruzione appropriata per la tua versione di Aurora My. SQL

- Aurora My SQL versione 3 (compatibile con My SQL 8.0)

  ```
  GRANT AWS_COMPREHEND_ACCESS TO user@domain-or-ip-address;
  ```

- Aurora My SQL versione 2 (compatibile con My SQL 5.7)

  ```
  GRANT INVOKE COMPREHEND ON *.* TO user@domain-or-ip-address;
  ```

Le funzioni di Amazon Comprehend sono ora disponibili per l'uso. Per esempi di utilizzo, consultare Utilizzo di Amazon Comprehend con il cluster Aurora My DB SQL.

Concessione dell'accesso alle funzioni AI SageMaker

Per consentire agli utenti del database di accedere alle funzioni di SageMaker intelligenza artificiale, usa l'istruzione appropriata per la tua versione di Aurora MySQL.

- Aurora My SQL versione 3 (compatibile con My SQL 8.0)

```
GRANT AWS_SAGEMAKER_ACCESS TO user@domain-or-ip-address;
```

- Aurora My SQL versione 2 (compatibile con My SQL 5.7)

```
GRANT INVOKE SAGEMAKER ON *.* TO user@domain-or-ip-address;
```

Agli utenti del database devono inoltre essere concesse `EXECUTE` le autorizzazioni per le funzioni create per lavorare con l'intelligenza artificiale. SageMaker Supponiamo di aver creato due funzioni `db1.anomoly_score` e di richiamare `db2.company_forecasts` i servizi del tuo SageMaker endpoint di intelligenza artificiale. Concedete i privilegi di esecuzione come illustrato nell'esempio seguente.

```
GRANT EXECUTE ON FUNCTION db1.anomaly_score TO user1@domain-or-ip-address1;
GRANT EXECUTE ON FUNCTION db2.company_forecasts TO user2@domain-or-ip-address2;
```

Le funzioni di SageMaker intelligenza artificiale sono ora disponibili per l'uso. Per esempi di utilizzo, consultare Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora My DB SQL.

Utilizzo di Amazon Bedrock con il cluster Aurora My DB SQL

Per usare Amazon Bedrock, crei una funzione definita dall'utente (UDF) nel tuo SQL database Aurora My che richiama un modello. Per ulteriori informazioni, consulta la sezione Modelli supportati in Amazon Bedrock nella Amazon Bedrock User Guide.

A UDF utilizza la seguente sintassi:

```
CREATE FUNCTION function_name (argument type)
        [DEFINER = user]
    RETURNS mysql_data_type
    [SQL SECURITY {DEFINER | INVOKER}]
```

```
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'model_id'
    [CONTENT_TYPE 'content_type']
    [ACCEPT 'content_type']
    [TIMEOUT_MS timeout_in_milliseconds];
```

- Le funzioni di Amazon Bedrock non sono `RETURNS JSON` supportate. Puoi utilizzare `CONVERT` o convertire da `CAST TEXT` a, `JSON` se necessario.
- Se non specifichi `CONTENT_TYPE` o`ACCEPT`, l'impostazione predefinita è`application/json`.
- Se non si specifica`TIMEOUT_MS`, `aurora_ml_inference_timeout` viene utilizzato il valore per.

Ad esempio, quanto segue UDF richiama il modello Amazon Titan Text Express:

```
CREATE FUNCTION invoke_titan (request_body TEXT)
    RETURNS TEXT
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'amazon.titan-text-express-v1'
    CONTENT_TYPE 'application/json'
    ACCEPT 'application/json';
```

Per consentire a un utente DB di utilizzare questa funzione, usa il seguente comando: SQL

```
GRANT EXECUTE ON FUNCTION database_name.invoke_titan TO user@domain-or-ip-address;
```

Quindi l'utente può chiamare `invoke_titan` come qualsiasi altra funzione, come mostrato nell'esempio seguente. Assicurati di formattare il corpo della richiesta in base ai modelli di testo di Amazon Titan.

```
CREATE TABLE prompts (request varchar(1024));
INSERT INTO prompts VALUES (
'{
    "inputText": "Generate synthetic data for daily product sales in various categories
 - include row number, product name, category, date of sale and price. Produce output
 in JSON format. Count records and ensure there are no more than 5.",
    "textGenerationConfig": {
        "maxTokenCount": 1024,
        "stopSequences": [],
        "temperature":0,
        "topP":1
    }
```

```
}');

SELECT invoke_titan(request) FROM prompts;

{"inputTextTokenCount":44,"results":[{"tokenCount":296,"outputText":"
```tabular-data-json
{
    "rows": [
        {
            "Row Number": "1",
            "Product Name": "T-Shirt",
            "Category": "Clothing",
            "Date of Sale": "2024-01-01",
            "Price": "$20"
        },
        {
            "Row Number": "2",
            "Product Name": "Jeans",
            "Category": "Clothing",
            "Date of Sale": "2024-01-02",
            "Price": "$30"
        },
        {
            "Row Number": "3",
            "Product Name": "Hat",
            "Category": "Accessories",
            "Date of Sale": "2024-01-03",
            "Price": "$15"
        },
        {
            "Row Number": "4",
            "Product Name": "Watch",
            "Category": "Accessories",
            "Date of Sale": "2024-01-04",
            "Price": "$40"
        },
        {
            "Row Number": "5",
            "Product Name": "Phone Case",
            "Category": "Accessories",
            "Date of Sale": "2024-01-05",
            "Price": "$25"
        }
    ]
```
```

```
}
```","completionReason":"FINISH"}]}
```

Per gli altri modelli che utilizzi, assicurati di formattare il corpo della richiesta in modo appropriato per loro. Per ulteriori informazioni, consulta i parametri di inferenza per i modelli di base nella Amazon Bedrock User Guide.

## Utilizzo di Amazon Comprehend con il cluster Aurora My DB SQL

Per Aurora My, l'apprendimento automatico di SQL Aurora offre le seguenti due funzioni integrate per lavorare con Amazon Comprehend e i tuoi dati di testo. Viene fornito il testo da analizzare (`input_data`) e specificata la lingua (`language_code`).

aws_comprehend_detect_sentiment

Questa funzione identifica il testo come avente un assetto emotivo positivo, negativo, neutro o misto. La documentazione di riferimento di questa funzione è la seguente.

```
aws_comprehend_detect_sentiment(
  input_text,
  language_code
  [,max_batch_size]
)
```

Per ulteriori informazioni, consultare Sentiment nella Guida per gli sviluppatori di Amazon Comprehend.

aws_comprehend_detect_sentiment_confidence

Questa funzione misura il livello di affidabilità del sentiment rilevato per un determinato testo. Restituisce un valore (tipo, `double`) che indica l'affidabilità del sentiment assegnato dalla funzione aws_comprehend_detect_sentiment al testo. L'affidabilità è un parametro statistico compreso tra 0 e 1. Più alto è il livello di affidabilità, maggiore è il peso che è possibile assegnare al risultato. Di seguito è riportato un riepilogo della documentazione della funzione.

```
aws_comprehend_detect_sentiment_confidence(
  input_text,
  language_code
  [,max_batch_size]
)
```
```

In entrambe le funzioni (aws_comprehend_detect_sentiment_confidence, aws_comprehend_detect_sentiment) il parametro `max_batch_size` utilizza un valore predefinito pari a 25, se uno non è stato specificato. La dimensione del batch deve essere sempre maggiore di zero. `max_batch_size` può essere utilizzato per ottimizzare le prestazioni delle chiamate di funzione Amazon Comprehend. Un batch di grandi dimensioni compromette prestazioni più veloci con un maggiore utilizzo della memoria sul cluster Aurora SQL My DB. Per ulteriori informazioni, consulta Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora My SQL.

Per ulteriori informazioni sui parametri e sui tipi di rendimento per le funzioni di rilevamento del sentiment in Amazon Comprehend, consulta DetectSentiment

Example Esempio: una semplice query che utilizza funzioni Amazon Comprehend

Di seguito è riportato un esempio di una semplice query che richiama queste due funzioni per determinare il livello di soddisfazione dei clienti per il team di supporto. Si supponga di avere una tabella di database (`support`) che archivia il feedback dei clienti dopo ogni richiesta di assistenza. Questa query di esempio applica entrambe le funzioni integrate al testo nella colonna `feedback` della tabella e restituisce i risultati. I valori di affidabilità restituiti dalla funzione sono double compresi tra 0,0 e 1,0. Per un output più leggibile, questa query arrotonda i risultati a 6 punti decimali. Per semplificare i confronti, questa query ordina inoltre i risultati in senso decrescente, a partire dal risultato con il massimo grado di affidabilità.

```
SELECT feedback AS 'Customer feedback',
    aws_comprehend_detect_sentiment(feedback, 'en') AS Sentiment,
    ROUND(aws_comprehend_detect_sentiment_confidence(feedback, 'en'), 6)
    AS Confidence FROM support
    ORDER BY Confidence DESC;
+-----------------------------------------------------------+-----------+------------+
| Customer feedback                                         | Sentiment | Confidence |
+-----------------------------------------------------------+-----------+------------+
| Thank you for the excellent customer support!             | POSITIVE  |   0.999771 |
| The latest version of this product stinks!                | NEGATIVE  |   0.999184 |
| Your support team is just awesome! I am blown away.       | POSITIVE  |   0.997774 |
| Your product is too complex, but your support is great.   | MIXED     |   0.957958 |
| Your support tech helped me in fifteen minutes.           | POSITIVE  |   0.949491 |
| My problem was never resolved!                            | NEGATIVE  |   0.920644 |
| When will the new version of this product be released?    | NEUTRAL   |   0.902706 |
| I cannot stand that chatbot.                              | NEGATIVE  |   0.895219 |
| Your support tech talked down to me.                      | NEGATIVE  |   0.868598 |
| It took me way too long to get a real person.             | NEGATIVE  |   0.481805 |
```

```
+-----------------------------------------------------------+-----------+------------+
 10 rows in set (0.1898 sec)
```

Example Esempio: determinazione del sentiment medio per testo al di sopra di un livello di affidabilità specifico

Una query Amazon Comprehend tipica cerca le righe in cui il sentiment ha un determinato valore, con un livello di confidenza maggiore di un certo numero. Ad esempio, la query seguente mostra come è possibile determinare il sentiment medio dei documenti nel database. La query considera solo i documenti in cui la confidenza della valutazione è almeno dell'80%.

```
SELECT AVG(CASE aws_comprehend_detect_sentiment(productTable.document, 'en')
  WHEN 'POSITIVE' THEN 1.0
  WHEN 'NEGATIVE' THEN -1.0
  ELSE 0.0 END) AS avg_sentiment, COUNT(*) AS total
FROM productTable
WHERE productTable.productCode = 1302 AND
    aws_comprehend_detect_sentiment_confidence(productTable.document, 'en') >= 0.80;
```

Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora My DB SQL

Per utilizzare la funzionalità di SageMaker intelligenza artificiale del cluster Aurora My SQL DB, è necessario creare funzioni archiviate che incorporino le chiamate all'endpoint SageMaker AI e alle sue funzionalità di inferenza. Puoi farlo utilizzando SQL My's `CREATE FUNCTION` in genere nello stesso modo in cui lo fai per altre attività di elaborazione sul tuo cluster Aurora My SQL DB.

Per utilizzare i modelli implementati nell' SageMaker intelligenza artificiale per l'inferenza, crei funzioni definite dall'utente utilizzando le istruzioni My SQL data definition language (DDL) per le funzioni archiviate. Ogni funzione memorizzata rappresenta l'endpoint SageMaker AI che ospita il modello. Quando si definisce una funzione di questo tipo, si specificano i parametri di input per il modello, l'endpoint SageMaker AI specifico da richiamare e il tipo restituito. La funzione restituisce l'inferenza calcolata dall'endpoint SageMaker AI dopo aver applicato il modello ai parametri di input.

Tutte le funzioni archiviate Aurora Machine Learning restituiscono tipi numerici o `VARCHAR`. È possibile utilizzare qualsiasi tipo numerico tranne `BIT`. Altri tipi, ad esempio `JSON`, `BLOB`, `TEXT` e `DATE` non sono consentiti.

L'esempio seguente mostra la `CREATE FUNCTION` sintassi per lavorare con l'IA. SageMaker

```
CREATE FUNCTION function_name (
     arg1 type1,
     arg2 type2, ...)
     [DEFINER = user]
  RETURNS mysql_type
  [SQL SECURITY { DEFINER | INVOKER } ]
  ALIAS AWS_SAGEMAKER_INVOKE_ENDPOINT
  ENDPOINT NAME 'endpoint_name'
  [MAX_BATCH_SIZE max_batch_size];
```

Questa è un'estensione dell'`CREATE FUNCTION` DDListruzione regolare. Nell'`CREATE FUNCTION`istruzione che definisce la funzione SageMaker AI, non si specifica il corpo della funzione. Specifica invece la parola chiave `ALIAS` al posto del corpo della funzione. Attualmente, Aurora Machine Learning supporta solo `aws_sagemaker_invoke_endpoint` per questa sintassi estesa. È necessario specificare il parametro `endpoint_name`. Un endpoint SageMaker AI può avere caratteristiche diverse per ogni modello.

> **Note**
>
> Per ulteriori informazioni su`CREATE FUNCTION`, consulta CREATEPROCEDUREe CREATE FUNCTION Dichiarazioni nel manuale di riferimento My SQL 8.0.

Il parametro `max_batch_size` è facoltativo. Per impostazione predefinita, la dimensione batch massima è 10.000. È possibile utilizzare questo parametro nella funzione per limitare all'IA il numero massimo di input elaborati in una richiesta in batch. SageMaker Il `max_batch_size` parametro può aiutare a evitare un errore causato da input troppo grandi o a far sì che l' SageMaker IA restituisca una risposta più rapidamente. Questo parametro influisce sulla dimensione di un buffer interno utilizzato per l'elaborazione delle richieste SageMaker AI. Se specifichi un valore troppo grande per `max_batch_size`, è possibile che si verifichi un notevole sovraccarico di memoria nell'istanza database.

Ti consigliamo di lasciare l'impostazione `MANIFEST` sul valore predefinito `OFF`. Sebbene sia possibile utilizzare l'`MANIFEST ON`opzione, alcune funzionalità di SageMaker intelligenza artificiale non possono utilizzare direttamente ciò che è CSV stato esportato con questa opzione. Il formato manifesto non è compatibile con il formato manifesto previsto da SageMaker AI.

Crei una funzione memorizzata separata per ciascuno dei tuoi modelli di SageMaker intelligenza artificiale. Questa mappatura delle funzioni sui modelli è necessaria perché un endpoint è associato

a un modello specifico e ciascun modello accetta parametri diversi. L'utilizzo dei SQL tipi per gli input del modello e il tipo di output del modello aiuta a evitare errori di conversione dei tipi nel trasferimento dei dati avanti e indietro tra i AWS servizi. È possibile controllare chi può applicare il modello. È inoltre possibile controllare le caratteristiche di runtime specificando un parametro che rappresenta la dimensione massima del batch.

Attualmente tutte le funzioni Aurora Machine Learning hanno la proprietà `NOT DETERMINISTIC`. Se non specifichi la proprietà esplicitamente, Aurora imposta `NOT DETERMINISTIC` automaticamente. Questo requisito è dovuto al fatto che il modello di SageMaker intelligenza artificiale può essere modificato senza alcuna notifica al database. In tal caso, le chiamate a una funzione Aurora Machine Learning potrebbero restituire risultati diversi per lo stesso input all'interno di una singola transazione.

Non puoi usare le caratteristiche `CONTAINS SQL`, `NO SQL`, `READS SQL DATA` o `MODIFIES SQL DATA` nell'istruzione `CREATE FUNCTION`.

Di seguito è riportato un esempio di utilizzo dell'invocazione di un endpoint SageMaker AI per rilevare anomalie. Esiste un endpoint AI. SageMaker `random-cut-forest-model` Il modello corrispondente è già stato addestrato dall'algoritmo `random-cut-forest`. Per ogni input, il modello restituisce un punteggio di anomalia. Questo esempio mostra i punti dati il cui punteggio è maggiore di 3 deviazioni standard (circa il 99,9 percentile) dal punteggio medio.

```
CREATE FUNCTION anomaly_score(value real) returns real
  alias aws_sagemaker_invoke_endpoint endpoint name 'random-cut-forest-model-demo';

set @score_cutoff = (select avg(anomaly_score(value)) + 3 * std(anomaly_score(value))
 from nyc_taxi);

select *, anomaly_detection(value) score from nyc_taxi
  where anomaly_detection(value) > @score_cutoff;
```

Requisito del set di caratteri per le funzioni SageMaker AI che restituiscono stringhe

Ti consigliamo di specificare un set di caratteri `utf8mb4` come tipo restituito per le funzioni SageMaker AI che restituiscono valori di stringa. Se non è pratico, utilizza una lunghezza della stringa sufficiente per il tipo restituito per contenere un valore rappresentato nel set di caratteri `utf8mb4`. L'esempio seguente mostra come dichiarare il set di caratteri `utf8mb4` per la tua funzione.

```
CREATE FUNCTION my_ml_func(...) RETURNS VARCHAR(5) CHARSET utf8mb4 ALIAS ...
```

Attualmente, ogni funzione SageMaker AI che restituisce una stringa utilizza il set di caratteri `utf8mb4` per il valore restituito. Il valore restituito utilizza questo set di caratteri anche se la funzione SageMaker AI dichiara un set di caratteri diverso per il tipo restituito in modo implicito o esplicito. Se la funzione SageMaker AI dichiara un set di caratteri diverso per il valore restituito, i dati restituiti potrebbero essere troncati automaticamente se li memorizzi in una colonna della tabella che non è sufficientemente lunga. Ad esempio, una query con una clausola `DISTINCT` crea una tabella temporanea. Pertanto, il risultato della funzione SageMaker AI potrebbe essere troncato a causa del modo in cui le stringhe vengono gestite internamente durante una query.

Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)

Ti consigliamo di iniziare a utilizzare l'apprendimento automatico e l' SageMaker intelligenza artificiale di Aurora utilizzando alcuni degli algoritmi forniti e che i data scientist del tuo team ti forniscano gli endpoint di SageMaker intelligenza artificiale da utilizzare con il tuo codice. SQL Di seguito, puoi trovare informazioni minime sull'utilizzo del tuo bucket Amazon S3 con i tuoi modelli di SageMaker intelligenza artificiale e il tuo cluster Aurora My DB. SQL

La funzione machine learning comprende due fasi principali: training e inferenza. Per addestrare i modelli di SageMaker intelligenza artificiale, esporti i dati in un bucket Amazon S3. Il bucket Amazon S3 viene utilizzato da un'istanza di notebook Jupyter SageMaker AI per addestrare il modello prima della distribuzione. È possibile utilizzare l'`SELECT INTO OUTFILE S3`istruzione per interrogare i dati da un cluster Aurora My SQL DB e salvarli direttamente in file di testo archiviati in un bucket Amazon S3. Quindi, l'istanza notebook utilizza i dati dal bucket Simple Storage Service (Amazon S3) per il training.

L'apprendimento automatico Aurora estende la `SELECT INTO OUTFILE` sintassi esistente in Aurora My SQL per esportare i dati in formato. CSV Il CSV file generato può essere utilizzato direttamente dai modelli che richiedono questo formato per scopi di formazione.

```
SELECT * INTO OUTFILE S3 's3_uri' [FORMAT {CSV|TEXT} [HEADER]] FROM table_name;
```

L'estensione supporta il CSV formato standard.

- Il formato `TEXT` è lo stesso del formato My SQL export esistente. Questo è il formato predefinito.
- `CSV`Il formato è un formato di nuova introduzione che segue le specifiche in RFC-4180.
- Se si specifica la parola chiave facoltativa `HEADER`, il file di output contiene una riga di intestazione. Le etichette nella riga di intestazione corrispondono ai nomi di colonna dell'istruzione `SELECT`.

- Puoi sempre usare le parole chiave `CSV` e `HEADER` come identificatori.

La sintassi estesa e la grammatica di `SELECT INTO` ora sono le seguenti:

```
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
[FORMAT {CSV|TEXT} [HEADER]]
[{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']
]
```

Considerazioni sulle prestazioni per l'utilizzo del machine learning di Aurora con Aurora My SQL

I servizi Amazon Bedrock, Amazon Comprehend SageMaker e AI svolgono la maggior parte del lavoro quando vengono richiamati da una funzione di machine learning di Aurora. Ciò significa che è possibile dimensionare tali risorse in base alle esigenze, in modo indipendente. Per il tuo cluster Aurora My SQL DB, puoi rendere le chiamate di funzione il più efficienti possibile. Di seguito sono riportate alcune considerazioni sulle prestazioni da tenere presenti quando si utilizza machine learning di Aurora.

Modello e prompt

Le prestazioni quando si utilizza Amazon Bedrock dipendono in larga misura dal modello e dal prompt utilizzati. Scegli un modello e un prompt ottimali per il tuo caso d'uso.

Cache delle query

La cache Aurora My SQL query non funziona per le funzioni di machine learning di Aurora. Aurora My SQL non archivia i risultati delle query nella cache delle query per le SQL istruzioni che richiamano le funzioni di machine learning di Aurora.

Ottimizzazione batch per le chiamate di funzione Aurora Machine Learning

L'aspetto principale delle prestazioni Aurora Machine Learning che è possibile applicare dal cluster Aurora è l'impostazione della modalità batch per le chiamate alle funzioni archiviate Aurora Machine Learning. Le funzioni di machine learning in genere richiedono un sovraccarico considerevole, rendendo impossibile chiamare separatamente un servizio esterno per ogni riga. Il machine learning di Aurora può ridurre al minimo questo sovraccarico combinando le chiamate al servizio di machine learning Aurora esterno per molte righe in un singolo batch. Il machine learning di Aurora riceve le risposte per un batch di righe di input e quindi restituisce le risposte alla query in esecuzione una riga alla volta. Questa ottimizzazione migliora la velocità effettiva e la latenza delle query Aurora senza modificare i risultati.

Quando crei una funzione memorizzata Aurora connessa a un endpoint SageMaker AI, definisci il parametro della dimensione del batch. Questo parametro influenza il numero di righe trasferite per ogni chiamata sottostante all' SageMaker IA. Per le query che elaborano un numero elevato di righe, il sovraccarico necessario per effettuare una chiamata SageMaker AI separata per ogni riga può essere notevole. Maggiore è il set di dati elaborato dalla stored procedure, maggiore è la dimensione del batch.

Se l'ottimizzazione in modalità batch può essere applicata a una funzione di SageMaker intelligenza artificiale, puoi capirlo controllando il piano di interrogazione prodotto dall'`EXPLAIN PLAN`istruzione. In questo caso, la colonna `extra` nel piano di esecuzione include `Batched machine learning`. L'esempio seguente mostra una chiamata a una funzione SageMaker AI che utilizza la modalità batch.

```
mysql> CREATE FUNCTION anomaly_score(val real) returns real alias
 aws_sagemaker_invoke_endpoint endpoint name 'my-rcf-model-20191126';
Query OK, 0 rows affected (0.01 sec)

mysql> explain select timestamp, value, anomaly_score(value) from nyc_taxi;
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows | filtered | Extra                   |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
|  1 | SIMPLE      | nyc_taxi | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |   48 |   100.00 | Batched machine learning |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
```

```
1 row in set, 1 warning (0.01 sec)
```

Quando si chiama una delle funzioni Amazon Comprehend integrate, è possibile controllare la dimensione del batch specificando il parametro facoltativo `max_batch_size` che limita il numero massimo di valori `input_text` elaborati in ogni batch. L'invio di più elementi contemporaneamente riduce il numero di passaggi tra Aurora e Amazon Comprehend. Limitare la dimensione del batch è utile in situazioni come le query con una clausola `LIMIT`. Utilizzando un valore basso per `max_batch_size`, puoi evitare di invocare Amazon Comprehend più volte di quanti sono i testi di input.

L'ottimizzazione batch per la valutazione delle funzioni Aurora Machine Learning si applica nei seguenti casi:

- Chiamate di funzioni all'interno dell'elenco di selezione o della `WHERE` clausola di istruzioni `SELECT`
- Chiamate di funzione nell'`VALUES`Selenco delle istruzioni `INSERT` and `REPLACE`
- SageMaker L'intelligenza artificiale funziona nei `SET` valori nelle `UPDATE` dichiarazioni:

```
INSERT INTO MY_TABLE (col1, col2, col3) VALUES
  (ML_FUNC(1), ML_FUNC(2), ML_FUNC(3)),
  (ML_FUNC(4), ML_FUNC(5), ML_FUNC(6));
UPDATE MY_TABLE SET col1 = ML_FUNC(col2), SET col3 = ML_FUNC(col4) WHERE ...;
```

Monitoraggio del machine learning di Aurora

È possibile monitorare le operazioni batch di machine learning di Aurora interrogando diverse variabili globali, come illustrato nell'esempio seguente.

```
show status like 'Aurora_ml%';
```

Lo stato di queste variabili può essere ripristinato utilizzando un'istruzione `FLUSH STATUS`. Pertanto, tutte le cifre rappresentano i totali, le medie e così via, dall'ultima volta che la variabile è stata ripristinata.

`Aurora_ml_logical_request_cnt`

Il numero di richieste logiche valutate dall'istanza database per essere inviate ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato. A seconda del fatto che sia stato utilizzato il batching, questo valore può essere superiore a `Aurora_ml_actual_request_cnt`.

`Aurora_ml_logical_response_cnt`

Il numero di risposte aggregate che Aurora SQL My riceve dai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB.

`Aurora_ml_actual_request_cnt`

Il conteggio aggregato delle richieste che Aurora SQL My invia ai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB.

`Aurora_ml_actual_response_cnt`

Il numero di risposte aggregate che Aurora SQL My riceve dai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB.

`Aurora_ml_cache_hit_cnt`

Il numero aggregato di accessi alla cache interna che Aurora SQL My riceve dai servizi di machine learning Aurora su tutte le query eseguite dagli utenti dell'istanza DB.

`Aurora_ml_retry_request_cnt`

Il numero di richieste ripetute inviate dall'istanza database ai servizi di machine learning di Aurora dall'ultimo ripristino dello stato.

`Aurora_ml_single_request_cnt`

Il conteggio aggregato delle funzioni di machine learning di Aurora valutate in modalità non batch per tutte le query eseguite dagli utenti dell'istanza database.

Per informazioni sul monitoraggio delle prestazioni delle operazioni di SageMaker intelligenza artificiale richiamate dalle funzioni di machine learning di Aurora, consulta Monitor Amazon SageMaker AI.

Utilizzo dell'apprendimento automatico di Amazon Aurora con Aurora Postgre SQL

Utilizzando l'apprendimento automatico di Amazon Aurora con il tuo cluster Aurora Postgre SQL DB, puoi usare Amazon Comprehend o Amazon SageMaker AI o Amazon Bedrock, a seconda delle tue esigenze. Ciascuno di questi servizi supporta casi d'uso specifici di machine learning.

L'apprendimento automatico di Aurora è supportato solo in alcune versioni Regioni AWS e per specifiche di Aurora Postgre. SQL Prima di provare a configurare l'apprendimento automatico di

Aurora, verifica la disponibilità della tua versione di Aurora SQL Postgre e della tua regione. Per informazioni dettagliate, consultare Apprendimento automatico Aurora con Aurora Postgre SQL.

Argomenti

- Requisiti per l'utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL
- Funzionalità e limitazioni supportate dell'apprendimento automatico di Aurora con Aurora Postgre SQL
- Configurazione del cluster Aurora Postgre SQL DB per utilizzare l'apprendimento automatico Aurora
- Utilizzo di Amazon Bedrock con il cluster Aurora SQL Postgre DB
- Utilizzo di Amazon Comprehend con il cluster Aurora Postgre DB SQL
- Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora SQL Postgre DB
- Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)
- Considerazioni sulle prestazioni per l'utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL
- Monitoraggio del machine learning di Aurora

Requisiti per l'utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL

AWS i servizi di machine learning sono servizi gestiti che vengono configurati ed eseguiti nei propri ambienti di produzione. L'apprendimento automatico Aurora supporta l'integrazione con Amazon Comprehend SageMaker , AI e Amazon Bedrock. Prima di provare a configurare il cluster Aurora Postgre SQL DB per utilizzare l'apprendimento automatico di Aurora, assicurati di comprendere i seguenti requisiti e prerequisiti.

- I servizi Amazon Comprehend, SageMaker AI e Amazon Bedrock devono essere eseguiti nello stesso cluster Aurora Regione AWS Postgre DB. SQL Non puoi utilizzare i servizi Amazon Comprehend o SageMaker AI o Amazon Bedrock da un cluster Aurora Postgre SQL DB in un'altra regione.
- Se il cluster Aurora Postgre SQL DB si trova in un cloud pubblico virtuale diverso (VPC) basato sul VPC servizio Amazon rispetto SageMaker ai servizi Amazon Comprehend e AI, il gruppo Security deve consentire le connessioni in uscita al servizio VPC di machine learning Aurora

di destinazione. Per ulteriori informazioni, consulta Abilitazione della comunicazione di rete da Amazon Aurora ad altri servizi AWS.

- Per l' SageMaker intelligenza artificiale, i componenti di machine learning che desideri utilizzare per le inferenze devono essere configurati e pronti all'uso. Durante il processo di configurazione per il tuo cluster Aurora Postgre SQL DB, devi avere a disposizione l'Amazon Resource Name (ARN) dell' SageMaker endpoint AI. I data scientist del tuo team sono probabilmente i più adatti a gestire il lavoro con l' SageMaker intelligenza artificiale per preparare i modelli e gestire le altre attività simili. Per iniziare a usare Amazon SageMaker AI, consulta Get Started with Amazon SageMaker AI. Per ulteriori informazioni su inferenze ed endpoint, consulta Real-time inference (Inferenza in tempo reale).
- Per Amazon Bedrock, devi avere l'ID del modello dei modelli Bedrock che desideri utilizzare per le inferenze disponibili durante il processo di configurazione del tuo cluster Aurora Postgre DB. SQL I data scientist del tuo team sono probabilmente i più adatti a collaborare con Bedrock per decidere quali modelli utilizzare, perfezionarli se necessario e gestire altre attività simili. Per iniziare a usare Amazon Bedrock, consulta Come configurare Bedrock.
- Gli utenti di Amazon Bedrock devono richiedere l'accesso ai modelli prima di poterli utilizzare. Se vuoi aggiungere altri modelli per la generazione di testo, chat e immagini, devi richiedere l'accesso ai modelli in Amazon Bedrock. Per ulteriori informazioni, consulta Model access.

Funzionalità e limitazioni supportate dell'apprendimento automatico di Aurora con Aurora Postgre SQL

L'apprendimento automatico Aurora supporta qualsiasi endpoint di SageMaker intelligenza artificiale in grado di leggere e scrivere il formato value separato da virgole (CSV) tramite un valore di. `ContentType` `text/csv` Gli algoritmi di SageMaker intelligenza artificiale integrati che attualmente accettano questo formato sono i seguenti.

- Linear Learner
- Random Cut Forest
- XGBoost

  Per ulteriori informazioni su questi algoritmi, consulta Choose an Algorithm nella Amazon SageMaker AI Developer Guide.

Quando si utilizza Amazon Bedrock con l'apprendimento automatico Aurora, si applicano le seguenti limitazioni:

- Le funzioni definite dall'utente (UDFs) forniscono un modo nativo per interagire con Amazon Bedrock. UDFsNon hanno requisiti specifici di richiesta o risposta, quindi possono utilizzare qualsiasi modello.
- Puoi utilizzarlo UDFs per creare qualsiasi flusso di lavoro desiderato. Ad esempio, è possibile combinare primitive di base, ad esempio eseguire una query, recuperare dati, generare inferenze e scrivere su tabelle per servire direttamente le query. `pg_cron`
- UDFsnon supportano chiamate in batch o parallele.
- L'estensione Aurora Machine Learning non supporta interfacce vettoriali. Come parte dell'estensione, è disponibile una funzione per emettere gli incorporamenti della risposta del modello nel `float8[]` formato per archiviare tali incorporamenti in Aurora. Per ulteriori informazioni sull'utilizzo di, vedere. `float8[]` Utilizzo di Amazon Bedrock con il cluster Aurora SQL Postgre DB

Configurazione del cluster Aurora Postgre SQL DB per utilizzare l'apprendimento automatico Aurora

Affinché l'apprendimento automatico Aurora funzioni con il cluster Aurora Postgre SQL DB, è necessario creare un ruolo AWS Identity and Access Management (IAM) per ciascuno dei servizi che si desidera utilizzare. Il IAM ruolo consente al cluster Aurora Postgre SQL DB di utilizzare il servizio di machine learning Aurora per conto del cluster. Dovrai inoltre installare l'estensione di machine learning di Aurora. Nei seguenti argomenti sono disponibili le procedure di configurazione per ciascuno di questi servizi di machine learning di Aurora.

Argomenti

- Configurazione di Aurora Postgre per SQL l'utilizzo di Amazon Bedrock
- Configurazione di Aurora Postgre SQL per l'utilizzo di Amazon Comprehend
- Configurazione di Aurora Postgre per l'utilizzo di Amazon AI SQL SageMaker
  - Configurazione di Aurora Postgre per l'utilizzo di SQL Amazon S3 for AI (Advanced) SageMaker
- Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora Postgre per SQL l'utilizzo di Amazon Bedrock

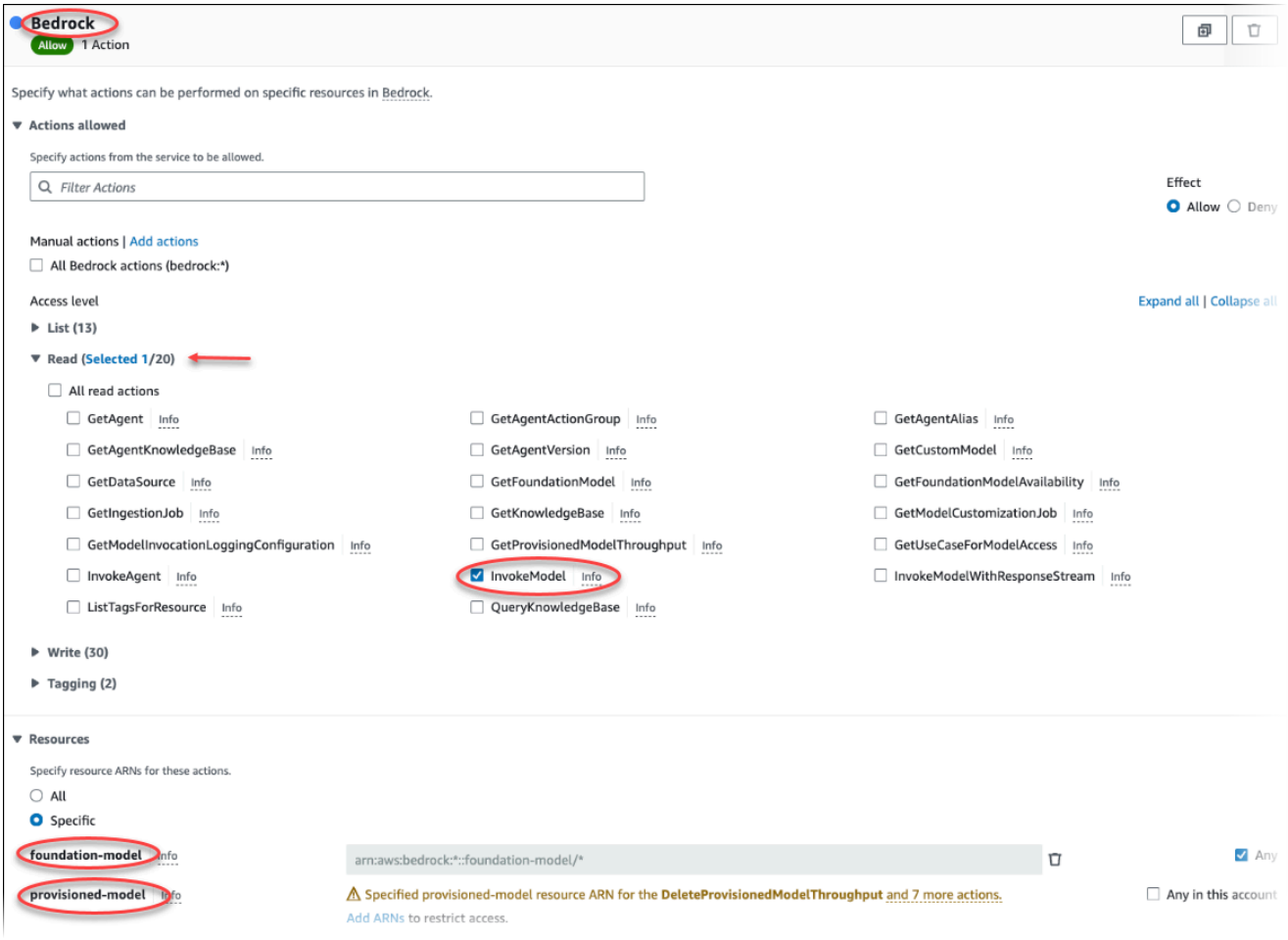
Nella procedura seguente, devi innanzitutto creare il IAM ruolo e la policy che autorizzano Aurora Postgre SQL a utilizzare Amazon Bedrock per conto del cluster. Quindi colleghi la policy a un IAM ruolo che il tuo cluster Aurora Postgre SQL DB utilizza per lavorare con Amazon Bedrock. Per semplificare, questa procedura utilizza il per completare tutte AWS Management Console le attività.

Per configurare il cluster Aurora Postgre SQL DB per utilizzare Amazon Bedrock

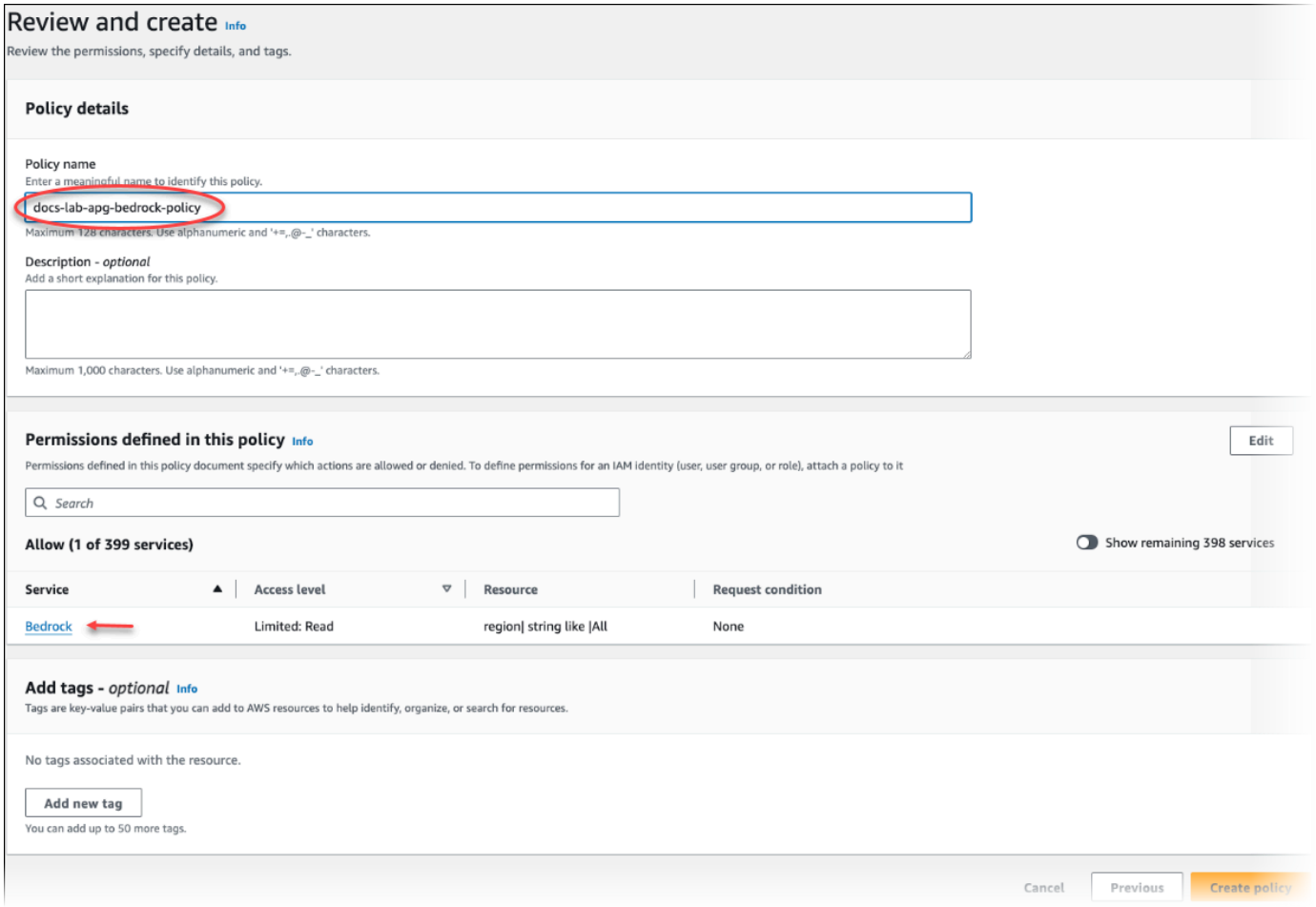
1. Accedi a AWS Management Console e apri la console all'indirizzo. IAM https://console.aws.amazon.com/iam/
2. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
3. Scegli Politiche (in Gestione degli accessi) nel menu AWS Identity and Access Management (IAM) della console.

    a. Scegli Create Policy (Crea policy). Nella pagina Visual Editor, scegli Service, quindi inserisci Bedrock nel campo Seleziona un servizio. Espandi il livello di accesso Read (Lettura). InvokeModelScegli tra le impostazioni di lettura di Amazon Bedrock.

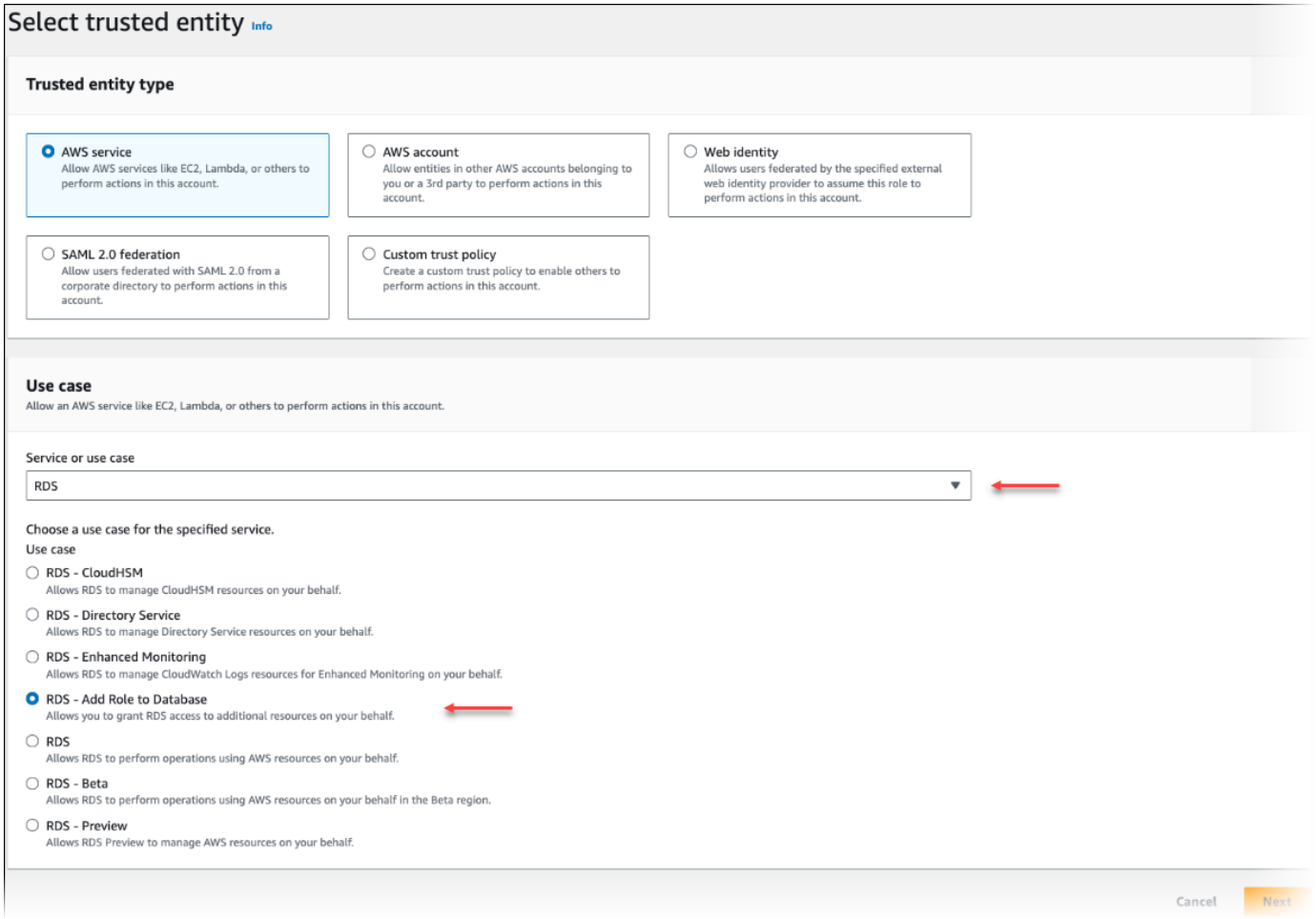
    b. Scegli il modello Foundation/Provisioned a cui desideri concedere l'accesso in lettura tramite la policy.



4. Scegli Next: Tags (Successivo: Tag) e definisci i tag (questo passaggio è facoltativo). Seleziona Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per la policy, come illustrato nell'immagine.

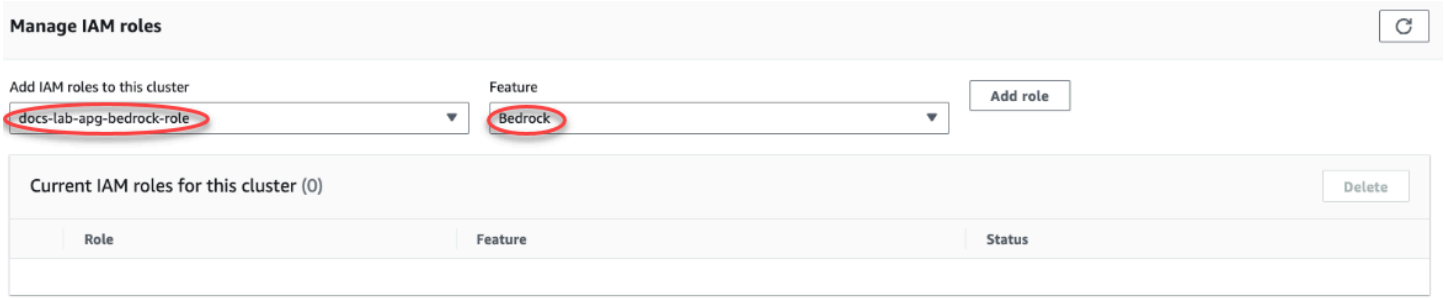


5. Scegli Create Policy (Crea policy). Viene visualizzato un avviso nella console quando la policy viene salvata e diventa disponibile nell'elenco delle policy.
6. Scegli Ruoli (in Gestione degli accessi) sulla IAM console.
7. Scegliere Crea ruolo.
8. Nella pagina Seleziona entità affidabile, scegli il riquadro del AWS servizio, quindi scegli RDSdi aprire il selettore.
9. Scegli RDS: aggiungi ruolo al database.



10. Scegli Next (Successivo). Nella pagina Add permissions (Aggiungi autorizzazioni), trova la policy creata nel passaggio precedente e selezionala tra quelle elencate. Scegli Next (Successivo).
11. Next: Review (Successivo: Rivedi). Inserisci un nome per il IAM ruolo e una descrizione.
12. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
13. Vai alla posizione Regione AWS in cui si trova il tuo cluster Aurora Postgre SQL DB.
14. Nel riquadro di navigazione, scegli Database, quindi scegli il cluster Aurora Postgre SQL DB che desideri utilizzare con Bedrock.
15. Scegli la scheda Connettività e sicurezza e scorri fino a trovare la sezione Gestisci IAM ruoli della pagina. Dal selettore Aggiungi IAM ruoli a questo cluster, scegli il ruolo che hai creato nei passaggi precedenti. Nel selettore Feature, scegli Bedrock, quindi scegli Aggiungi ruolo.

Il ruolo (con la relativa policy) è associato al cluster Aurora SQL Postgre DB. Al termine del processo, il ruolo viene elencato nell'elenco IAM Ruoli correnti per questo cluster, come illustrato di seguito.



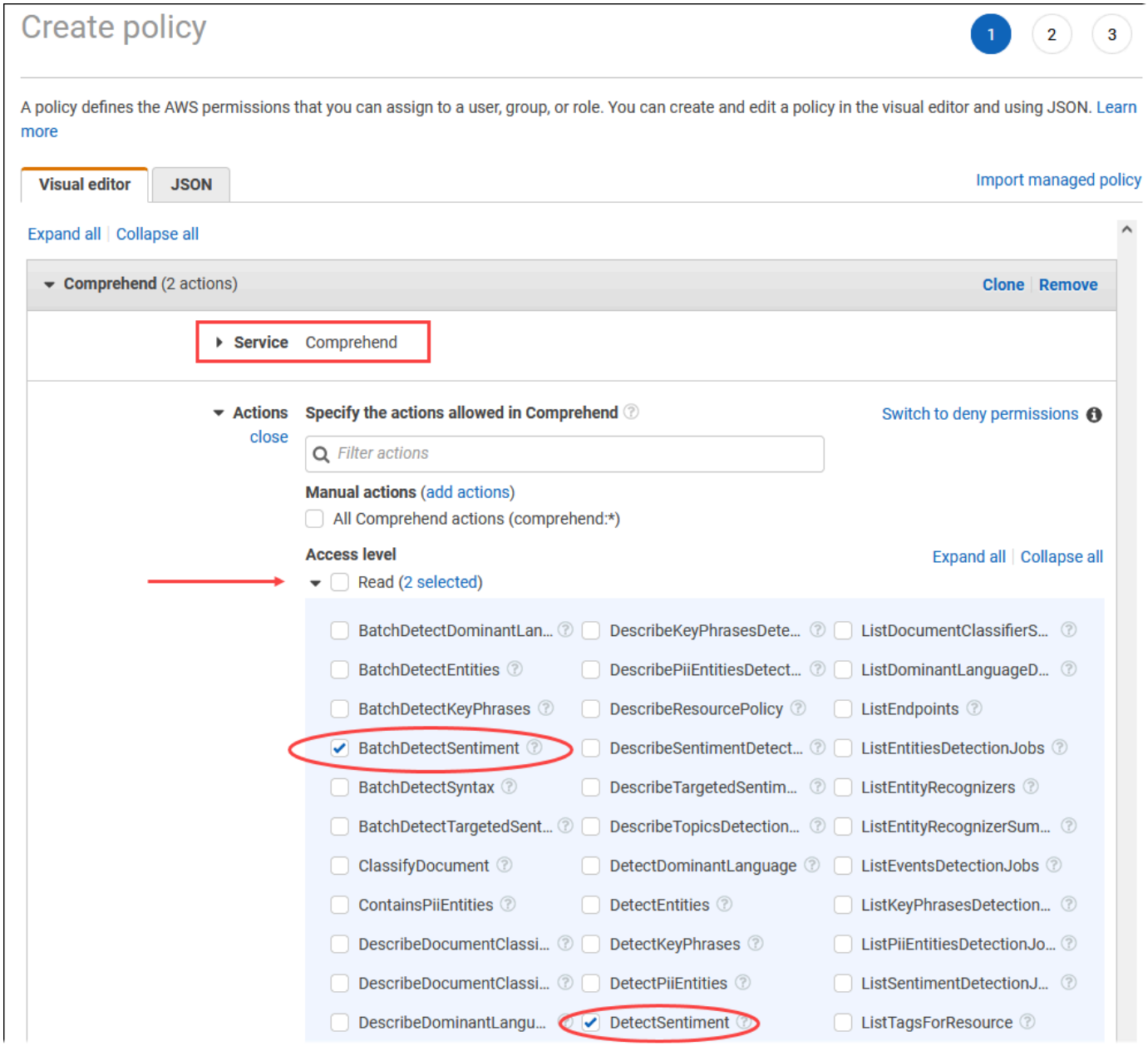
La IAM configurazione per Amazon Bedrock è completa. Continua a configurare Aurora Postgre SQL per l'utilizzo con Aurora Machine Learning installando l'estensione come descritto in Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora Postgre SQL per l'utilizzo di Amazon Comprehend

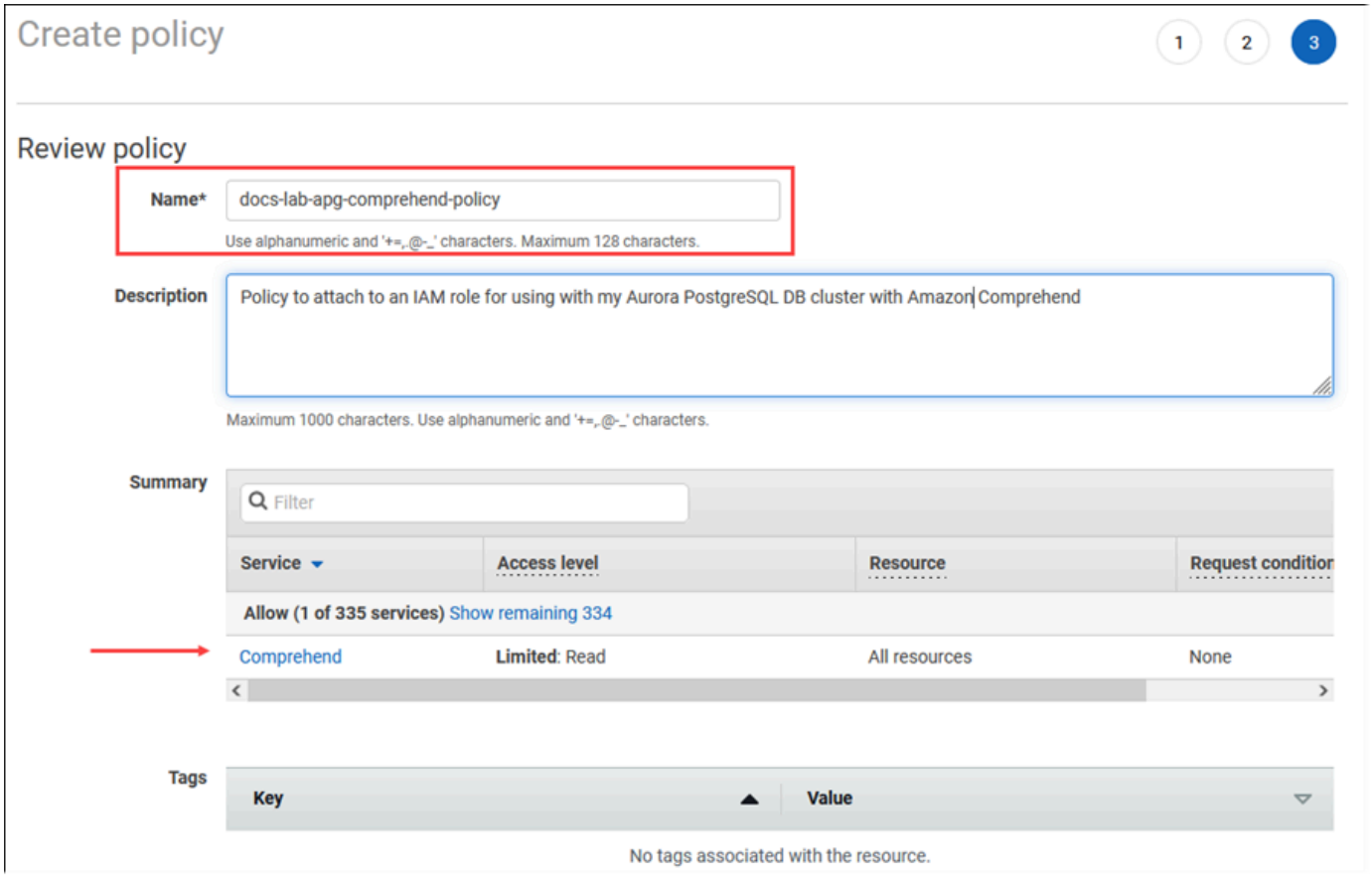
Nella procedura seguente, devi innanzitutto creare il IAM ruolo e la policy che SQL autorizzano Aurora Postgre a utilizzare Amazon Comprehend per conto del cluster. Quindi colleghi la policy a un IAM ruolo che il tuo cluster Aurora Postgre SQL DB utilizza per lavorare con Amazon Comprehend. Per semplicità, questa procedura utilizza il per completare tutte le attività. AWS Management Console

Per configurare il cluster Aurora Postgre SQL DB per utilizzare Amazon Comprehend

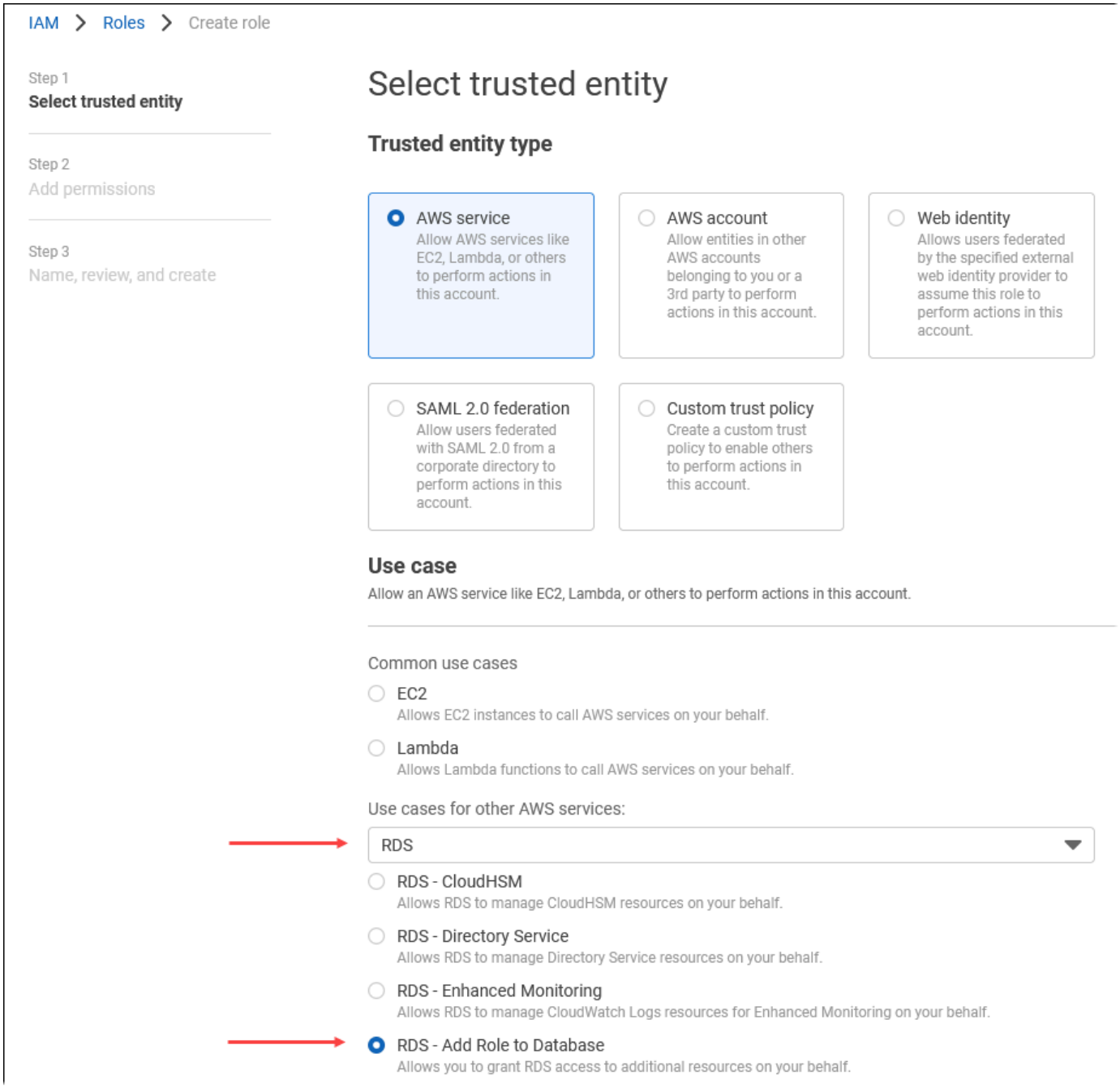
1. Accedi a AWS Management Console e apri la console all'indirizzo. IAM https://console.aws.amazon.com/iam/
2. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
3. Scegli Politiche (in Gestione degli accessi) nel menu AWS Identity and Access Management (IAM) della console.



4. Scegli Create Policy (Crea policy). Nella pagina Visual editor (Editor visivo) scegli Service (Servizio), quindi immetti Comprehend nel campo Select a service (Seleziona un servizio). Espandi il livello di accesso Read (Lettura). Scegli BatchDetectSentimente tra le DetectSentimentimpostazioni di lettura di Amazon Comprehend.
5. Scegli Next: Tags (Successivo: Tag) e definisci i tag (questo passaggio è facoltativo). Seleziona Next: Review (Successivo: Rivedi). Immetti un nome e una descrizione per la policy, come illustrato nell'immagine.



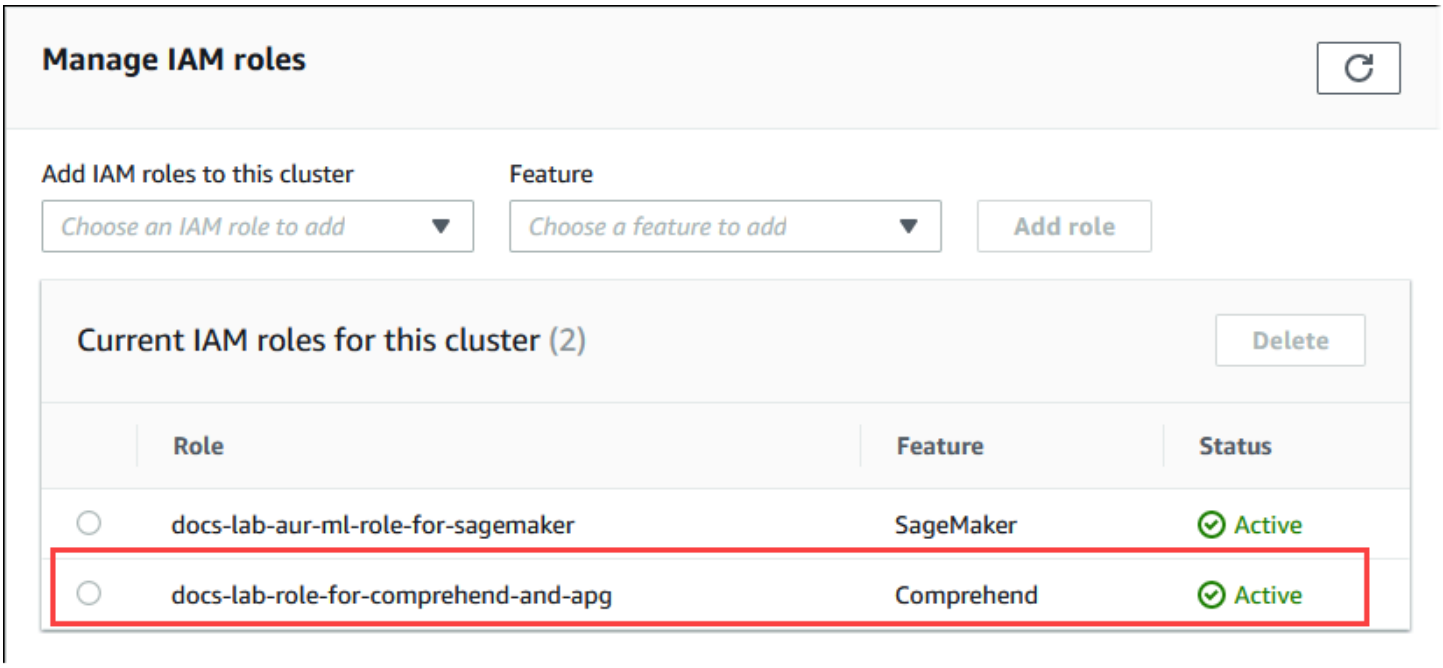
6. Scegli Create Policy (Crea policy). Viene visualizzato un avviso nella console quando la policy viene salvata e diventa disponibile nell'elenco delle policy.
7. Scegli Ruoli (in Gestione degli accessi) sulla IAM console.
8. Scegliere Crea ruolo.
9. Nella pagina Seleziona entità affidabile, scegli il riquadro del AWS servizio, quindi scegli RDSdi aprire il selettore.
10. Scegli RDS: aggiungi ruolo al database.



11. Scegli Next (Successivo). Nella pagina Add permissions (Aggiungi autorizzazioni), trova la policy creata nel passaggio precedente e selezionala tra quelle elencate. Seleziona Next (Successivo).
12. Next: Review (Successivo: Rivedi). Inserisci un nome per il IAM ruolo e una descrizione.
13. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
14. Vai alla posizione Regione AWS in cui si trova il tuo cluster Aurora Postgre SQL DB.
15. Nel riquadro di navigazione, scegli Databases, quindi scegli il cluster SQL DB Aurora Postgre che desideri utilizzare con Amazon Comprehend.

16. Scegli la scheda Connettività e sicurezza e scorri fino a trovare la sezione Gestisci IAM ruoli della pagina. Dal selettore Aggiungi IAM ruoli a questo cluster, scegli il ruolo che hai creato nei passaggi precedenti. Nel selettore Feature, scegliete Comprehend, quindi scegliete Aggiungi ruolo.

Il ruolo (con la relativa policy) è associato al cluster Aurora SQL Postgre DB. Al termine del processo, il ruolo viene elencato nell'elenco IAM Ruoli correnti per questo cluster, come illustrato di seguito.



La IAM configurazione per Amazon Comprehend è completa. Continua a configurare Aurora Postgre SQL per l'utilizzo con Aurora Machine Learning installando l'estensione come descritto in Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora Postgre per l'utilizzo di Amazon AI SQL SageMaker

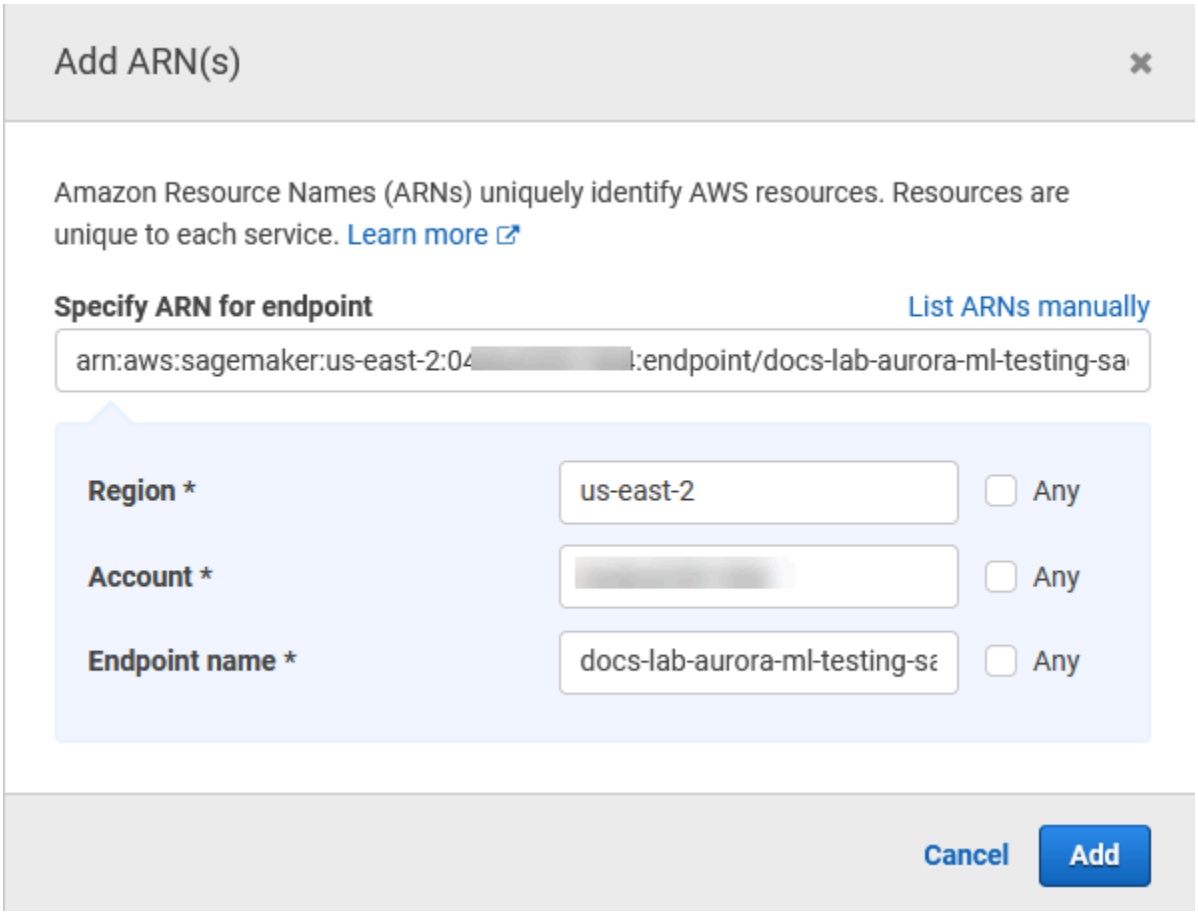
Prima di poter creare la IAM policy e il ruolo per il tuo cluster Aurora Postgre SQL DB, devi avere a disposizione la configurazione del modello di SageMaker intelligenza artificiale e l'endpoint.

Per configurare il cluster Aurora Postgre SQL DB per l'utilizzo dell'IA SageMaker

1. Accedi a AWS Management Console e apri la console all'IAMindirizzo. https://console.aws.amazon.com/iam/
2. Scegli Politiche (in Gestione degli accessi) nel menu AWS Identity and Access Management (IAM) della console, quindi scegli Crea politica. Nell'editor visivo, scegli SageMakeril servizio.

Per Azioni, apri il selettore di lettura (in Livello di accesso) e scegli InvokeEndpoint. Viene visualizzata un'icona di avviso quando si esegue questa operazione.

3. Apri il selettore Risorse e scegli il link Aggiungi ARN per limitare l'accesso sotto Specificare la risorsa dell'endpoint ARN per l'azione. InvokeEndpoint
4. Inserisci le Regione AWS tue risorse SageMaker AI e il nome del tuo endpoint. Il tuo AWS account è precompilato.



5. Scegli Add (Aggiungi) per salvare. Scegli Next: Tags (Successivo: Tag) e Next: Review (Successivo: Rivedi) per passare all'ultima pagina del processo di creazione della policy.
6. Immetti un nome e una descrizione per la policy e quindi scegli Create policy (Crea policy). La policy viene creata e aggiunta all'elenco delle policy. Quindi, viene visualizzato un avviso nella console.
7. Nella IAM console, scegli Ruoli.
8. Scegliere Crea ruolo.
9. Nella pagina Seleziona entità attendibile, scegli il riquadro del AWS servizio, quindi scegli RDSdi aprire il selettore.

10. Scegli RDS: aggiungi ruolo al database.
11. Scegli Next (Successivo). Nella pagina Add permissions (Aggiungi autorizzazioni), trova la policy creata nel passaggio precedente e selezionala tra quelle elencate. Seleziona Next (Successivo).
12. Next: Review (Successivo: Rivedi). Inserisci un nome per il IAM ruolo e una descrizione.
13. Apri la RDS console Amazon all'indirizzo https://console.aws.amazon.com/rds/.
14. Vai alla posizione Regione AWS in cui si trova il tuo cluster Aurora Postgre SQL DB.
15. Nel riquadro di navigazione, scegli Database, quindi scegli il cluster SQL DB Aurora Postgre che desideri utilizzare con AI. SageMaker
16. Scegli la scheda Connettività e sicurezza e scorri fino a trovare la sezione Gestisci IAM ruoli della pagina. Dal selettore Aggiungi IAM ruoli a questo cluster, scegli il ruolo che hai creato nei passaggi precedenti. Nel selettore Funzionalità, scegli SageMaker AI, quindi scegli Aggiungi ruolo.

Il ruolo (con la relativa policy) è associato al cluster Aurora SQL Postgre DB. Al termine del processo, il ruolo viene elencato nell'elenco IAM Ruoli correnti per questo cluster.

La IAM configurazione per l' SageMaker IA è completa. Continua a configurare Aurora Postgre SQL per l'utilizzo con Aurora Machine Learning installando l'estensione come descritto in. Installazione dell'estensione di machine learning di Aurora

Configurazione di Aurora Postgre per l'utilizzo di SQL Amazon S3 for AI (Advanced) SageMaker

Per utilizzare l' SageMaker intelligenza artificiale con i tuoi modelli anziché utilizzare i componenti predefiniti forniti dall' SageMaker IA, devi configurare un bucket Amazon Simple Storage Service (Amazon S3) per il cluster Aurora Postgre DB da utilizzare. SQL Questa è un'attività avanzata non completamente documentata in questa Guida per l'utente di Amazon Aurora. Il processo generale è lo stesso dell'integrazione del supporto per l'IA, come segue. SageMaker

1. Crea la IAM politica e il ruolo per Amazon S3.
2. Aggiungi il IAM ruolo e l'importazione o l'esportazione di Amazon S3 come funzionalità nella scheda Connettività e sicurezza del tuo cluster Aurora Postgre DB. SQL
3. Aggiungi il ruolo al gruppo ARN di parametri del cluster DB personalizzato per il tuo cluster Aurora DB.

Per informazioni di base sull'utilizzo, consulta Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced).

Installazione dell'estensione di machine learning di Aurora

Le estensioni di machine learning Aurora `aws_ml 1.0` offrono due funzioni che puoi usare per richiamare Amazon Comprehend, i servizi SageMaker AI e `aws_ml 2.0` offrono due funzioni aggiuntive che puoi usare per richiamare i servizi Amazon Bedrock. L'installazione di queste estensioni sul cluster Aurora Postgre SQL DB crea anche un ruolo amministrativo per la funzionalità.

> **Note**
>
> L'utilizzo di queste funzioni dipende dal completamento della IAM configurazione del servizio di machine learning Aurora (Amazon Comprehend, SageMaker AI, Amazon Bedrock), come descritto in. Configurazione del cluster Aurora Postgre SQL DB per utilizzare l'apprendimento automatico Aurora

- aws_comprehend.detect_sentiment — Utilizzi questa funzione per applicare l'analisi del sentimento al testo archiviato nel database del cluster Aurora Postgre DB. SQL
- aws_sagemaker.invoke_endpoint: utilizzi questa funzione nel codice per comunicare con l'endpoint AI del tuo cluster. SQL SageMaker
- aws_bedrock.invoke_model — Utilizzi questa funzione nel codice per comunicare con i modelli Bedrock del tuo cluster. SQL La risposta di questa funzione sarà nel formato di aTEXT, quindi se un modello risponde nel formato di un JSON corpo, l'output di questa funzione verrà inoltrato nel formato di una stringa all'utente finale.
- aws_bedrock.invoke_model_get_embeddings — Utilizzi questa funzione nel codice per richiamare i modelli Bedrock che restituiscono gli incorporamenti di output all'interno di una risposta. SQL JSON Questo può essere sfruttato quando si desidera estrarre gli incorporamenti direttamente associati alla chiave json per semplificare la risposta con qualsiasi flusso di lavoro autogestito.

Per installare l'estensione Aurora Machine Learning nel cluster Aurora Postgre DB SQL

- Utilizzalo `psql` per connetterti all'istanza writer del tuo cluster Aurora SQL Postgre DB. Collega il database specifico in cui installare l'estensione `aws_ml`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=labdb
```

```
labdb=> CREATE EXTENSION IF NOT EXISTS aws_ml CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
labdb=>
```

L'installazione delle `aws_ml` estensioni crea anche il ruolo `aws_ml` amministrativo e tre nuovi schemi, come segue.

- `aws_comprehend`: schema del servizio Amazon Comprehend e origine della funzione `detect_sentiment` (`aws_comprehend.detect_sentiment`).
- `aws_sagemaker`— Schema per il servizio di SageMaker intelligenza artificiale e origine della `invoke_endpoint` funzione (`aws_sagemaker.invoke_endpoint`).
- `aws_bedrock`— Schema per il servizio Amazon Bedrock e origine delle `invoke_model_get_embeddings(aws_bedrock.invoke_model_get_embeddings)` funzioni `invoke_model(aws_bedrock.invoke_model)` e.

Al `rds_superuser` ruolo viene assegnato il ruolo `aws_ml` amministrativo ed è composto da questi tre schemi `OWNER` di apprendimento automatico Aurora. Per consentire ad altri utenti del database di accedere alle funzioni di machine learning di Aurora, `rds_superuser` deve concedere i privilegi `EXECUTE` per le funzioni di machine learning di Aurora. Per impostazione predefinita, i privilegi `EXECUTE` sono revocati da `PUBLIC` per le funzioni nei due schemi di machine learning di Aurora.

In una configurazione di database multi-tenant, puoi impedire ai tenant di accedere alle funzioni di machine learning di Aurora utilizzando `REVOKE USAGE` nello specifico schema di machine learning di Aurora che desideri proteggere.

Utilizzo di Amazon Bedrock con il cluster Aurora SQL Postgre DB

Per Aurora PostgreSQL, l'apprendimento automatico di Aurora offre la seguente funzione Amazon Bedrock per lavorare con i dati di testo. Questa funzione è disponibile solo dopo aver installato l'estensione `aws_ml` 2.0 e aver completato tutte le procedure di configurazione. Per ulteriori informazioni, consulta Configurazione del cluster Aurora Postgre SQL DB per utilizzare l'apprendimento automatico Aurora.

aws_bedrock.invoke_model

Questa funzione accetta il testo formattato JSON come input e lo elabora per una varietà di modelli ospitati su Amazon Bedrock e recupera la risposta JSON testuale dal modello. Questa

risposta può contenere testo, immagini o incorporamenti. Di seguito è riportato un riepilogo della documentazione della funzione.

```
aws_bedrock.invoke_model(
   IN model_id      varchar,
   IN content_type  text,
   IN accept_type   text,
   IN model_input   text,
   OUT model_output varchar)
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `model_id`— Identificatore del modello.
- `content_type`— Il tipo di richiesta al modello di Bedrock.
- `accept_type`— Il tipo di risposta che ci si aspetta dal modello di Bedrock. Di solito applicazione/JSON per la maggior parte dei modelli.
- `model_input`— Richieste; un insieme specifico di input per il modello nel formato specificato da content_type. Per ulteriori informazioni sul formato/struttura della richiesta accettato dal modello, consultate Parametri di inferenza per i modelli di base.
- `model_output`— L'output del modello Bedrock come testo.

L'esempio seguente mostra come richiamare un modello Anthropic Claude 2 per Bedrock utilizzando invoke_model.

Example Esempio: una semplice query che utilizza le funzioni di Amazon Bedrock

```
SELECT aws_bedrock.invoke_model (
    model_id    := 'anthropic.claude-v2',
    content_type:= 'application/json',
    accept_type := 'application/json',
    model_input := '{"prompt": "\n\nHuman: You are a helpful assistant that answers
 questions directly and only using the information provided in the context below.
\nDescribe the answer
    in detail.\n\nContext: %s \n\nQuestion: %s \n
\nAssistant:","max_tokens_to_sample":4096,"temperature":0.5,"top_k":250,"top_p":0.5,"stop_seque
[]}'
);
```

aws_bedrock.invoke_model_get_embeddings

L'output del modello può indicare incorporamenti vettoriali in alcuni casi. Dato che la risposta varia in base al modello, è possibile sfruttare un'altra funzione invoke_model_get_embeddings che funziona esattamente come invoke_model ma restituisce gli incorporamenti specificando la chiave json appropriata.

```
aws_bedrock.invoke_model_get_embeddings(
   IN model_id      varchar,
   IN content_type  text,
   IN json_key      text,
   IN model_input   text,
   OUT model_output float8[])
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `model_id`— Identificatore del modello.
- `content_type`— Il tipo di richiesta al modello di Bedrock. Qui, accept_type è impostato sul valore predefinito. `application/json`
- `model_input`— Richieste; un set specifico di input per il modello nel formato specificato da content_type. Per ulteriori informazioni sul formato/struttura della richiesta accettato dal modello, vedere Parametri di inferenza per i modelli di base.
- `json_key`— Riferimento al campo da cui estrarre l'incorporamento. Questo può variare se il modello di incorporamento cambia.
- `model_output`— L'output del modello Bedrock sotto forma di matrice di incorporamenti con decimali a 16 bit.

L'esempio seguente mostra come generare un incorporamento utilizzando il modello Titan Embeddings G1 — Text embedding per la frase Postgre I/O monitoring views. SQL

Example Esempio: una semplice query che utilizza le funzioni di Amazon Bedrock

```
SELECT aws_bedrock.invoke_model_get_embeddings(
   model_id      := 'amazon.titan-embed-text-v1',
   content_type  := 'application/json',
   json_key      := 'embedding',
   model_input   := '{ "inputText": "PostgreSQL I/O monitoring views"}') AS embedding;
```

Utilizzo di Amazon Comprehend con il cluster Aurora Postgre DB SQL

Per Aurora PostgreSQL, l'apprendimento automatico di Aurora fornisce la seguente funzione Amazon Comprehend per lavorare con i dati di testo. Questa funzione è disponibile solo dopo aver installato l'estensione `aws_ml` e completato tutte le procedure di configurazione. Per ulteriori informazioni, consulta Configurazione del cluster Aurora Postgre SQL DB per utilizzare l'apprendimento automatico Aurora.

aws_comprehend.detect_sentiment

Questa funzione utilizza il testo come input e valuta se ha un intento emotivo positivo, negativo, neutro o misto. Genera questo sentiment insieme a un livello di confidenza per la valutazione. Di seguito è riportato un riepilogo della documentazione della funzione.

```
aws_comprehend.detect_sentiment(
   IN input_text varchar,
   IN language_code varchar,
   IN max_rows_per_batch int,
   OUT sentiment varchar,
   OUT confidence real)
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `input_text`: il testo da valutare per assegnare il sentiment (negativo, positivo, neutro, misto).
- `language_code`— La lingua dell'`input_text`identificato utilizzando l'identificatore ISO 639-1 a 2 lettere con sottotag regionale (se necessario) o il codice 639-2 a tre lettere, a seconda dei casi. ISO Ad esempio, `en` è il codice per l'inglese, `zh` è il codice per il cinese semplificato. Per ulteriori informazioni, consulta Lingue supportate nella Guida per gli sviluppatori di Amazon Comprehend.
- `max_rows_per_batch`: il numero massimo di righe per batch per l'elaborazione in modalità batch. Per ulteriori informazioni, consulta Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora.
- `sentiment`— Il sentimento del testo di input, identificato come,, o. POSITIVE NEGATIVE NEUTRAL MIXED
- `confidence`: il livello di confidenza della precisione del `sentiment` specificato. I valori sono compresi tra 0,0 e 1,0.

Di seguito sono riportati alcuni esempi di come utilizzare questa funzione.

Example Esempio: una semplice query che utilizza le funzioni di Amazon Comprehend

Di seguito è riportato l'esempio di una semplice query che richiama la funzione per valutare la soddisfazione del cliente nei confronti del team di supporto. Supponi di avere una tabella di database (`support`) che archivia il feedback dei clienti dopo ogni richiesta di assistenza. Questa query di esempio applica la funzione `aws_comprehend.detect_sentiment` al testo nella colonna `feedback` della tabella e restituisce il sentiment e il livello di confidenza del sentiment. Inoltre i risultati vengono restituiti in ordine decrescente.

```
SELECT feedback, s.sentiment,s.confidence
    FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
    ORDER BY s.confidence DESC;
 feedback                          | sentiment | confidence
 ------------------------------------------------------------+-----------+------------
 Thank you for the excellent customer support!              | POSITIVE  |   0.999771
 The latest version of this product stinks!                 | NEGATIVE  |   0.999184
 Your support team is just awesome! I am blown away.        | POSITIVE  |   0.997774
 Your product is too complex, but your support is great.    | MIXED     |   0.957958
 Your support tech helped me in fifteen minutes.            | POSITIVE  |   0.949491
 My problem was never resolved!                             | NEGATIVE  |   0.920644
 When will the new version of this product be released?     | NEUTRAL   |   0.902706
 I cannot stand that chatbot.                               | NEGATIVE  |   0.895219
 Your support tech talked down to me.                       | NEGATIVE  |   0.868598
 It took me way too long to get a real person.              | NEGATIVE  |   0.481805

 (10 rows)
```

Per evitare che ti venga addebitato il rilevamento del sentiment più di una volta per riga della tabella, puoi materializzare i risultati. Esegui questa operazione sulle righe di interesse. Ad esempio, le note del medico vengono aggiornate in modo che solo quelle in francese (`fr`) utilizzino la funzione di rilevamento del sentiment.

```
UPDATE clinician_notes
SET sentiment = (aws_comprehend.detect_sentiment (french_notes, 'fr')).sentiment,
    confidence = (aws_comprehend.detect_sentiment (french_notes, 'fr')).confidence
WHERE
    clinician_notes.french_notes IS NOT NULL AND
    LENGTH(TRIM(clinician_notes.french_notes)) > 0 AND
    clinician_notes.sentiment IS NULL;
```

Per ulteriori informazioni sull'ottimizzazione delle chiamate di funzione, consulta Considerazioni sulle prestazioni per l'utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL.

Utilizzo dell' SageMaker intelligenza artificiale con il cluster Aurora SQL Postgre DB

Dopo aver configurato l'ambiente di SageMaker intelligenza artificiale e aver effettuato l'integrazione con Aurora SQL Postgre come descritto Configurazione di Aurora Postgre per l'utilizzo di Amazon AI SQL SageMaker in, puoi richiamare le operazioni utilizzando la funzione. `aws_sagemaker.invoke_endpoint` La funzione `aws_sagemaker.invoke_endpoint` si connette a un solo endpoint del modello nella stessa Regione AWS. Se la tua istanza di database ha repliche multiple, Regioni AWS assicurati di configurare e distribuire ogni modello di intelligenza artificiale su tutti. SageMaker Regione AWS

Le chiamate a `aws_sagemaker.invoke_endpoint` vengono autenticate utilizzando il IAM ruolo che hai impostato per associare il cluster Aurora SQL Postgre DB al servizio AI e SageMaker all'endpoint che hai fornito durante il processo di configurazione. SageMaker Gli endpoint del modello AI sono limitati a un singolo account e non sono pubblici. `endpoint_name`URLNon contiene l'ID dell'account. SageMaker L'IA determina l'ID dell'account dal token di autenticazione fornito dal IAM ruolo SageMaker AI dell'istanza del database.

aws_sagemaker.invoke_endpoint

Questa funzione accetta l'endpoint SageMaker AI come input e il numero di righe che devono essere elaborate come batch. Prende anche come input i vari parametri previsti dall'endpoint del modello SageMaker AI. La documentazione di riferimento della funzione è indicata di seguito.

```
aws_sagemaker.invoke_endpoint(
  IN endpoint_name varchar,
  IN max_rows_per_batch int,
  VARIADIC model_input "any",
  OUT model_output varchar
  )
```

Gli input e gli output di questa funzione sono indicati di seguito.

- `endpoint_name`— Un endpoint URL indipendente. Regione AWS

- `max_rows_per_batch`: il numero massimo di righe per batch per l'elaborazione in modalità batch. Per ulteriori informazioni, consulta Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora.
- `model_input`: uno o più parametri di input per il modello. Questi possono essere qualsiasi tipo di dati necessario al modello di SageMaker intelligenza artificiale. Postgre SQL consente di specificare fino a 100 parametri di input per una funzione. I tipi di dati dell'array devono essere unidimensionali, ma possono contenere tutti gli elementi previsti dal modello di intelligenza artificiale. SageMaker Il numero di input per un modello SageMaker AI è limitato solo dal limite di dimensione dei messaggi SageMaker AI di 6 MB.
- `model_output`— L'output del modello SageMaker AI come testo.

Creazione di una funzione definita dall'utente per richiamare un SageMaker modello di intelligenza artificiale

Crea una funzione definita dall'utente separata da chiamare `aws_sagemaker.invoke_endpoint` per ciascuno dei tuoi SageMaker modelli di intelligenza artificiale. La funzione definita dall'utente rappresenta l'endpoint di SageMaker intelligenza artificiale che ospita il modello. La funzione `aws_sagemaker.invoke_endpoint` viene eseguita all'interno della funzione definita dall'utente. Le funzioni definite dall'utente offrono molti vantaggi:

- Puoi dare un nome al tuo modello di SageMaker intelligenza artificiale invece di richiamare solo tutti `aws_sagemaker.invoke_endpoint` i tuoi modelli di SageMaker intelligenza artificiale.
- È possibile specificare l'endpoint del modello URL in un'unica posizione nel codice SQL dell'applicazione.
- Puoi controllare i privilegi `EXECUTE` per ogni funzione di machine learning di Aurora in modo indipendente.
- È possibile dichiarare i tipi di input e output del modello utilizzando SQL i tipi. SQLimpone il numero e il tipo di argomenti passati al modello di SageMaker intelligenza artificiale ed esegue la conversione dei tipi, se necessario. L'utilizzo dei SQL tipi si `SQL NULL` tradurrà anche nel valore predefinito appropriato previsto dal modello di SageMaker intelligenza artificiale.
- Puoi ridurre la dimensione massima del batch se desideri restituire le prime righe un po' più velocemente.

Per specificare una funzione definita dall'utente, utilizzate l'istruzione SQL `CREATE FUNCTION` Data Definition Language (DDL). Quando definisci la funzione, specifica quanto segue:

- I parametri di input per il modello.
- L'endpoint SageMaker AI specifico da richiamare.
- Il tipo restituito.

La funzione definita dall'utente restituisce l'inferenza calcolata dall'endpoint SageMaker AI dopo aver eseguito il modello sui parametri di input. L'esempio seguente crea una funzione definita dall'utente per un modello di SageMaker intelligenza artificiale con due parametri di input.

```
CREATE FUNCTION classify_event (IN arg1 INT, IN arg2 DATE, OUT category INT)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', NULL,
        arg1, arg2                          -- model inputs are separate arguments
        )::INT                              -- cast the output to INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Tieni presente quanto segue:

- L'input della funzione `aws_sagemaker.invoke_endpoint` può essere uno o più parametri di qualsiasi tipo di dati.
- Questo esempio utilizza un tipo INT di output. Se esegui il cast dell'output da un `varchar` tipo a un tipo diverso, deve essere eseguito il cast su un tipo scalare SQL integrato di Postgre come`INTEGER`,, `REAL` o. `FLOAT` `NUMERIC` Per ulteriori informazioni su questi tipi, consulta Tipi di dati nella documentazione di Postgre. SQL
- Specifica `PARALLEL SAFE` per abilitare l'esecuzione di query parallele. Per ulteriori informazioni, consulta Miglioramento dei tempi di risposta con l'elaborazione di query parallela.
- Specificare `COST 5000` per stimare il costo di esecuzione della funzione. Utilizza un numero positivo che fornisce i costi di esecuzione stimati della funzione, in unità di `cpu_operator_cost`.

Passare un array come input a un modello di intelligenza artificiale SageMaker

La `aws_sagemaker.invoke_endpoint` funzione può avere fino a 100 parametri di input, che è il limite per le funzioni Postgre. SQL Se il modello SageMaker AI richiede più di 100 parametri dello stesso tipo, passa i parametri del modello come array.

L'esempio seguente definisce una funzione che passa un array come input al modello di regressione SageMaker AI. L'output viene trasmesso a un valore `REAL`.

```
CREATE FUNCTION regression_model (params REAL[], OUT estimate REAL)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
      'sagemaker_model_endpoint_name',
      NULL,
      params
      )::REAL
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Specificazione della dimensione del batch quando si richiama un modello di intelligenza artificiale SageMaker

L'esempio seguente crea una funzione definita dall'utente per un modello di SageMaker intelligenza artificiale che imposta la dimensione del batch predefinita su. NULL La funzione consente inoltre di fornire una dimensione batch diversa quando viene invocata.

```
CREATE FUNCTION classify_event (
    IN event_type INT, IN event_day DATE, IN amount REAL, -- model inputs
    max_rows_per_batch INT DEFAULT NULL,  -- optional batch size limit
    OUT category INT)                     -- model output
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', max_rows_per_batch,
        event_type, event_day, COALESCE(amount, 0.0)
        )::INT              -- casts output to type INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Tieni presente quanto segue:

- Utilizza il parametro `max_rows_per_batch` opzionale per fornire il controllo del numero di righe per una chiamata della funzione in modalità batch. Se si utilizza un valore diNULL, l'ottimizzatore di query sceglie automaticamente la dimensione massima del batch. Per ulteriori informazioni, consulta Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora.
- Per impostazione predefinita, il passaggio NULL come valore di un parametro viene tradotto in una stringa vuota prima di passare all'AI SageMaker . Per questo esempio i tipi di input sono diversi.
- Se si dispone di un input non di testo o di un input di testo il cui valore predefinito deve essere diverso da una stringa vuota, utilizza l'istruzione `COALESCE`. `COALESCE`Da utilizzare per NULL tradurre nel valore sostitutivo nullo desiderato nella chiamata

aaws_sagemaker.invoke_endpoint. Per il amount parametro di questo esempio, un NULL valore viene convertito in 0,0.

Invocando un modello di SageMaker intelligenza artificiale con più output

L'esempio seguente crea una funzione definita dall'utente per un modello di SageMaker intelligenza artificiale che restituisce più output. La funzione deve eseguire il cast dell'output della funzione aws_sagemaker.invoke_endpoint a un tipo di dati corrispondente. Ad esempio, è possibile utilizzare il tipo di SQL punto Postgre integrato per le coppie (x, y) o un tipo composito definito dall'utente.

Questa funzione definita dall'utente restituisce valori da un modello che restituisce più output utilizzando un tipo composito per gli output.

```
CREATE TYPE company_forecasts AS (
    six_month_estimated_return real,
    one_year_bankruptcy_probability float);
CREATE FUNCTION analyze_company (
    IN free_cash_flow NUMERIC(18, 6),
    IN debt NUMERIC(18,6),
    IN max_rows_per_batch INT DEFAULT NULL,
    OUT prediction company_forecasts)
AS $$
    SELECT (aws_sagemaker.invoke_endpoint('endpt_name',
       max_rows_per_batch,free_cash_flow, debt))::company_forecasts;


$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Per il tipo composito, utilizza i campi nello stesso ordine in cui vengono visualizzati nell'output del modello ed esegui il cast dell'output del aws_sagemaker.invoke_endpoint al tipo composito. Il chiamante può estrarre i singoli campi per nome o con la notazione «.*» di SQL Postgre.

Esportazione di dati in Amazon S3 SageMaker per la formazione su modelli di intelligenza artificiale (Advanced)

Ti consigliamo di acquisire familiarità con l'apprendimento automatico e l' SageMaker intelligenza artificiale di Aurora utilizzando gli algoritmi e gli esempi forniti anziché provare ad addestrare i tuoi modelli. Per ulteriori informazioni, consulta Get Started with Amazon SageMaker AI

Per addestrare i modelli di SageMaker intelligenza artificiale, esporti i dati in un bucket Amazon S3. Il bucket Amazon S3 viene utilizzato dall' SageMaker intelligenza artificiale per addestrare il modello prima che venga distribuito. Puoi interrogare i dati da un cluster Aurora Postgre SQL DB e salvarli direttamente in file di testo archiviati in un bucket Amazon S3. Quindi l' SageMaker intelligenza artificiale utilizza i dati del bucket Amazon S3 per la formazione. Per ulteriori informazioni sulla formazione dei modelli di SageMaker intelligenza artificiale, consulta Addestrare un modello con Amazon SageMaker AI.

> **Note**
>
> Quando crei un bucket Amazon S3 per la formazione sui modelli di SageMaker intelligenza artificiale o il punteggio in batch, utilizzalo nel nome del bucket `sagemaker` Amazon S3. Per ulteriori informazioni, consulta Specificare un bucket Amazon S3 per caricare set di dati di formazione e archiviare i dati di output nella Amazon SageMaker AI Developer Guide.

Per ulteriori informazioni sull'esportazione dei dati, consulta Esportazione di dati da del cluster di database Aurora PostgreSQLRDS per PostgreSQL a Amazon S3.

Considerazioni sulle prestazioni per l'utilizzo dell'apprendimento automatico di Aurora con Aurora Postgre SQL

I servizi Amazon Comprehend e SageMaker AI svolgono la maggior parte del lavoro quando vengono richiamati da una funzione di machine learning di Aurora. Ciò significa che è possibile dimensionare tali risorse in base alle esigenze, in modo indipendente. Per il tuo cluster Aurora Postgre SQL DB, puoi rendere le chiamate di funzione il più efficienti possibile. Di seguito, sono riportate alcune considerazioni sulle prestazioni da tenere in considerazione quando si lavora con Aurora Machine Learning di Aurora Postgre. SQL

Argomenti

- Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora
- Miglioramento dei tempi di risposta con l'elaborazione di query parallela
- Utilizzo di viste materializzate e colonne materializzate

Informazioni sulla modalità batch e sulle funzioni di machine learning di Aurora

In genere, Postgre SQL esegue le funzioni una riga alla volta. Il machine learning di Aurora può ridurre questo sovraccarico combinando in batch le chiamate al servizio di machine learning di Aurora esterno per molte righe con un approccio chiamato esecuzione in modalità batch. In modalità batch, Aurora Machine Learning riceve le risposte per un batch di righe di input e quindi restituisce le risposte alla query in esecuzione una riga alla volta. Questa ottimizzazione migliora la velocità effettiva delle query Aurora senza limitare l'ottimizzatore di query Postgre. SQL

Aurora utilizza automaticamente la modalità batch se si fa riferimento alla funzione dall'elenco `SELECT`, da una clausola `WHERE` o da una clausola `HAVING`. Si noti che le espressioni `CASE` semplici di primo livello sono idonee per l'esecuzione in modalità batch. Anche le espressioni `CASE` ricercate di primo livello sono idonee per l'esecuzione in modalità batch, a condizione che la prima clausola `WHEN` sia un predicato semplice con una chiamata alla funzione in modalità batch.

La funzione definita dall'utente deve essere `LANGUAGE SQL` e deve specificare `PARALLEL SAFE` e `COST 5000`.

Migrazione della funzione dall'istruzione alla clausola SELECT FROM

In genere, una funzione `aws_ml` che è idonea per l'esecuzione in modalità batch viene migrata automaticamente da Aurora alla clausola `FROM`.

La migrazione delle funzioni idonee in modalità batch alla FROM clausola può essere esaminata manualmente a livello di query. A tale scopo, si utilizzano EXPLAIN le istruzioni (and eVERBOSE) ANALYZE e si trovano le informazioni «Elaborazione in batch» sotto ogni modalità batch`Function Scan`. Puoi anche usare EXPLAIN (con`VERBOSE`) senza eseguire la query. Osserva quindi se le chiamate alla funzione vengono visualizzate come un `Function Scan` in un join loop nidificato che non è stato specificato nell'istruzione originale.

Nell'esempio seguente, l'operatore join loop nidificato nel piano mostra che Aurora ha migrato la funzione `anomaly_score`. Ha migrato questa funzione dall'SELECTelenco alla FROM clausola, dove è idonea per l'esecuzione in modalità batch.

```
EXPLAIN (VERBOSE, COSTS false)
SELECT anomaly_score(ts.R.description) from ts.R;
                          QUERY PLAN
---------------------------------------------------------------
 Nested Loop
   Output: anomaly_score((r.description)::text)
   ->  Seq Scan on ts.r
```

```
        Output: r.id, r.description, r.score
  ->  Function Scan on public.anomaly_score
        Output: anomaly_score.anomaly_score
        Function Call: anomaly_score((r.description)::text)
```

Per disabilitare l'esecuzione in modalità batch, imposta il parametro `apg_enable_function_migration` su `false`. Ciò impedisce la migrazione delle `aws_ml` funzioni dalla clausola alla SELECT clausola. `FROM` L'esempio seguente mostra come eseguire questa operazione.

```
SET apg_enable_function_migration = false;
```

Il `apg_enable_function_migration` parametro è un parametro Grand Unified Configuration (GUC) riconosciuto dall'estensione Aurora SQL `apg_plan_mgmt` Postgre per la gestione del piano di query. Per disabilitare la migrazione di funzioni in una sessione, utilizza la gestione del piano di query per salvare il piano risultante come un piano `approved`. In fase di runtime, la gestione del piano di query applica il piano `approved` con la relativa impostazione `apg_enable_function_migration`. Questa applicazione avviene indipendentemente dall'impostazione del parametro. `apg_enable_function_migration` GUC Per ulteriori informazioni, consulta Gestione dei piani di esecuzione delle query per Aurora PostgreSQL.

Utilizzo del parametro max_rows_per_batch

Entrambe le funzioni `aws_comprehend.detect_sentiment` e `aws_sagemaker.invoke_endpoint` hanno un parametro `max_rows_per_batch`. Questo parametro specifica il numero di righe che possono essere inviate al servizio di machine learning di Aurora. Più grande è il set di dati elaborato dalla funzione, maggiore può essere la dimensione del batch.

Le funzioni in modalità batch migliorano l'efficienza creando batch di righe che distribuiscono il costo delle chiamate di funzione Aurora Machine Learning su un numero elevato di righe. Tuttavia, se un'istruzione `SELECT` termina in anticipo a causa di una clausola `LIMIT`, il batch può essere costruito su più righe rispetto a quelle utilizzate dalla query. Questo approccio può comportare costi aggiuntivi sul tuo AWS account. Per ottenere i vantaggi dell'esecuzione in modalità batch, ma evitare di creare batch troppo grandi, utilizza un valore più piccolo per il parametro `max_rows_per_batch` nelle chiamate di funzione.

Se esegui un `EXPLAIN` (`VERBOSE`, `ANALYZE`) di una query che utilizza l'esecuzione in modalità batch, viene visualizzato un operatore `FunctionScan` che si trova sotto un join loop nidificato.

Il numero di loop segnalati da `EXPLAIN` indica il numero di volte che una riga è stata recuperata dall'operatore `FunctionScan`. Se un'istruzione utilizza una LIMIT clausola, il numero di recuperi è costante. Per ottimizzare le dimensioni del batch, imposta il parametro `max_rows_per_batch` su questo valore. Tuttavia, se si fa riferimento alla funzione in modalità batch in un predicato nella clausola `WHERE` o `HAVING`, probabilmente non è possibile conoscere il numero di recuperi in anticipo. In questo caso, utilizza i loop come una linea guida ed esegui esperimenti con `max_rows_per_batch` per trovare un'impostazione che ottimizzi le prestazioni.

Verifica dell'esecuzione in modalità batch

Per verificare se una funzione è stata eseguita in modalità batch, utilizzare `EXPLAIN ANALYZE`. Se è stata utilizzata l'esecuzione in modalità batch, il piano di query includerà le informazioni in una sezione "Elaborazione in batch".

```
EXPLAIN ANALYZE SELECT user-defined-function();
 Batch Processing: num batches=1 avg/min/max batch size=3333.000/3333.000/3333.000
                                 avg/min/max batch call time=146.273/146.273/146.273
```

Questo esempio prevedeva 1 batch che conteneva 3.333 righe, che richiedevano 146,273 ms per l'elaborazione. La sezione "Elaborazione in batch" mostra quanto segue:

- Il numero di batch per questa operazione di scansione funzione
- La dimensione media, minima e massima del batch
- Il tempo di esecuzione medio, minimo e massimo del batch

In genere le dimensioni del batch finale sono inferiore al resto. Questo si traduce spesso in una dimensione batch minima molto inferiore rispetto alla media.

Per restituire le prime righe più rapidamente, imposta il parametro `max_rows_per_batch` su un valore più piccolo.

Per ridurre il numero di chiamate in modalità batch al servizio ML quando utilizzi un `LIMIT` nella funzione definita dall'utente, imposta il parametro `max_rows_per_batch` su un valore inferiore.

Miglioramento dei tempi di risposta con l'elaborazione di query parallela

Per ottenere risultati il più velocemente possibile da un numero elevato di righe, puoi combinare l'elaborazione di query parallela con l'elaborazione in modalità batch. Puoi utilizzare l'elaborazione

di query parallela per istruzioni `SELECT`, `CREATE TABLE AS SELECT` e `CREATE MATERIALIZED VIEW`.

> **Note**
>
> Postgre SQL non supporta ancora le interrogazioni parallele per le istruzioni del linguaggio di manipolazione dei dati ()DML.

L'elaborazione di query parallela avviene sia all'interno del database sia all'interno del servizio ML. Il numero di core nella classe di istanza del database limita il grado di parallelismo che può essere utilizzato durante l'esecuzione di query. Il server di database può costruire un piano di esecuzione di query parallela che partiziona l'attività tra un set di lavoratori paralleli. Quindi ciascuno di questi lavoratori può creare richieste in batch contenenti decine di migliaia di righe (o il numero consentito da ciascun servizio).

Le richieste in batch di tutti i parallel worker vengono inviate all'endpoint SageMaker AI. Il grado di parallelismo supportato dall'endpoint è limitato dal numero e dal tipo di istanze che lo supportano. Per i gradi K di parallelismo, è necessaria una classe di istanza database con almeno i core K. È inoltre necessario configurare l'endpoint SageMaker AI per il modello in modo che abbia K istanze iniziali di una classe di istanze sufficientemente performante.

Per utilizzare l'elaborazione di query parallela, puoi impostare il parametro di archiviazione `parallel_workers` della tabella contenente i dati che intendi passare. Imposta `parallel_workers` su una funzione in modalità batch, ad esempio `aws_comprehend.detect_sentiment`. Se l'ottimizzatore sceglie un piano di query parallelo, i servizi AWS ML possono essere chiamati sia in batch che in parallelo.

Puoi utilizzare i seguenti parametri con la funzione `aws_comprehend.detect_sentiment` per ottenere un piano con parallelismo a quattro vie. Se modifichi uno dei due parametri seguenti, è necessario riavviare l'istanza database per rendere effettive le modifiche.

```
-- SET max_worker_processes to 8;  -- default value is 8
-- SET max_parallel_workers to 8;  -- not greater than max_worker_processes
SET max_parallel_workers_per_gather to 4;  -- not greater than max_parallel_workers

-- You can set the parallel_workers storage parameter on the table that the data
-- for the Aurora machine learning function is coming from in order to manually
 override the degree of
```

```
-- parallelism that would otherwise be chosen by the query optimizer
--
ALTER TABLE yourTable SET (parallel_workers = 4);

-- Example query to exploit both batch-mode execution and parallel query
EXPLAIN (verbose, analyze, buffers, hashes)
SELECT aws_comprehend.detect_sentiment(description, 'en')).*
FROM yourTable
WHERE id < 100;
```

Per ulteriori informazioni sul controllo delle interrogazioni parallele, consulta Piani paralleli nella documentazione di Postgre. SQL

Utilizzo di viste materializzate e colonne materializzate

Quando richiami un AWS servizio come SageMaker AI o Amazon Comprehend dal tuo database, il tuo account viene addebitato in base alla politica tariffaria di quel servizio. Per ridurre al minimo gli addebiti sul tuo account, puoi inserire il risultato della chiamata al AWS servizio in una colonna materializzata in modo che il AWS servizio non venga richiamato più di una volta per riga di input. Se lo desideri, puoi aggiungere una colonna timestamp `materializedAt` per registrare l'ora in cui le colonne sono state materializzate.

La latenza di un'istruzione `INSERT` a riga singola ordinaria è in genere molto inferiore rispetto alla latenza di chiamata di una funzione in modalità batch. Pertanto, potrebbe non essere possibile soddisfare i requisiti di latenza dell'applicazione se si invoca la funzione in modalità batch per ogni singola riga `INSERT` eseguita dall'applicazione. Per concretizzare il risultato della chiamata di un AWS servizio in una colonna materializzata, le applicazioni ad alte prestazioni in genere devono compilare le colonne materializzate. A tale scopo, inviano periodicamente un'istruzione `UPDATE` che opera contemporaneamente su un batch di righe di grandi dimensioni.

`UPDATE` accetta un blocco a livello di riga che può influire su un'applicazione in esecuzione. Quindi potrebbe essere necessario utilizzare `SELECT ... FOR UPDATE SKIP LOCKED` o `MATERIALIZED VIEW`.

Le query analitiche che agiscono su un numero elevato di righe in tempo reale possono combinare la materializzazione in modalità batch con l'elaborazione in tempo reale. A tale scopo, queste query assemblano un `UNION ALL` dei risultati pre-materializzati con una query sulle righe che non dispongono ancora di risultati materializzati. In alcuni casi, tale `UNION ALL` è necessario in più posizioni o la query viene generata da un'applicazione di terze parti. In tal caso, puoi creare un file

per `VIEW` incapsulare l'`UNION ALL`operazione in modo che questo dettaglio non venga esposto al resto dell'applicazione. SQL

Puoi utilizzare una vista materializzata per materializzare i risultati di un'istruzione `SELECT` arbitraria in una snapshot nel tempo. Puoi inoltre utilizzarla per aggiornare la vista materializzata in qualsiasi momento in futuro. Attualmente Postgre SQL non supporta l'aggiornamento incrementale, quindi ogni volta che la vista materializzata viene aggiornata, la vista materializzata viene completamente ricalcolata.

Puoi aggiornare le viste materializzate con l'opzione `CONCURRENTLY`, che aggiorna il contenuto della vista materializzata senza acquisire un blocco esclusivo. Ciò consente a un'SQLapplicazione di leggere dalla vista materializzata mentre viene aggiornata.

Monitoraggio del machine learning di Aurora

È possibile monitorare le funzioni `aws_ml` impostando su `all` il parametro `track_functions` nel gruppo di parametri personalizzato del cluster database. Per impostazione predefinita, questo parametro è impostato su `pl`, il che significa che vengono monitorate solo le funzioni del linguaggio procedurale. Modificando questa impostazione su `all`, vengono monitorate anche le funzioni `aws_ml`. Per ulteriori informazioni, consulta Runtime Statistics nella documentazione di Postgre. SQL

Per informazioni sul monitoraggio delle prestazioni delle operazioni di SageMaker intelligenza artificiale richiamate dalle funzioni di machine learning di Aurora, consulta Monitor Amazon SageMaker AI nella Amazon SageMaker AI Developer Guide.

Con `track_functions` impostato su `all`, puoi eseguire le query nella visualizzazione `pg_stat_user_functions` per ottenere le statistiche sulle funzioni che definisci e utilizzi per richiamare i servizi di machine learning di Aurora. La visualizzazione fornisce il numero di `calls`, `total_time` e `self_time` per ogni funzione.

Per visualizzare le statistiche per le funzioni `aws_sagemaker.invoke_endpoint` e `aws_comprehend.detect_sentiment` puoi filtrare i risultati in base al nome dello schema utilizzando la seguente query.

```
SELECT * FROM pg_stat_user_functions
    WHERE schemaname
    LIKE 'aws_%';
```

Per cancellare le statistiche, procedi come indicato di seguito.

```
SELECT pg_stat_reset();
```

Puoi ottenere i nomi delle SQL funzioni che chiamano la `aws_sagemaker.invoke_endpoint` funzione interrogando il catalogo del sistema SQL `pg_proc` Postgre. Questo catalogo contiene informazioni su funzioni, procedure e altri elementi. Per ulteriori informazioni, consulta pg_proc nella documentazione di Postgre. SQL Di seguito è riportato un esempio di query nella tabella per ottenere i nomi delle funzioni (`proname`) la cui origine (`prosrc`) include il testo invoke_endpoint.

```
SELECT proname FROM pg_proc WHERE prosrc LIKE '%invoke_endpoint%';
```

Esempi di codice per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come usare Aurora con un kit di sviluppo AWS software (SDK).

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Gli scenari sono esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Nozioni di base

Hello Aurora

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Aurora.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
```

```
namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        foreach (var cluster in response.DBClusters)
        {
            Console.WriteLine($"\tCluster: database: {cluster.DatabaseName}
 identifier: {cluster.DBClusterIdentifier}.");
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
```

```
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }
```

```
    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)
```

```
# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
```

```
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
```

```
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters nella AWS guida di riferimento all'API SDK for Rust.

Esempi di codice

- Esempi di base per l'utilizzo di Aurora AWS SDKs
  - Hello Aurora
  - Scopri le basi di Aurora con un SDK AWS
  - Azioni per l'utilizzo di Aurora AWS SDKs
    - Utilizzo CreateDBCluster con un AWS SDK
    - Utilizzo CreateDBClusterParameterGroup con un AWS SDK
    - Utilizzo CreateDBClusterSnapshot con un AWS SDK
    - Utilizzo CreateDBInstance con un AWS SDK
    - Utilizzo DeleteDBCluster con un AWS SDK
    - Utilizzo DeleteDBClusterParameterGroup con un AWS SDK
    - Utilizzo DeleteDBInstance con un AWS SDK

    - Utilizzo DescribeDBClusterParameterGroups con un AWS SDK
    - Utilizzo DescribeDBClusterParameters con un AWS SDK
    - Utilizzo DescribeDBClusterSnapshots con un AWS SDK
    - Utilizzo DescribeDBClusters con un AWS SDK
    - Utilizzo DescribeDBEngineVersions con un AWS SDK
    - Utilizzo DescribeDBInstances con un AWS SDK
    - Utilizzo DescribeOrderableDBInstanceOptions con un AWS SDK o una CLI
    - Utilizzo ModifyDBClusterParameterGroup con un AWS SDK
- Scenari per l'utilizzo di Aurora AWS SDKs
    - Creazione di una REST API per la libreria di prestiti
    - Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Esempi di base per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di Amazon Aurora AWS SDKs con.

Esempi

- Hello Aurora
- Scopri le basi di Aurora con un SDK AWS
- Azioni per l'utilizzo di Aurora AWS SDKs
    - Utilizzo CreateDBCluster con un AWS SDK
    - Utilizzo CreateDBClusterParameterGroup con un AWS SDK
    - Utilizzo CreateDBClusterSnapshot con un AWS SDK
    - Utilizzo CreateDBInstance con un AWS SDK
    - Utilizzo DeleteDBCluster con un AWS SDK
    - Utilizzo DeleteDBClusterParameterGroup con un AWS SDK
    - Utilizzo DeleteDBInstance con un AWS SDK
    - Utilizzo DescribeDBClusterParameterGroups con un AWS SDK
    - Utilizzo DescribeDBClusterParameters con un AWS SDK
    - Utilizzo DescribeDBClusterSnapshots con un AWS SDK

- Utilizzo DescribeDBClusters con un AWS SDK
- Utilizzo DescribeDBEngineVersions con un AWS SDK
- Utilizzo DescribeDBInstances con un AWS SDK
- Utilizzo DescribeOrderableDBInstanceOptions con un AWS SDK o una CLI
- Utilizzo ModifyDBClusterParameterGroup con un AWS SDK

Hello Aurora

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Aurora.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        foreach (var cluster in response.DBClusters)
        {
            Console.WriteLine($"\tCluster: database: {cluster.DatabaseName}
 identifier: {cluster.DBClusterIdentifier}.");
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Codice per il CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
```

```
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
```

```
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
```

```
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import boto3
```

```
# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per Ruby API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
```

```
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters nella AWS guida di riferimento all'API SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scopri le basi di Aurora con un SDK AWS

Gli esempi di codice seguenti mostrano come:

- Crea un gruppo di parametri del cluster di database Aurora personalizzati e imposta i relativi valori.
- Crea un cluster di database che utilizza il gruppo di parametri.
- Crea un'istanza database che contiene un database.
- Acquisisci uno snapshot del cluster di database, quindi elimina le risorse.

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
using Amazon.RDS;
```

```
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
    4.  Get some parameters in the group using the
 DescribeDBClusterParametersAsync method.
    5.  Parse and display some parameters in the group.
    6.  Modify the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBClusterParameterGroupAsync method.
    7.  Get and display the updated parameters using the
 DescribeDBClusterParametersAsync method with a source of "user".
    8.  Get a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Create an Aurora DB cluster that contains a MySql database and uses the
 parameter group.
        using the CreateDBClusterAsync method.
    10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
 method.
    11. Display and select from a list of instance classes available for the
 selected engine and version
        using the paginated DescribeOrderableDBInstanceOptions method.
```

```
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances
method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<AuroraWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<AuroraScenario>();

       auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();
```

```
        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);
```

```
            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)

        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
```

```
        {
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family
by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBClusterParameterGroup>
CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await
auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet
```

```
        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }
```

```
    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
 group.");

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
 "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));
```

```
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
```

```
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
```

```
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;


        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
```

```
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";
```

```
        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {

            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
 auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
 "available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
```

```
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
{cluster.DBClusterIdentifier}.");
        var snapshot = await
auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
```

```
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
```

```
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
 newCluster.DBClusterIdentifier);
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Metodi wrapper che vengono richiamati dallo scenario per gestire le operazioni Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;
```

```
namespace AuroraActions;

/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
```

```
   public async Task<DBClusterParameterGroup>
CreateCustomClusterParameterGroupAsync(
       string parameterGroupFamily,
       string groupName,
       string description)
   {
       var request = new CreateDBClusterParameterGroupRequest
       {
           DBParameterGroupFamily = parameterGroupFamily,
           DBClusterParameterGroupName = groupName,
           Description = description,
       };

       var response = await
_amazonRDS.CreateDBClusterParameterGroupAsync(request);
       return response.DBClusterParameterGroup;
   }

   /// <summary>
   /// Describe the cluster parameters in a parameter group.
   /// </summary>
   /// <param name="groupName">The name of the parameter group.</param>
   /// <param name="source">The optional name of the source filter.</param>
   /// <returns>The collection of parameters.</returns>
   public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
   {
       var paramList = new List<Parameter>();

       DescribeDBClusterParametersResponse response;
       var request = new DescribeDBClusterParametersRequest
       {
           DBClusterParameterGroupName = groupName,
           Source = source,
       };

       // Get the full list if there are multiple pages.
       do
       {
           response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
           paramList.AddRange(response.Parameters);

           request.Marker = response.Marker;
```

```
        }
        while (response.Marker is not null);

        return paramList;
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");
```

```
                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
```

```
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
```

```
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
```

```
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            results.AddRange(response.DBClusters);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
```

```
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
```

```
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
```

```
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per .NET .
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

C++

SDK per C++

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
```

```
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " <<
DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
```

```
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
```

```
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
```

```
   else {
       std::cout << "Let's create a DB cluster." << std::endl;
       const Aws::String administratorName = askQuestion(
               "Enter an administrator username for the database: ");
       const Aws::String administratorPassword = askQuestion(
               "Enter a password for the administrator (at least 8 characters):
");
       Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

       // 8.  Get a list of engine versions for the parameter group family.
       if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                client)) {
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
           return false;
       }

       std::cout << "The available engines for your parameter group family are:"
                 << std::endl;

       int index = 1;
       for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
           std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                     << std::endl;
           ++index;
       }
       int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
       const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

       engineName = engineVersion.GetEngine();
       engineVersionName = engineVersion.GetEngineVersion();
       std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                 << "' and database '" << DB_NAME << "'.\n"
                 << "The DB cluster is configured to use your custom cluster
parameter group '"
                 << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
```

```
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

       Aws::RDS::Model::CreateDBClusterRequest request;
       request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
       request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
       request.SetEngine(engineName);
       request.SetEngineVersion(engineVersionName);
       request.SetMasterUsername(administratorName);
       request.SetMasterUserPassword(administratorPassword);

       Aws::RDS::Model::CreateDBClusterOutcome outcome =
               client.CreateDBCluster(request);

       if (outcome.IsSuccess()) {
           std::cout << "The DB cluster creation has started."
                     << std::endl;
       }
       else {
           std::cerr << "Error with Aurora::CreateDBCluster. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
           return false;
       }
   }

   std::cout << "Waiting for the DB cluster to become available." << std::endl;

   int counter = 0;
   // 11. Wait for the DB cluster to become available.
   do {
       std::this_thread::sleep_for(std::chrono::seconds(1));
       ++counter;
       if (counter > 900) {
           std::cerr << "Wait for cluster to become available timed out ofter "
                     << counter
                     << " seconds." << std::endl;
           cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                            DB_CLUSTER_IDENTIFIER, "", client);
           return false;
       }
```

```
        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
""",
                             client);
            return false;
        }
```

```
        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
```

```
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." <<
std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
```

```
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
```

```
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
```

```
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

```
//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
```

```
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }
```

```
    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
```

```
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
```

```
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
```

```
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
```

```
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per C++ .
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup

- EliminaDBInstance
- Descriva DBCluster ParameterGroups
- DBClusterDescrivi parametri
- Descrivi le DBCluster istantanee
- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

Go

SDK per Go V2

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
```

```
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
```

```
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
```

```
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
```

```
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
```

```
   log.Println("Cluster created and available.")
  }
 }
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
 if err != nil {
  panic(err)
 }
 if dbInstance == nil {
  log.Println("Let's create a database instance in your DB cluster.")
  log.Println("First, choose a DB instance type:")
  instOpts, err := scenario.dbClusters.GetOrderableInstances(
   ctx, *cluster.Engine, *cluster.EngineVersion)
  if err != nil {
   panic(err)
  }
  var instChoices []string
  for _, opt := range instOpts {
   instChoices = append(instChoices, *opt.DBInstanceClass)
  }
  slices.Sort(instChoices)
  instChoices = slices.Compact(instChoices)
  instIndex := scenario.questioner.AskChoice(
   "Which DB instance class do you want to use?\n", instChoices)
```

```
  log.Println("Creating a database instance. This typically takes several
 minutes.")
  dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
   ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
 *cluster.Engine,
   instChoices[instIndex])
  if err != nil {
   panic(err)
  }
  for *dbInstance.DBInstanceStatus != "available" {
   scenario.helper.Pause(30)
   dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
   if err != nil {
    panic(err)
   }
  }
 }
 log.Println("Instance data:")
 log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *dbInstance.Engine)
 log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
```

```
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
```

```
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
```

```
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Definisci le funzioni richiamate dallo scenario per gestire le operazioni Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
```

```
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

```
// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
```

```
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
```

```
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

```
// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
```

```
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK per Go .
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee

- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

Java

SDK per Java 2.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
```

```
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
 group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
 family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
 identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
 +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials\"\n";
```

```
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
```

```
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
```

```
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }
```

```
   private static SecretsManagerClient getSecretClient() {
       Region region = Region.US_WEST_2;
       return SecretsManagerClient.builder()
               .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
               .build();
   }

   private static String getSecretValues(String secretName) {
       SecretsManagerClient secretClient = getSecretClient();
       GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

       GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
       return valueResponse.secretString();
   }

   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
```

```
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
               List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
               for (DBClusterSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
```

```
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
```

```
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();
```

```
            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
```

```
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
```

```
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
```

```
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
```

```
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
 User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()
```

```
    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected
engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)
```

```
    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
```

```
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
```

```
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
```

```
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
```

```
            dbInstanceClass = instanceClassVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
```

```
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }
```

```
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
```

```
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Kotlin.

- CreaDBCluster
- CreaDBClusterParameterGroup
- Crea DBCluster istantanea
- CreaDBInstance
- EliminaDBCluster
- EliminaDBClusterParameterGroup
- EliminaDBInstance
- Descriva DBCluster ParameterGroups
- DBClusterDescrivi parametri
- Descrivi le DBCluster istantanee
- Descriva DBClusters
- Descrivi DBEngine versioni
- Descriva DBInstances
- DescribeOrderableDBInstanceOpzioni
- ModificaDBClusterParameterGroup

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
```

```
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB cluster parameter group that is compatible with a selected
engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
{parameter_group_name}."
        )
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
```

```
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.
```

```
        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
```

```
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
```

```
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
 how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)
```

```
   def create_snapshot(self, cluster_name):
       """
       Shows how to create a DB cluster snapshot and wait until it's available.

       :param cluster_name: The name of a DB cluster to snapshot.
       """
       if q.ask(
           "Do you want to create a snapshot of your DB cluster (y/n)? ",
q.is_yesno
       ):
           snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
           print(
               f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes."
           )
           snapshot = self.aurora_wrapper.create_cluster_snapshot(
               snapshot_id, cluster_name
           )
           while snapshot.get("Status") != "available":
               wait(30)
               snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
           pp(snapshot)
           print("-" * 88)

   def cleanup(self, db_inst, cluster, parameter_group):
       """
       Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
       Before the DB cluster parameter group can be deleted, all associated DB
instances and
       DB clusters must first be deleted.

       :param db_inst: The DB instance to delete.
       :param cluster: The DB cluster to delete.
       :param parameter_group: The DB cluster parameter group to delete.
       """
       cluster_name = cluster["DBClusterIdentifier"]
       parameter_group_name = parameter_group["DBClusterParameterGroupName"]
       if q.ask(
           "\nDo you want to delete the database instance, DB cluster, and
parameter "
           "group (y/n)? ",
           q.is_yesno,
```

```
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definisci le funzioni richiamate dallo scenario per gestire le operazioni Aurora.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
```

```
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
```

```
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
```

```
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

```
   def create_db_cluster(
       self,
       cluster_name,
       parameter_group_name,
       db_name,
       db_engine,
       db_engine_version,
       admin_name,
       admin_password,
   ):
       """
       Creates a DB cluster that is configured to use the specified parameter
group.
       The newly created DB cluster contains a database that uses the specified
engine and
       engine version.

       :param cluster_name: The name of the DB cluster to create.
       :param parameter_group_name: The name of the parameter group to associate
with
                                    the DB cluster.
       :param db_name: The name of the database to create.
       :param db_engine: The database engine of the database that is created,
such as MySql.
       :param db_engine_version: The version of the database engine.
       :param admin_name: The user name of the database administrator.
       :param admin_password: The password of the database administrator.
       :return: The newly created DB cluster.
       """
       try:
           response = self.rds_client.create_db_cluster(
               DatabaseName=db_name,
               DBClusterIdentifier=cluster_name,
               DBClusterParameterGroupName=parameter_group_name,
               Engine=db_engine,
               EngineVersion=db_engine_version,
               MasterUsername=admin_name,
               MasterUserPassword=admin_password,
           )
           cluster = response["DBCluster"]
       except ClientError as err:
           logger.error(
               "Couldn't create database %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
```

```
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
       :param instance_class: The DB instance class for the newly created DB
instance.
       :return: Data about the newly created DB instance.
       """
       try:
           response = self.rds_client.create_db_instance(
               DBInstanceIdentifier=instance_id,
               DBClusterIdentifier=cluster_id,
               Engine=db_engine,
               DBInstanceClass=instance_class,
           )
           db_inst = response["DBInstance"]
       except ClientError as err:
           logger.error(
               "Couldn't create DB instance %s. Here's why: %s: %s",
               instance_id,
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           return db_inst


   def get_engine_versions(self, engine, parameter_group_family=None):
       """
       Gets database engine versions that are available for the specified engine
       and parameter group family.

       :param engine: The database engine to look up.
       :param parameter_group_family: When specified, restricts the returned
list of
                                      engine versions to those that are
compatible with
                                      this parameter group family.
       :return: The list of database engine versions.
       """
       try:
           kwargs = {"Engine": engine}
           if parameter_group_family is not None:
               kwargs["DBParameterGroupFamily"] = parameter_group_family
```

```
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

```
    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Python (Boto3).
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

Rust

SDK per Rust

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Una libreria contenente le funzioni specifiche per lo scenario Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
```

```
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
```

```
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
```

```
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
 ScenarioError> {
        let describe_db_engine_versions =
 self.rds.describe_db_engine_versions(DB_ENGINE).await;
        trace!(versions=?describe_db_engine_versions, "full list of versions");

        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
```

```
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
```

```
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }
```

```
        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
```

```
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
```

```
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
    // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
    // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
    // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
    pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
```

```
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
```

```
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
```

```
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
```

```
   // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
   pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
       let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
       let snapshot = self
           .rds
           .snapshot_cluster(id, format!("{id}_{name}").as_str())
           .await;
       match snapshot {
           Ok(output) => match output.db_cluster_snapshot {
               Some(snapshot) => Ok(snapshot),
               None => Err(ScenarioError::with("Missing Snapshot")),
           },
           Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
       }
   }

   pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
       let mut clean_up_errors: Vec<ScenarioError> = vec![];

       // Delete the instance. rds.DeleteDbInstance.
       let delete_db_instance = self
           .rds
           .delete_db_instance(
               self.db_instance_identifier
                   .as_deref()
                   .expect("instance identifier"),
           )
           .await;
       if let Err(err) = delete_db_instance {
           let identifier = self
               .db_instance_identifier
               .as_deref()
               .unwrap_or("Missing Instance Identifier");
           let message = format!("failed to delete db instance {identifier}");
           clean_up_errors.push(ScenarioError::new(message, &err));
       } else {
           // Wait for the instance to delete
           let waiter = Waiter::default();
           while waiter.sleep().await.is_ok() {
               let describe_db_instances =
self.rds.describe_db_instances().await;
```

```
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
```

```
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
 rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Test per la libreria attraverso automock per il wrapper del client RDS.

```
use crate::rds::MockRdsImpl;
```

```
use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
```

```
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
```

```
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
```

```
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
```

```
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}
```

```
#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
```

```
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()
```

```
 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
```

```
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
```

```
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}
```

```
#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Un file binario per eseguire lo scenario dall'inizio alla fine, utilizzando un inquirer per consentire all'utente di prendere decisioni.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }
```

```
    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
```

```
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
```

```
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
 characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
```

```
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
```

```
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
```

```
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
```

```
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Un wrapper per il servizio Amazon RDS che consente l'automocking per i test.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
```

```
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
```

```
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
```

```
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
```

```
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
```

```
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

Il file Cargo.toml con dipendenze utilizzato in questo scenario.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
```

```
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Per informazioni dettagliate sulle API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Rust.
  - CreaDBCluster
  - CreaDBClusterParameterGroup
  - Crea DBCluster istantanea
  - CreaDBInstance
  - EliminaDBCluster
  - EliminaDBClusterParameterGroup
  - EliminaDBInstance
  - Descriva DBCluster ParameterGroups
  - DBClusterDescrivi parametri
  - Descrivi le DBCluster istantanee
  - Descriva DBClusters
  - Descrivi DBEngine versioni
  - Descriva DBInstances
  - DescribeOrderableDBInstanceOpzioni
  - ModificaDBClusterParameterGroup

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole azioni Aurora con. AWS SDKs Ogni esempio include un collegamento a GitHub, dove è possibile trovare le istruzioni per la configurazione e l'esecuzione del codice.

Questi estratti chiamano l'API Aurora e sono estratti di codice da programmi più grandi che devono essere eseguiti in modo contestuale. È possibile visualizzare le azioni nel contesto in Scenari per l'utilizzo di Aurora AWS SDKs .

Gli esempi seguenti includono solo le operazioni più comunemente utilizzate. Per l'elenco completo, consulta la Documentazione di riferimento delle API di Amazon Aurora.

Esempi

- Utilizzo CreateDBCluster con un AWS SDK
- Utilizzo CreateDBClusterParameterGroup con un AWS SDK
- Utilizzo CreateDBClusterSnapshot con un AWS SDK
- Utilizzo CreateDBInstance con un AWS SDK
- Utilizzo DeleteDBCluster con un AWS SDK
- Utilizzo DeleteDBClusterParameterGroup con un AWS SDK
- Utilizzo DeleteDBInstance con un AWS SDK
- Utilizzo DescribeDBClusterParameterGroups con un AWS SDK
- Utilizzo DescribeDBClusterParameters con un AWS SDK
- Utilizzo DescribeDBClusterSnapshots con un AWS SDK
- Utilizzo DescribeDBClusters con un AWS SDK
- Utilizzo DescribeDBEngineVersions con un AWS SDK
- Utilizzo DescribeDBInstances con un AWS SDK
- Utilizzo DescribeOrderableDBInstanceOptions con un AWS SDK o una CLI
- Utilizzo ModifyDBClusterParameterGroup con un AWS SDK

Utilizzo **`CreateDBCluster`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBCluster`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
/// <param name="dbEngine">The engine to use for the new cluster.</param>
/// <param name="dbEngineVersion">The version of the engine to use.</param>
/// <param name="adminName">The admin username.</param>
/// <param name="adminPassword">The primary admin password.</param>
/// <returns>The cluster object.</returns>
public async Task<DBCluster> CreateDBClusterWithAdminAsync(
    string dbName,
    string clusterIdentifier,
    string parameterGroupName,
    string dbEngine,
    string dbEngineVersion,
    string adminName,
    string adminPassword)
{
    var request = new CreateDBClusterRequest
    {
        DatabaseName = dbName,
        DBClusterIdentifier = clusterIdentifier,
        DBClusterParameterGroupName = parameterGroupName,
        Engine = dbEngine,
```

```
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetEngine(engineName);
    request.SetEngineVersion(engineVersionName);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBClusterOutcome outcome =
            client.CreateDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster creation has started."
                  << std::endl;
```

```
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
```

```
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
```

```
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
```

```
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Per i dettagli sull'API, consulta Create DBCluster in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
```

```
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }
```

```
        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
```

```
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
```

```
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
```

```
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
```

```
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
```

```
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });
```

```
    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta DBClusterCreate in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateDBClusterParameterGroup`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBClusterParameterGroup`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
```

```
    DBParameterGroupFamily:      aws.String(parameterGroupFamily),
    Description:                 aws.String(description),
   })
  if err != nil {
   log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
   return nil, err
  } else {
   return output.DBClusterParameterGroup, err
  }
 }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.
```

```
        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta Create DBCluster ParameterGroup in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
               db_cluster_parameter_group: None,
               ..
           }) => {
               return Err(ScenarioError::with(
                   "CreateDBClusterParameterGroup had empty response",
               ));
           }
           Err(error) => {
               if error.code() == Some("DBParameterGroupAlreadyExists") {
                   info!("Cluster Parameter Group already exists, nothing to
do");
               } else {
                   return Err(ScenarioError::new(
                       "Could not create Cluster Parameter Group",
                       &error,
                   ));
               }
           }
           _ => {
               info!("Created Cluster Parameter Group");
           }
       }

       Ok(())
   }
```

```
    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}
```

```
#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Per i dettagli sull'API, consulta DBClusterParameterGroupCreate in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateDBClusterSnapshot`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBClusterSnapshot`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
```

```
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
```

```
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
```

```
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }
```

```
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
```

```
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
```

```
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
```

```
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
```

```
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
```

```
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
```

```
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Create DBCluster Snapshot in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateDBInstance`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `CreateDBInstance`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
   /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
   /// to determine when the DB instance is ready to use.
   /// </summary>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <param name="dbEngine">The engine for the DB instance.</param>
   /// <param name="dbEngineVersion">Version for the DB instance.</param>
   /// <param name="instanceClass">Class for the DB instance.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> CreateDBInstanceInClusterAsync(
       string dbClusterIdentifier,
       string dbInstanceIdentifier,
       string dbEngine,
       string dbEngineVersion,
       string instanceClass)
   {
       // When creating the instance within a cluster, do not specify the name
or size.
       var response = await _amazonRDS.CreateDBInstanceAsync(
           new CreateDBInstanceRequest()
           {
               DBClusterIdentifier = dbClusterIdentifier,
               DBInstanceIdentifier = dbInstanceIdentifier,
```

```
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetEngine(engineName);
    request.SetDBInstanceClass(dbInstanceClass);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
            client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
 "",
                             client);
            return false;
        }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
```

```
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
 CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
```

```
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
 rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
 response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }
```

```
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
```

```
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Per i dettagli sull'API, consulta Create DBInstance in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));
```

```
            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
```

```
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
```

```
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
```

```
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
```

```
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
```

```
 .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta DBInstanceCreate in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteDBCluster`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DeleteDBCluster`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Delete a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>DB cluster object.</returns>
   public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
   {
       var response = await _amazonRDS.DeleteDBClusterAsync(
           new DeleteDBClusterRequest()
           {
               DBClusterIdentifier = dbClusterIdentifier,
               SkipFinalSnapshot = true
           });
```

```
        return response.DBCluster;
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
```

```
                        << std::endl;
                result = false;
            }
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
```

```
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
```

```
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
```

```
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
```

```
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
```

```
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Delete DBCluster in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteDBClusterParameterGroup`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DeleteDBClusterParameterGroup`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
```

```
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
```

```
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
```

```
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
```

```
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
```

```
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
```

```
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Delete DBCluster ParameterGroup in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteDBInstance`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DeleteDBInstance`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Delete a particular DB instance.
   /// </summary>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
   {
       var response = await _amazonRDS.DeleteDBInstanceAsync(
           new DeleteDBInstanceRequest()
           {
```

```
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
```

```
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
```

```
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
```

```
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
```

```
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
```

```
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
```

```
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
```

```
    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
```

```
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Delete DBInstance in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusterParameterGroups`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusterParameterGroups`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

```
        else:
            return parameter_group
```

- Per i dettagli sull'API, consulta Descrivi DBCluster ParameterGroups in AWS SDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusterParameters`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusterParameters`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
```

```
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
```

```
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset
or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
```

```
                    System.out.println("*** The parameter data type is " +
 para.dataType());
                    System.out.println("*** The parameter description is " +
 para.description());
                    System.out.println("*** The parameter allowed values  is " +
 para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
```

```
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri nell'AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this
 prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
 self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
```

```
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri nella AWS guida di riferimento all'API SDK for Python (Boto3).

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
```

```
            .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
            .await;

        if let Err(err) = parameters_output {
            return Err(ScenarioError::new(
                format!("Failed to retrieve parameters for
 {DB_CLUSTER_PARAMETER_GROUP_NAME}"),
                &err,
            ));
        }

        let parameters = parameters_output
            .unwrap()
            .into_iter()
            .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
            .filter(|p|
 FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
```

```
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Per i dettagli sull'API, consulta DBClusterDescrivi i parametri nell'AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusterSnapshots`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusterSnapshots`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
```

```
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
               List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
               for (DBClusterSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
                   } else {
                       System.out.println(".");
                       Thread.sleep(sleepTime * 5000);
                   }
               }
           }

           System.out.println("The Snapshot is available!");
```

```
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
```

```
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee nell'AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Per i dettagli sull'API, consulta Descrivi le DBCluster istantanee nella guida di riferimento all'API AWS SDK for Python (Boto3).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBClusters`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBClusters`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Returns a list of DB clusters.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
   /// <returns>List of DB clusters.</returns>
   public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
   {
       var results = new List<DBCluster>();

       DescribeDBClustersResponse response;
       DescribeDBClustersRequest request = new DescribeDBClustersRequest
       {
           DBClusterIdentifier = dbClusterIdentifier
       };
       // Get the full list if there are multiple pages.
       do
       {
           response = await _amazonRDS.DescribeDBClustersAsync(request);
           results.AddRange(response.DBClusters);
           request.Marker = response.Marker;
       }
       while (response.Marker is not null);
       return results;
   }
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
```

```
            Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
```

```
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
```

```
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
```

```
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
```

```
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Per i dettagli sull'API, consulta Descrivi DBClusters in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));
```

```
            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
```

```
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
```

```
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
```

```
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
```

```
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
```

```
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Descrivi DBClusters nella AWS guida di riferimento all'API SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBEngineVersions`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBEngineVersions`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
```

```
    }
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
```

```
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
```

```
  return output.DBEngineVersions, nil
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni nell'AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
```

```
                                        engine versions to those that are
compatible with
                                        this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
```

```
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };

       let version_count = describe_db_engine_versions
           .as_ref()
           .map(|o| o.db_engine_versions().len())
           .unwrap_or_default();
       info!(version_count, "got list of versions");

       // Create a map of engine families to their available versions.
       let mut versions = HashMap::<String, Vec<String>>::new();
       describe_db_engine_versions
           .unwrap()
           .db_engine_versions()
           .iter()
           .filter_map(
               |v| match (&v.db_parameter_group_family, &v.engine_version) {
                   (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                   _ => None,
               },
           )
           .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

       Ok(versions)
   }

   pub async fn describe_db_engine_versions(
       &self,
       engine: &str,
   ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
       self.inner
```

```
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
```

```
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Per i dettagli sull'API, consulta Descrivi DBEngine le versioni nell'AWS SDK per il riferimento all'API Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeDBInstances`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `DescribeDBInstances`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
```

```
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
```

```
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    // Waits until the database instance is available.
```

```
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
                       instanceReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }
           System.out.println("Database cluster is available!");

       } catch (RdsException | InterruptedException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK per il riferimento all'API Kotlin.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Per i dettagli sull'API, consulta Descrivi DBInstances in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
```

```
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
```

```
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
```

```
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
```

```
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
```

```
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Cluster
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
```

```
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Per i dettagli sull'API, consulta Descrivi DBInstances nella AWS guida di riferimento all'API SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DescribeOrderableDBInstanceOptions`** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeOrderableDBInstanceOptions`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
```

```
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
```

```
 })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();
```

```
            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOpzioni in AWS SDK for Java 2.x API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: Questo esempio elenca le versioni del motore DB che supportano una classe di istanze DB specifica in un Regione AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Esempio 2: Questo esempio elenca le classi di istanze DB supportate per una versione specifica del motore DB in un Regione AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
```

```
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Per i dettagli sull'API, vedere DescribeOrderableDBInstanceOptions in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Python

SDK per Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.
```

```
        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOptions in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
       let describe_orderable_db_instance_options_items = self
           .rds
           .describe_orderable_db_instance_options(
               DB_ENGINE,
               self.engine_version
                   .as_ref()
                   .expect("engine version for db instance options")
                   .as_str(),
           )
           .await;

       describe_orderable_db_instance_options_items
           .map(|options| {
               options
                   .iter()
                   .filter(|o| o.storage_type() == Some("aurora"))
                   .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                   .collect::<Vec<String>>()
           })
           .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
   }

   pub async fn describe_orderable_db_instance_options(
       &self,
       engine: &str,
       engine_version: &str,
   ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
   {
       self.inner
           .describe_orderable_db_instance_options()
           .engine(engine)
           .engine_version(engine_version)
           .into_paginator()
           .items()
           .send()
           .try_collect()
           .await
   }
```

```
#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");
```

```
    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Per i dettagli sull'API, consulta DescribeOrderableDBInstanceOptions in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ModifyDBClusterParameterGroup`** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare `ModifyDBClusterParameterGroup`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Informazioni di base

.NET

SDK per .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
```

```
                        $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK per .NET API Reference.

C++

SDK per C++

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
 modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK per C++ API Reference.

Go

SDK per Go V2

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
```

```
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
```

```
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK for Python (Boto3) API Reference.

Rust

SDK per Rust

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
                       .parameter_value(format!("{increment}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
               ],
           )
           .await;

       if let Err(error) = modify_db_cluster_parameter_group {
           return Err(ScenarioError::new(
               "Failed to modify cluster parameter group",
               &error,
           ));
```

```
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
```

```
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Per i dettagli sull'API, consulta Modify DBCluster ParameterGroup in AWS SDK for Rust API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di questo servizio con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l'utilizzo di Aurora AWS SDKs

I seguenti esempi di codice mostrano come implementare scenari comuni in Aurora con. AWS SDKs Questi scenari mostrano come eseguire attività specifiche richiamando più funzioni all'interno di Aurora o combinandole con altre. Servizi AWS Ogni scenario include un collegamento al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice.

Gli scenari sono relativi a un livello intermedio di esperienza per aiutarti a comprendere le azioni di servizio nel contesto.

Esempi

- Creazione di una REST API per la libreria di prestiti
- Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

Creazione di una REST API per la libreria di prestiti

L'esempio di codice seguente mostra come creare una libreria di prestiti in cui gli utenti possono prendere in prestito e restituire libri tramite una REST API supportata da un database Amazon Aurora.

Python

SDK per Python (Boto3)

Mostra come utilizzare l' AWS SDK per Python (Boto3) API Amazon Relational Database Service (Amazon RDS) e AWS Chalice per creare un'API REST supportata da un database Amazon Aurora. Il servizio Web è completamente serverless e rappresenta una semplice libreria di prestiti in cui gli utenti possono prendere in prestito e restituire libri. Scopri come:

- Creare e gestire un cluster di database Aurora serverless.
- Utilizzalo per gestire le credenziali AWS Secrets Manager del database.
- Implementare un livello di archiviazione di dati che utilizza Amazon RDS per spostare i dati dentro e fuori dal database.

- Usa AWS Chalice per distribuire un'API REST serverless su Amazon API Gateway e. AWS Lambda
- Utilizza il pacchetto Richieste per inviare le richieste al servizio Web.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un tracciatore di elementi di lavoro di Aurora Serverless

I seguenti esempi di codice mostrano come creare un'applicazione Web che traccia gli elementi di lavoro in database Amazon Aurora Serverless e utilizza il Servizio di email semplice Amazon (Amazon SES) per inviare report.

.NET

SDK per .NET

Mostra come utilizzare per AWS SDK per .NET creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon Aurora e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un RESTful backend.NET.

- Integra un'applicazione web React con AWS i servizi.
- Elenco, aggiunta e aggiornamento di elementi in una tabella Aurora.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

C++

SDK per C++

Mostra come creare un'applicazione Web che traccia gli elementi di lavoro archiviati in un database Amazon Aurora Serverless, con i relativi report.

Per il codice sorgente completo e le istruzioni su come configurare un'API REST C++ che interroga dati Amazon Aurora Serverless e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Java

SDK per Java 2.x

Mostra come creare un'applicazione Web che traccia e segnala gli elementi di lavoro archiviati in un database Amazon RDS.

Per il codice sorgente completo e le istruzioni su come configurare un'API Spring REST che interroga i dati Serverless di Amazon Aurora e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. GitHub

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire un esempio che utilizza l'API JDBC, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

JavaScript

SDK per (v3 JavaScript)

Mostra come utilizzare AWS SDK per JavaScript (v3) per creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon Aurora e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend Express Node.js.

- Integra un'applicazione web React.js con. Servizi AWS
- Elenca, aggiungi e aggiorna elementi in una tabella Aurora.
- Invia un report per e-mail degli elementi di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Kotlin

SDK per Kotlin

Mostra come creare un'applicazione Web che traccia e segnala gli elementi di lavoro archiviati in un database Amazon RDS.

Per il codice sorgente completo e le istruzioni su come configurare un'API Spring REST che interroga i dati Serverless di Amazon Aurora e per l'utilizzo da parte di un'applicazione React, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

PHP

SDK per PHP

Mostra come utilizzare per AWS SDK per PHP creare un'applicazione Web che tenga traccia degli elementi di lavoro in un database Amazon RDS e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza un front-end creato con React.js per interagire con un backend RESTful PHP.

- Integra un'applicazione web React.js con AWS i servizi.
- Elenca, aggiungi, aggiorna ed elimina gli elementi in una tabella Amazon RDS.
- Invia un report per e-mail degli articoli di lavoro filtrati tramite Amazon SES.
- Distribuisci e gestisci risorse di esempio con lo AWS CloudFormation script incluso.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS

- Amazon SES

Python

SDK per Python (Boto3)

Mostra come utilizzare per AWS SDK per Python (Boto3) creare un servizio REST che tenga traccia degli elementi di lavoro in un database Amazon Aurora Serverless e invii report tramite e-mail utilizzando Amazon Simple Email Service (Amazon SES). Questo esempio utilizza il framework Web Flask per gestire il routing HTTP e si integra con una pagina Web React per presentare un'applicazione Web completamente funzionale.

- Crea un servizio Flask REST che si integri con. Servizi AWS
- Lettura, scrittura e aggiornamento degli elementi di lavoro archiviati in un database Aurora Serverless.
- Crea un AWS Secrets Manager segreto che contenga le credenziali del database e usalo per autenticare le chiamate al database.
- Utilizzo di Amazon SES per inviare report via e-mail sugli elementi di lavoro.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Aurora
- Amazon RDS
- Servizi di dati di Amazon RDS
- Amazon SES

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di questo servizio con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Best practice con Amazon Aurora

Questo argomento include informazioni sulle best practice e sulle opzioni generali per l'utilizzo o la migrazione di dati a un cluster database Amazon Aurora.

Alcune delle best practice per Amazon Aurora sono specifiche di un motore di database particolare. Per ulteriori informazioni sulle best practice Aurora specifiche per un motore di database, consulta quanto segue:

Motore di database	Best practice
Amazon Aurora MySQL	Per informazioni, consulta Le migliori pratiche con Amazon Aurora My SQL
Amazon Aurora PostgreSQL	Per informazioni, consulta Best practice con Amazon Aurora PostgreSQL

> **Note**
>
> Per suggerimenti comuni per Aurora, consulta .

Argomenti

- Linee guida operative di base per Amazon Aurora
- Suggerimenti relativi alla RAM per un'istanza di database
- AWS driver di database
- Monitoraggio di Amazon Aurora
- Uso di gruppi di parametri database e gruppi di parametri cluster database
- Video sulle best practice Amazon Aurora

Linee guida operative di base per Amazon Aurora

Di seguito sono illustrate le linee guida operative di base e le best practice che tutti dovrebbero seguire durante l'utilizzo di Amazon Aurora. Il contratto sul livello di servizio di Amazon RDS richiede che tu segua queste linee guida.

- Monitora l'utilizzo della memoria, della CPU e dello storage. Puoi configurare Amazon in modo che CloudWatch ti avvisi quando i modelli di utilizzo cambiano o quando ti avvicini alla capacità della tua implementazione. In questo modo, è possibile mantenere le prestazioni e la disponibilità del sistema.
- Se l'applicazione client memorizza nella cache i dati DNS (Domain Name Service) delle istanze DB, imposta un valore time-to-live (TTL) inferiore a 30 secondi. L'indirizzo IP sottostante di un'istanza DB può cambiare dopo un failover. Pertanto, la memorizzazione nella cache dei dati DNS per un periodo di tempo prolungato può causare errori di connessione se l'applicazione tenta di connettersi a un indirizzo IP che non è più in servizio. Nei cluster di database Aurora con più repliche di lettura si possono verificare errori di connessione quando le connessioni utilizzano l'endpoint di lettura e una delle istanze di replica di lettura è in modalità di manutenzione o è stata eliminata.
- Prova il failover per il cluster per capire quanto tempo impiega il processo per il tuo caso d'uso. Il test di failover consente di garantire che l'applicazione che accede al cluster DB possa connettersi automaticamente al nuovo cluster DB dopo il failover.

Suggerimenti relativi alla RAM per un'istanza di database

Per ottimizzare le prestazioni, alloca RAM sufficiente in modo che il working set risieda quasi interamente nella memoria. Per determinare se il tuo set di lavoro è quasi tutto in memoria, esamina le seguenti metriche su Amazon CloudWatch:

- `VolumeReadIOPS` – Misura il numero medio di operazioni I/O di lettura da un volume di cluster, indicato a intervalli di 5 minuti. Il valore di `VolumeReadIOPS` dovrebbe essere di piccola entità e stabile. In alcuni casi, potresti scoprire che l'I/O di lettura è in aumento o è più alto del solito. In tal caso, esaminare le istanze DB nel cluster DB per vedere quali istanze DB stanno causando l'aumento dell'I/O.

> **Tip**
>
> Se il tuo cluster Aurora MySQL utilizza una query parallela, potresti vedere un aumento dei valori di `VolumeReadIOPS`. Le query parallele non utilizzano il pool di buffer. Pertanto, sebbene le query siano veloci, questa elaborazione ottimizzata può comportare un aumento delle operazioni di lettura e degli addebiti associati.

- `BufferCacheHitRatio` – Questo parametro misura la percentuale di richieste gestite dalla cache del buffer di un'istanza database nel cluster di database. Questo parametro offre visibilità sulla quantità di dati serviti dalla memoria.

  Una percentuale di riscontri elevata indica che l'istanza database dispone di memoria sufficiente. Una percentuale di riscontri bassa indica che le query su questa istanza database vengono spesso trasferite su disco. In questo caso esamina il carico di lavoro per stabilire quali query causano questo comportamento.

Se dall'esame del carico di lavoro emerge la necessità di aumentare la memoria, il dimensionamento della classe di istanza database con passaggio a una classe con più RAM potrebbe risultare vantaggioso. In seguito, potrai esaminare i parametri riportati sopra e incrementare ulteriormente in base alle esigenze. Se il cluster Aurora è più grande di 40 TB, non utilizzare le classi di istanza db.t2, db.t3 or db.t4g.

Per ulteriori informazioni, consulta CloudWatch Parametri Amazon per Amazon Aurora.

AWS driver di database

Consigliamo la AWS suite di driver per la connettività delle applicazioni. I driver sono stati progettati per fornire supporto per tempi di switchover e failover più rapidi e per l'autenticazione con AWS Secrets Manager, AWS Identity and Access Management (IAM) e Federated Identity. I AWS driver si basano sul monitoraggio dello stato del cluster DB e sulla conoscenza della topologia del cluster per determinare il nuovo writer. Questo approccio riduce i tempi di switchover e failover a secondi a una cifra, rispetto alle decine di secondi dei driver open source.

Con l'introduzione di nuove funzionalità di servizio, l'obiettivo della AWS suite di driver è disporre di un supporto integrato per queste funzionalità di servizio.

Per ulteriori informazioni, consulta Connessione ai cluster Aurora DB con i driver AWS.

Monitoraggio di Amazon Aurora

Amazon Aurora fornisce varie metriche e approfondimenti che è possibile monitorare per determinare lo stato e le prestazioni del cluster database Aurora. Puoi utilizzare vari strumenti, come, e l' CloudWatch API AWS Management Console AWS CLI, per visualizzare le metriche di Aurora. Puoi visualizzare la combinazione di Performance Insights e CloudWatch metriche nella dashboard di Performance Insights e monitorare la tua istanza DB. Per utilizzare questa visualizzazione di

monitoraggio, Performance Insights deve essere attivato per l'istanza database specifica. Per ulteriori informazioni su questa visualizzazione di monitoraggio, consulta Visualizzazione delle metriche combinate con la dashboard Performance Insights.

È possibile creare un report di analisi delle prestazioni per un periodo di tempo specifico e visualizzare le informazioni dettagliate identificate e i suggerimenti per risolvere i problemi. Per ulteriori informazioni, consulta Creazione di un rapporto di analisi delle prestazioni in Performance Insights.

Uso di gruppi di parametri database e gruppi di parametri cluster database

È consigliabile provare le modifiche al gruppo di parametri database e al gruppo di parametri del cluster di database su un cluster di database di test prima di applicarle al cluster di database di produzione. Un'impostazione non corretta dei parametri del motore di database può avere ripercussioni negative non previste, tra cui prestazioni scadenti e instabilità del sistema.

Fai sempre attenzione quando modifichi i parametri del motore di database ed effettui il backup di un cluster di database prima di modificare un gruppo di parametri database. Per ulteriori informazioni sul backup di un cluster di database, consulta Backup e ripristino di un cluster DB Amazon Aurora.

Video sulle best practice Amazon Aurora

Il canale AWS Online Tech Talks YouTube include una presentazione video sulle migliori pratiche per creare e configurare un cluster Amazon Aurora DB per renderlo più sicuro e ad alta disponibilità. Visualizzare il video sulle best practice per l'elevata disponibilità di Amazon Aurora.

Esecuzione di un proof of concept con Amazon Aurora

Di seguito, troverai una spiegazione di come impostare ed eseguire un proof of concept per Aurora. Un proof of concept è una verifica che esegui per capire se Aurora è la scelta giusta per la tua applicazione. Il proof of concept permette di comprendere le funzionalità di Aurora nel contesto delle tue applicazioni di database e di confrontare Aurora con il tuo attuale ambiente di database. Può anche mostrare il livello di impegno necessario per spostare dati, SQL codice porta, ottimizzare le prestazioni e adattare le procedure di gestione correnti.

In questo argomento, è possibile trovare una panoramica e una step-by-step descrizione delle procedure e delle decisioni di alto livello necessarie per l'esecuzione di un proof of concept, elencate di seguito. Per istruzioni dettagliate, puoi utilizzare i link alla documentazione completa per argomenti specifici.

Panoramica di un proof of concept di Aurora

Quando esegui un proof of concept per Amazon Aurora, impari cosa serve per trasferire i dati e SQL le applicazioni esistenti su Aurora. Puoi mettere in pratica gli aspetti importanti di Aurora su scala, utilizzando un volume di dati e attività rappresentativo del tuo ambiente di produzione. L'obiettivo è accertarti che i punti di forza di Aurora rispondano efficacemente alle sfide che ti hanno portato ad abbandonare la tua precedente infrastruttura di database. Al termine di un proof of concept, disporrai di un solido piano per il benchmark delle prestazioni su più vasta scala e il test dell'applicazione. A quel punto, sarai in grado di comprendere i maggiori elementi di lavoro per una distribuzione in produzione.

I seguenti consigli sulle best practice ti aiutano a evitare errori comuni che causano problemi durante il benchmark. Tuttavia, questo argomento non copre il step-by-step processo di esecuzione dei benchmark e di ottimizzazione delle prestazioni. Tali procedure variano a seconda del carico di lavoro e delle funzionalità di Aurora che utilizzi. Per informazioni dettagliate, consulta la documentazione relativa alle prestazioni, ad esempio Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora, Miglioramenti alle prestazioni di Amazon Aurora MySQL, Prestazioni e scalabilità per Amazon Aurora PostgreSQL e Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Le informazioni contenute in questo argomento si applicano principalmente alle applicazioni in cui l'organizzazione scrive il codice e progetta lo schema e che supportano i motori di database open source My SQL e SQL Postgre. Se stai testando un'applicazione commerciale o un codice generato dal framework di un'applicazione, potresti non avere la flessibilità necessaria per applicare tutte

le linee guida. In questi casi, rivolgiti al tuo AWS rappresentante per verificare se esistono le best practice o i case study di Aurora per il tuo tipo di applicazione.

1. Individuare gli obiettivi

Quando valuti Aurora nell'ambito di un proof of concept, puoi scegliere quali misurazioni eseguire e come valutare il successo della prova.

Devi accertarti che tutte le funzionalità dell'applicazione siano compatibili con Aurora. Poiché le versioni principali di Aurora sono compatibili via cavo con le corrispondenti versioni principali di My SQL e PostgreSQL, la maggior parte delle applicazioni sviluppate per tali motori sono compatibili anche con Aurora. Tuttavia, è sempre necessario convalidare la compatibilità per ciascuna applicazione.

Ad esempio, alcune scelte relative alla configurazione fatte durante l'impostazione di un cluster Aurora influenzano la possibilità o la necessità di utilizzare specifiche funzionalità del database. Potresti iniziare con il tipo di cluster Aurora destinato a scopi più generici, detto assegnato. Successivamente, puoi stabilire se una configurazione specifica, ad esempio serverless o con query in parallelo, offre vantaggi per il tuo carico di lavoro.

Le domande seguenti aiutano a individuare e quantificare gli obiettivi:

- Aurora supporta tutti i casi d'uso funzionali del tuo carico di lavoro?
- Qual è la dimensione del set di dati o il livello di carico che desideri? Sei in grado di raggiungere tale livello?
- Quali sono i requisiti specifici in termini di throughput delle query o latenza? Sei in grado di soddisfarli?
- Qual è la quantità minima accettabile di tempo di inattività pianificato e non pianificato per il carico di lavoro? Sei in grado di raggiungerla?
- Quali sono i parametri necessari per l'efficienza operativa? Sei in grado di monitorarli in modo accurato?
- Aurora supporta obiettivi di business specifici, come la riduzione dei costi e l'aumento della velocità di distribuzione o provisioning? Hai un modo per quantificare questi obiettivi?
- Puoi soddisfare tutti i requisiti di sicurezza e conformità per il carico di lavoro?

Ti invitiamo ad acquisire conoscenza sui motori di database e sulle funzionalità della piattaforma di Aurora e a consultare la documentazione sui servizi. Prendi nota di tutte le funzionalità che possono

aiutarti a raggiungere i risultati desiderati. Uno di questi potrebbe essere il consolidamento del carico di lavoro, descritto nel post del blog AWS Database Come pianificare e ottimizzare Amazon Aurora with SQL My compatibility for consolidated workload. Un'altra potrebbe essere la scalabilità in base alla domanda, descritta in Auto Scaling di Amazon Aurora con repliche Aurora nella Guida per l'utente di Amazon Aurora. Altre ancora potrebbero essere l'aumento delle prestazioni o la semplificazione delle operazioni di database.

2. Comprendere le caratteristiche del carico di lavoro

Valuta Aurora nel contesto del caso d'uso previsto. Aurora è un'ottima scelta per i carichi di lavoro di elaborazione delle transazioni online (OLTP). È inoltre possibile eseguire report sul cluster che contiene i OLTP dati in tempo reale senza dover fornire un cluster di data warehouse separato. Per capire se il tuo caso d'uso rientra in queste categorie, verifica le caratteristiche seguenti:

- Elevata simultaneità, con decine, centinaia o migliaia di client simultanei.
- Ampio volume di query a bassa latenza (da millisecondi a secondi).
- Transazioni brevi e in tempo reale.
- Modelli di query altamente selettivi, con ricerche basate su indici.
- PerHTAP, query analitiche che possono sfruttare la query parallela Aurora.

Uno dei principali fattori che influenzano le scelte relative al database è la velocità dei dati. Velocità elevata significa che i dati vengono inseriti e aggiornati molto frequentemente. Un sistema di questo tipo potrebbe avere migliaia di connessioni e centinaia di migliaia di query simultanee con lettura e scrittura da e verso un database. Di solito, le query in sistemi a velocità elevata interessano un numero relativamente ridotto di righe e in genere accedono a più colonne nella stessa riga.

Aurora è concepito per gestire dati a velocità elevata. A seconda del carico di lavoro, un cluster Aurora con un'unica istanza di database r4.16xlarge può elaborare più di 600.000 istruzioni `SELECT` al secondo. Anche in questo caso, a seconda del carico di lavoro, tale cluster può elaborare 200.000 `INSERT`, `UPDATE` e `DELETE` istruzioni al secondo Aurora è un database di archiviazione a righe ed è ideale per carichi di lavoro ad alto volume, throughput elevato e altamente parallelizzati. OLTP

Aurora può anche eseguire query di reporting sullo stesso cluster che gestisce il carico di lavoro. OLTP Aurora supporta fino a 15 repliche, ciascuna delle quali è in media a 10-20 millisecondi dall'istanza primaria. Gli analisti possono interrogare OLTP i dati in tempo reale senza copiarli in un cluster di data warehouse separato. Con i cluster Aurora che utilizzano la funzionalità di query in

parallelo, puoi effettuare l'offload di gran parte del lavoro di elaborazione, filtraggio e aggregazione al sottosistema di storage Aurora a elevata distribuzione.

Utilizza questa fase di pianificazione per acquisire familiarità con le funzionalità di Aurora, altri AWS servizi, e. AWS Management Console AWS CLI Inoltre, verifica il modo in cui queste funzionalità interagiscono con gli strumenti che prevedi di utilizzare nel proof of concept.

3. Esercitati con o AWS Management ConsoleAWS CLI

Come passaggio successivo, esercitati con il AWS Management Console o il AWS CLI, per acquisire familiarità con questi strumenti e con Aurora.

Esercitati con AWS Management Console

Le seguenti attività iniziali con i cluster di database Aurora servono principalmente a familiarizzare con l' AWS Management Console ambiente ed esercitarsi a configurare e modificare i cluster Aurora. Se utilizzi i motori di SQL database compatibili con My e Postgre SQL con AmazonRDS, puoi sfruttare queste conoscenze quando usi Aurora.

Sfruttando il modello di storage condiviso di Aurora e funzionalità come la replica e le snapshot, puoi trattare interi cluster database come un altro tipo di oggetto che gestisci liberamente. Durante il proof of concept, puoi frequentemente impostare, eliminare e modificare la capacità dei cluster Aurora. Non sei vincolato alle scelte iniziali relative a capacità, impostazioni del database e layout fisico dei dati.

Per iniziare, imposta un cluster Aurora vuoto. Scegli il tipo di capacità assegnato e l'ubicazione regionale per gli esperimenti iniziali.

Connect a quel cluster utilizzando un programma client, ad esempio un'applicazione da SQL riga di comando. Inizialmente, ti colleghi tramite l'endpoint del cluster. Ti connetti a quell'endpoint per eseguire qualsiasi operazione di scrittura, come le istruzioni Data Definition Language (DDL) e i processi di estrazione, trasformazione, caricamento (ETL). Nelle fasi successive del proof of concept, colleghi sessioni con un numero elevato di query utilizzando l'endpoint di lettura, che distribuisce il carico di lavoro di query su più istanze database nel cluster.

Dimensiona il cluster aggiungendo altre repliche Aurora. Per queste procedure, consulta Replica con Amazon Aurora. Aumenta o riduci le istanze DB modificando la classe dell' AWS istanza. Osserva come Aurora semplifica questi tipi di operazioni in modo tale che, se le stime iniziali della capacità di sistema non sono accurate, puoi regolarle in un secondo momento senza dover ricominciare da capo.

Crea una snapshot e ripristinala in un cluster diverso.

Esamina i parametri del cluster per visualizzare l'attività nel tempo e il modo in cui i parametri si applicano alle istanze database nel cluster.

È utile acquisire familiarità con queste operazioni sin dall' AWS Management Console inizio. Dopo aver capito cosa è possibile fare con Aurora, puoi procedere con l'automazione di tali operazioni tramite la AWS CLI. Nelle sezioni seguenti, puoi trovare maggiori dettagli sulle procedure e le migliori pratiche per queste attività durante il proof-of-concept periodo.

Esercitati con AWS CLI

Consigliamo di automatizzare le procedure di implementazione e gestione, anche in un proof-of-concept ambiente. A tal fine, acquisisci familiarità con il AWS CLI se non lo sei già. Se utilizzi i motori di SQL database compatibili con My e Postgre SQL con AmazonRDS, puoi sfruttare queste conoscenze quando usi Aurora.

Aurora include generalmente gruppi di istanze database disposte in cluster. Pertanto, molte operazioni prevedono di stabilire quali istanze database sono associate a un cluster ed eseguire poi le operazioni amministrative in un loop per tutte le istanze.

Ad esempio, potresti automatizzare passaggi come la creazione di cluster Aurora e il successivo dimensionamento con classi di istanze più grandi o il dimensionamento orizzontale con ulteriori istanze database. Questo aiuta a ripetere qualsiasi fase nel proof of concept e a esplorare scenari ipotetici con diversi tipi di configurazioni di cluster Aurora.

Scopri le funzionalità e le limitazioni degli strumenti di distribuzione dell'infrastruttura come AWS CloudFormation. Potresti scoprire che le attività che svolgi in un proof-of-concept contesto non sono adatte all'uso in produzione. Ad esempio, il AWS CloudFormation comportamento di modifica consiste nel creare una nuova istanza ed eliminare quella corrente, inclusi i relativi dati. Per maggiori dettagli su questo comportamento, consulta Aggiornamento dei comportamenti delle risorse stack nella Guida per l'utente di AWS CloudFormation .

4. Creare il cluster Aurora

Con Aurora, puoi esplorare scenari ipotetici aggiungendo istanze database al cluster e dimensionando le istanze database a classi di istanze più potenti. Puoi anche creare cluster con diverse impostazioni di configurazione per eseguire lo stesso carico di lavoro contemporaneamente. Con Aurora, hai molta flessibilità per impostare, eliminare e riconfigurare i cluster database. Detto

questo, è utile mettere in pratica queste tecniche nelle prime fasi del proof-of-concept processo. Per le procedure generali per la creazione di cluster Aurora, consulta Creazione di un cluster database Amazon Aurora.

Se utile, inizia con un cluster con le seguenti impostazioni. Salta questo passaggio solo se prevedi casi d'uso specifici. Ad esempio, potresti saltare questo passaggio se il caso d'uso richiede un tipo speciale di cluster Aurora oppure se necessiti di una particolare combinazione di versione e motore di database.

- Disattiva Creazione rapida. Per il proof of concept, ti consigliamo di conoscere tutte le impostazioni scelte in modo da poter creare cluster identici o leggermente diversi in un secondo momento.
- Usa una versione recente del motore DB. Queste combinazioni di motore di database e versione hanno un'ampia compatibilità con altre funzionalità di Aurora e un notevole utilizzo da parte dei clienti per le applicazioni di produzione.
  - Aurora My SQL versione 3.x (compatibilità con My SQL 8.0)
  - Aurora Postgre SQL versione 15.x o 16.x
- Scegli il modello Dev/Test (Sviluppo/Test). Questa scelta non è importante per le tue attività. proof-of-concept
- Per DB instance class (Classe dell'istanza database), scegli Memory optimized classes (Classi ottimizzate per la memoria) e una delle classi di istanze xlarge. Puoi modificare la classe di istanza in un secondo momento.
- Sotto Multi-AZ Deployment (Implementazione Multi-AZ), scegli Create an Aurora Replica or Reader node in a different AZ (Crea una replica Aurora o nodo di lettura in un AZ differente). Molti degli aspetti più utili di Aurora riguardano i cluster e le diverse istanze database. È utile iniziare sempre con almeno due istanze database in qualsiasi nuovo cluster. L'utilizzo di una diversa zona di disponibilità per la seconda istanza database aiuta a testare diversi scenari di disponibilità elevata.
- Quando scegli i nomi per le istanze database, utilizza una convenzione di denominazione generica. Non fate riferimento a nessuna istanza DB del cluster come «writer», perché istanze DB diverse assumono quei ruoli in base alle esigenze. Consigliamo di utilizzare una denominazione di tipo `clustername-az-serialnumber`, ad esempio `myprodappdb-a-01`. Tali denominazioni identificano l'istanza database e il suo posizionamento.
- Imposta una conservazione elevata del backup per il cluster Aurora. Con un periodo di conservazione lungo, puoi eseguire point-in-time recovery (PITR) per un periodo fino a 35 giorni. È possibile ripristinare lo stato noto del database dopo aver eseguito test che coinvolgono DDL le istruzioni Data Manipulation Language (DML). Inoltre, puoi eseguire il ripristino in caso di eliminazione o modifica involontaria dei dati.

- Abilita ulteriori funzionalità di ripristino, registrazione e monitoraggio delle funzioni al momento della creazione del cluster. Attiva tutte le opzioni disponibili in Backtrack, Performance Insights, Monitoring e Log exports. L'attivazione di queste funzionalità consente di testare l'idoneità di funzionalità come backtracking, Enhanced Monitoring (Monitoraggio avanzato) o Performance Insights per il carico di lavoro in uso. Inoltre, puoi eseguire facilmente l'analisi delle prestazioni e la risoluzione dei problemi durante il proof of concept.

5. Configurare lo schema

Nel cluster Aurora, configura i database, le tabelle, gli indici, le chiavi esterne e gli altri oggetti dello schema per l'applicazione. Se stai passando da un altro sistema di database SQL SQL compatibile con My o Postgre, aspettati che questa fase sia semplice e diretta. Utilizzate la stessa SQL sintassi e la stessa riga di comando o altre applicazioni client che conoscete per il vostro motore di database.

Per emettere SQL istruzioni sul tuo cluster, individua l'endpoint del cluster e fornisci quel valore come parametro di connessione all'applicazione client. Puoi trovare l'endpoint del cluster nella scheda Connectivity (Connettività) della pagina dei dettagli del cluster. L'endpoint del cluster è quello con l'etichetta Writer (di scrittura). L'altro endpoint, con etichetta Reader (di lettura), rappresenta una connessione di sola lettura che puoi fornire agli utenti finali che eseguono report o altre query di sola lettura. Per eventuali problemi di connessione al cluster, consulta Connessione a un cluster database Amazon Aurora.

Se trasferisci lo schema e i dati da un sistema di database diverso, probabilmente a questo punto dovrai apportare modifiche allo schema. Queste modifiche allo schema devono corrispondere alla SQL sintassi e alle funzionalità disponibili in Aurora. A questo punto, potresti tralasciare alcuni vincoli, colonne e trigger o altri oggetti dello schema. Questo può essere utile soprattutto se tali oggetti richiedono la rielaborazione per la compatibilità Aurora e non sono importanti per gli obiettivi del proof of concept.

Se stai migrando da un sistema di database con un motore sottostante diverso da quello di Aurora, prendi in considerazione l'utilizzo di AWS Schema Conversion Tool (AWS SCT) per semplificare il processo. Per informazioni dettagliate, consulta la Guida per l'utente di AWS Schema Conversion Tool. Per dettagli generali sulle attività di migrazione e porting, consulta il white paper Migrating Your Databases to Amazon AWS Aurora.

Durante questa fase, puoi valutare se vi sono inefficienze nell'impostazione dello schema, ad esempio nella strategia di indicizzazione o altre strutture a tabella come le tabelle partizionate. Tali inefficienze possono essere amplificate quando implementi l'applicazione su un cluster con

più istanze database e un carico di lavoro pesante. Valuta se è possibile ottimizzare questi aspetti prestazionali ora o durante attività successive come un test di benchmark completo.

6. Importare i dati

Durante il proof of concept, trasferisci i dati o un campione rappresentativo degli stessi dal precedente sistema di database. Se utile, imposta almeno alcuni dati in ciascuna delle tue tabelle. Ciò aiuta a testare la compatibilità di tutti i tipi di dati e le funzionalità dello schema. Dopo aver fatto pratica con le funzionalità base di Aurora, aumenta la quantità dei dati. Una volta completata la dimostrazione concettuale, dovresti testare ETL gli strumenti, le query e il carico di lavoro complessivo con un set di dati sufficientemente grande da trarre conclusioni accurate.

Puoi utilizzare diverse tecniche per importare dati di backup fisici o logici in Aurora. Per i dettagli, consulta Migrazione dei dati verso un cluster Amazon Aurora My DB SQL o Migrazione dei dati su Amazon Aurora con compatibilità Postgre SQL a seconda del motore di database che utilizzi nel proof of concept.

Sperimenta con gli ETL strumenti e le tecnologie che stai considerando. Individua quelli che soddisfano meglio le tue esigenze. Prendi in considerazione sia il throughput che la flessibilità. Ad esempio, alcuni ETL strumenti eseguono un trasferimento una tantum e altri prevedono la replica continua dal vecchio sistema ad Aurora.

Se stai migrando da un sistema SQL compatibile con My ad Aurora MySQL, puoi utilizzare gli strumenti di trasferimento dati nativi. Lo stesso vale se stai migrando da un SQL sistema compatibile con Postgre ad Aurora Postgre. SQL Se stai migrando da un sistema di database che utilizza un motore sottostante diverso da Aurora, puoi sperimentare con AWS Database Migration Service ().AWS DMS Per ulteriori informazioni AWS DMS, consulta la Guida per l'AWS Database Migration Service utente.

Per i dettagli sulle attività di migrazione e porting, consulta il AWS white paper Aurora migration handbook.

7. Trasferisci il tuo codice SQL

Provare SQL le applicazioni associate richiede diversi livelli di impegno a seconda dei diversi casi. In particolare, il livello di impegno dipende dal fatto che si passi da un sistema SQL SQL compatibile con My o Postgre o da un altro tipo.

- Se stai passando da RDS for My SQL o RDS PostgreSQL, le SQL modifiche sono abbastanza piccole da poter provare il SQL codice originale con Aurora e incorporare manualmente le modifiche necessarie.
- Allo stesso modo, se passi da un database locale compatibile con My SQL o PostgreSQL, puoi provare il codice originale SQL e incorporare manualmente le modifiche.
- Se provieni da un database commerciale diverso, le SQL modifiche richieste sono più ampie. In questo caso, prendi in considerazione l'utilizzo di AWS SCT.

Durante questa fase, puoi valutare se vi sono inefficienze nell'impostazione dello schema, ad esempio nella strategia di indicizzazione o altre strutture a tabella come le tabelle partizionate. Valuta se è possibile ottimizzare questi aspetti prestazionali ora o durante attività successive come un test di benchmark completo.

Puoi verificare la logica di connessione del database nella tua applicazione. Per sfruttare l'elaborazione distribuita di Aurora, potresti dover utilizzare connessioni separate per le operazioni di lettura e scrittura e utilizzare sessioni relativamente brevi per le operazioni di query. Per informazioni sulle connessioni, consulta 9. Connettere ad Aurora.

Prendi in considerazione l'eventualità di dover scendere a compromessi o fare rinunce per risolvere problemi nel database di produzione. Dedica tempo alla proof-of-concept pianificazione per apportare miglioramenti alla progettazione dello schema e alle query. Per capire se puoi facilmente ottenere vantaggi in termini di prestazioni, costi operativi e scalabilità, prova l'applicazione originale e quella modificata contemporaneamente su cluster Aurora diversi.

Per i dettagli sulle attività di migrazione e porting, consulta il AWS white paper Aurora migration handbook.

8. Specificare le impostazioni di configurazione

È inoltre possibile esaminare i parametri di configurazione del database come parte dell'esercizio Aurora proof-of-concept . È possibile che le impostazioni di SQL configurazione di My SQL o Postgre siano già state ottimizzate per garantire prestazioni e scalabilità nell'ambiente corrente. Il sottosistema di storage Aurora è adattato e ottimizzato per un ambiente basato sul cloud e distribuito con un sottosistema di storage ad alta velocità. Di conseguenza, molte impostazioni del motore di database precedente non sono valide. Consigliamo di condurre gli esperimenti iniziali con le impostazioni di configurazione di Aurora predefinite. Riapplica le impostazioni dell'ambiente corrente

solo in caso di colli di bottiglia di prestazioni e scalabilità. Se sei interessato, puoi approfondire questo argomento in Introduzione al motore di archiviazione Aurora sul blog del AWS database.

Aurora agevola il riutilizzo delle impostazioni di configurazione ottimali per una particolare applicazione o caso d'uso. Anziché modificare un file di configurazione separato per ciascuna istanza database, gestisci set di parametri che assegni a interi cluster o a specifiche istanze database. Ad esempio, l'impostazione del fuso orario si applica a tutte le istanze database nel cluster e puoi regolare l'impostazione della dimensione della cache della pagina per ciascuna istanza database.

Inizi con uno dei set di parametri predefiniti e applichi le modifiche solo ai parametri che devi ottimizzare. Per informazioni sull'utilizzo dei gruppi di parametri, consulta Parametri dell'istanza database e del cluster database di Amazon Aurora. Per le impostazioni di configurazione che sono o non sono applicabili ai cluster Aurora, consulta Parametri di configurazione Aurora MySQL o Amazon Aurora PostgreSQL parametri a seconda del motore di database in uso.

9. Connettere ad Aurora

Proprio come per l'impostazione dello schema e dei dati iniziali e l'esecuzione di query di esempio, puoi connetterti a diversi endpoint in un cluster Aurora. L'endpoint da utilizzare dipende dal fatto che l'operazione sia una lettura come l'istruzione `SELECT` oppure una scrittura come l'istruzione `CREATE` o `INSERT`. Quando incrementi il carico di lavoro su un cluster Aurora e sperimenti le funzionalità di Aurora, è importante che l'applicazione assegni ciascuna operazione all'endpoint appropriato.

Utilizzando l'endpoint del cluster per operazioni di scrittura, ti connetti sempre a un'istanza database nel cluster che ha funzionalità di lettura e scrittura. Per impostazione predefinita, solo un'istanza database in un cluster Aurora ha funzionalità di lettura e scrittura. Questa istanza database è detta istanza primaria. Se l'istanza primaria originale diventa non disponibile, Aurora attiva un meccanismo di failover e un'istanza database diversa diventa quella primaria.

Analogamente, indirizzando istruzioni `SELECT` all'endpoint di lettura, distribuisci il lavoro di elaborazione delle query tra le istanze database nel cluster. Ogni connessione di lettura viene assegnata a un'istanza DB diversa utilizzando una risoluzione DNS round-robin. L'esecuzione della maggior parte del lavoro di interrogazione su DB Aurora Replicas di sola lettura riduce il carico sull'istanza principale, liberandola di gestione e istruzioni. DDL DML

L'utilizzo di questi endpoint riduce la dipendenza a nomi host hardcoded e aiuta l'applicazione a eseguire un ripristino più rapido in caso di errori delle istanze database.

> **Note**
>
> Inoltre Aurora consente di creare endpoint personalizzati. Questi endpoint non sono generalmente necessari durante un proof of concept.

Le repliche Aurora sono soggette a un ritardo di replica, anche se tale ritardo è di solito da 10 a 20 millisecondi. Puoi monitorare il ritardo di replica e stabilire se rientra nell'intervallo dei requisiti di coerenza dei tuoi dati. In alcuni casi, le query di lettura potrebbero richiedere una forte coerenza di lettura (coerenza). read-after-write In questi casi, puoi continuare a utilizzare l'endpoint del cluster anziché l'endpoint di lettura.

Per sfruttare appieno le funzionalità di Aurora per l'esecuzione in parallelo distribuita, potresti dover modificare la logica della connessione. L'obiettivo è evitare di inviare tutte le richieste di lettura all'istanza primaria. Le repliche Aurora di sola lettura sono in stand by, con tutti gli stessi dati, pronte per gestire le istruzioni `SELECT`. Codifica la logica dell'applicazione per utilizzare l'endpoint appropriato per ciascun tipo di operazione. Segui queste linee guida generali:

- Evita di utilizzare un'unica stringa di connessione hardcoded per tutte le sessioni di database.
- Se possibile, includi operazioni di scrittura come DDL e DML istruzioni nelle funzioni nel codice dell'applicazione client. In questo modo, diversi tipi di applicazioni possono utilizzare connessioni specifiche.
- Creare funzioni separate per le operazioni di query. Aurora assegna ciascuna nuova connessione all'endpoint di lettura a una replica Aurora diversa per bilanciare il carico per applicazioni con attività di lettura intensiva.
- Per le operazioni che implicano set di query, chiudi e riapri la connessione all'endpoint di lettura quando ciascun set di query correlate è terminato. Utilizza il pooling della connessione se tale funzionalità è disponibile nello stack del software. L'indirizzamento di query a diverse connessioni aiuta Aurora a distribuire il carico di lavoro di lettura tra le istanze database nel cluster.

Per informazioni generali sulla gestione della connessione e sugli endpoint per Aurora, consulta Connessione a un cluster database Amazon Aurora. Per un approfondimento su questo argomento, consulta Aurora My SQL database administrator's handbook — Connection management.

10. Eseguire il carico di lavoro

Dopo aver eseguito le impostazioni di schema, dati e configurazione, puoi iniziare a esercitarti con il cluster eseguendo il carico di lavoro. Nel proof of concept utilizza un carico di lavoro che rispecchi i numerosi aspetti del tuo carico di lavoro di produzione. Consigliamo sempre di prendere decisioni sulle prestazioni utilizzando test e carichi di lavoro reali anziché benchmark sintetici come sysbench o -C. TPC Ovunque sia possibile, raccogliete le misurazioni in base al vostro schema, ai modelli di query e al volume di utilizzo.

Nella misura in cui risulta utile, replica le effettive condizioni in cui verrà eseguita l'applicazione. Ad esempio, in genere esegui il codice dell'applicazione su EC2 istanze Amazon nella stessa AWS regione e sullo stesso cloud privato virtuale (VPC) del cluster Aurora. Se la tua applicazione di produzione viene eseguita su più EC2 istanze che si estendono su più zone di disponibilità, configura l' proof-of-concept ambiente nello stesso modo. Per ulteriori informazioni sulle AWS regioni, consulta Regioni e zone di disponibilità nella Amazon RDS User Guide. Per ulteriori informazioni sul VPC servizio Amazon, consulta What is AmazonVPC? nella Amazon VPC User Guide.

Dopo aver verificato il funzionamento delle funzionalità di base dell'applicazione e la possibilità di accedere ai dati tramite Aurora, puoi fare pratica con gli aspetti del cluster Aurora. Alcune funzionalità che potresti voler provare sono connessioni simultanee con il bilanciamento del carico, transazioni simultanee e replica automatica.

A questo punto, i meccanismi di trasferimento dei dati dovrebbero essere familiari e puoi quindi eseguire i test con una maggiore quantità di dati campione.

Questa è la fase in cui puoi vedere gli effetti della modifica delle impostazioni di configurazione come i limiti di memoria e di connessione. Rivedi le procedure che hai esplorato in 8. Specificare le impostazioni di configurazione.

Puoi inoltre sperimentare meccanismi come la creazione e il ripristino di snapshot. Ad esempio, puoi creare cluster con classi di AWS istanze diverse, numeri di AWS repliche e così via. In ogni cluster, puoi quindi ripristinare la stessa snapshot che contiene lo schema e tutti i dati. Per i dettagli su questo ciclo, consulta Creazione di uno snapshot del cluster database e Ripristino da uno snapshot cluster database.

11. Misurare le prestazioni

Le best practice in questa area sono state concepite per assicurare l'impostazione di tutti gli strumenti e i processi giusti per isolare velocemente comportamenti anomali durante le operazioni del carico di

lavoro. Servono inoltre a constatare la possibilità di identificare in modo attendibile eventuali cause applicabili.

Puoi sempre visualizzare lo stato corrente del cluster o esaminare le tendenze nel tempo, analizzando la scheda Monitoring (Monitoraggio). Questa scheda è disponibile nella pagina dei dettagli della console per ciascun cluster Aurora o istanza database. Visualizza le metriche del servizio di CloudWatch monitoraggio Amazon sotto forma di grafici. Puoi filtrare i parametri per nome, per istanza database e per periodo di tempo.

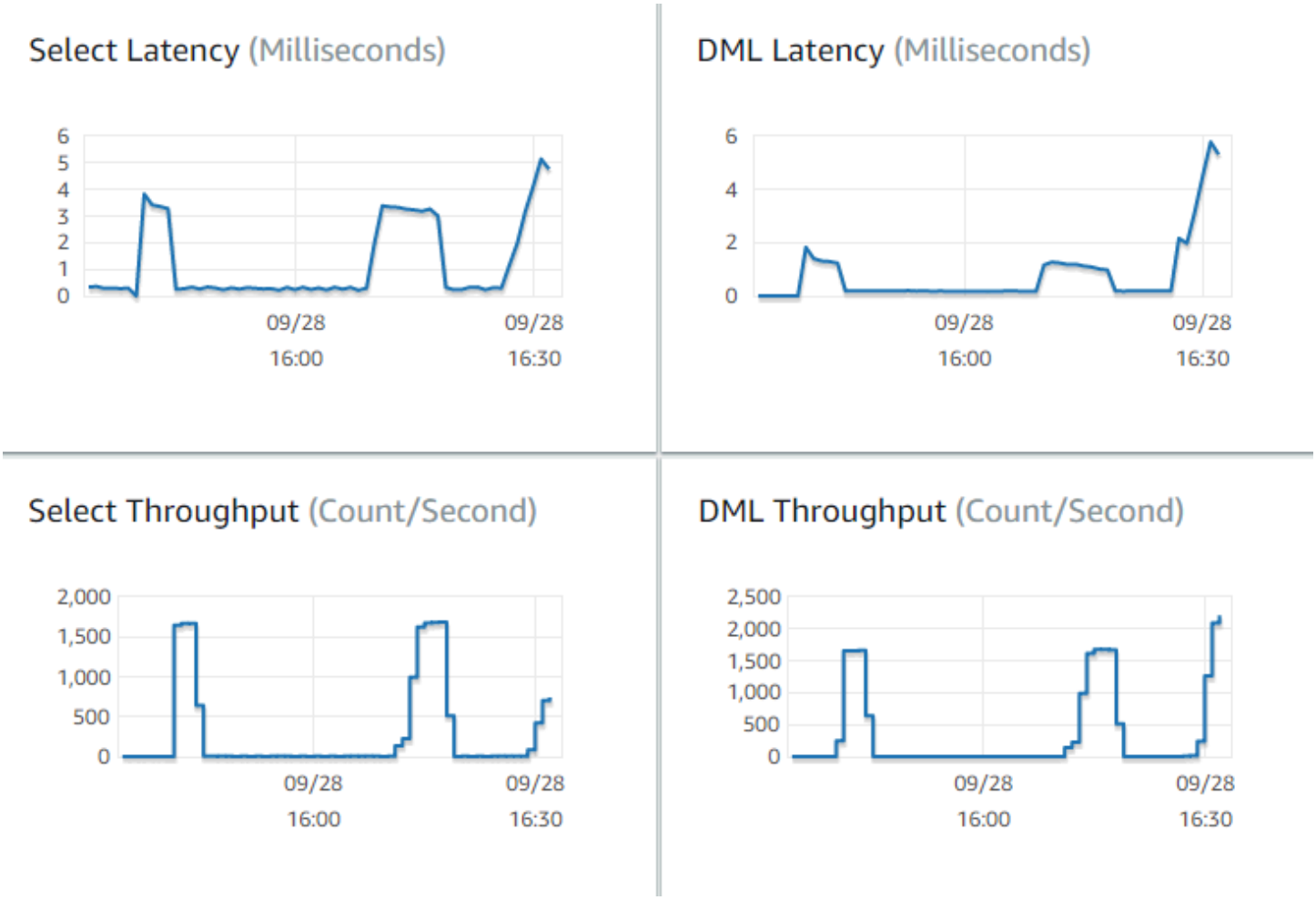
Per avere più scelte nella scheda Monitoring (Monitoraggio), abilita Enhanced Monitoring (Monitoraggio avanzato) e Performance Insights nelle impostazioni del cluster. Inoltre, se non abiliti queste scelte al momento della configurazione del cluster, puoi farlo in un secondo momento.

Per misurare le prestazioni, puoi utilizzare i grafici che mostrano l'attività dell'intero cluster Aurora. Puoi verificare se le repliche Aurora hanno simili tempi di caricamento e risposta. Puoi anche visualizzare la suddivisione del lavoro tra l'istanza primaria di lettura e scrittura e le repliche Aurora di sola lettura. Se le istanze database non sono bilanciate o nel caso in cui un problema interessi solo un'istanza database, puoi esaminare la scheda Monitoring (Monitoraggio) di quella specifica istanza.

Dopo l'impostazione dell'ambiente e del carico di lavoro effettivo per emulare l'applicazione di produzione, puoi misurare le prestazioni di Aurora. Le domande più importanti da porsi sono le seguenti:

- Quante query al secondo elabora Aurora? Puoi esaminare i parametri Throughput per visualizzare i dati per diversi tipi di operazioni.
- Quanto tempo impiega in media Aurora per elaborare una determinata query? Puoi esaminare i parametri Latency (Latenza) per visualizzare i dati per diversi tipi di operazioni.

Per visualizzare i parametri di throughput e latenza, controlla la scheda Monitoraggio per un determinato cluster Aurora nella console Amazon. RDS La schermata seguente mostra un esempio delle metriche Select Latency, Latency, DML Select Throughput e Throughput nella scheda Monitoraggio. DML



Se possibile, stabilisci valori di base per questi parametri nell'ambiente corrente. Se ciò non è pratico, crea una baseline nel cluster Aurora eseguendo un carico di lavoro equivalente all'applicazione di produzione. Ad esempio, esegui il carico di lavoro Aurora con un numero simile di utenti e query simultanei. Osserva quindi in che modo i valori cambiano durante l'esperimento con diverse classi di istanze, dimensioni del cluster, impostazioni di configurazione e così via.

Se i valori del throughput sono inferiori a quanto previsto, analizza ulteriormente i fattori che influenzano le prestazioni del database per il carico di lavoro. Analogamente, se i valori della latenza sono superiori rispetto a quanto previsto, effettua ulteriori analisi. A tale scopo, monitora le metriche secondarie per il server DB (CPU, memoria e così via). Puoi vedere se le istanze database sono vicine ai loro limiti. Inoltre, puoi vedere quanta capacità aggiuntiva hanno le istanze database per gestire più query simultanee, query su tabelle di grandi dimensioni e così via.

> **Tip**
>
> Per rilevare i valori delle metriche che non rientrano negli intervalli previsti, imposta gli allarmi CloudWatch .

Quando valuti le dimensioni e la capacità ideali del cluster Aurora, puoi trovare la configurazione che raggiunge le massime prestazioni dell'applicazione senza l'overprovisioning delle risorse. Un fattore importante è trovare le dimensioni giuste per le istanze database nel cluster Aurora. Inizia selezionando un'istanza di dimensioni CPU e capacità di memoria simili a quelle dell'ambiente di produzione corrente. Raccogli i valori di throughput e latenza per il carico di lavoro con queste dimensioni dell'istanza. Dopo di che, aumenta l'istanza alla dimensione successiva più grande. Verifica se i valori di throughput e latenza migliorano. Inoltre, riduci le dimensioni dell'istanza per vedere se i valori di latenza e throughput rimangono invariati. L'obiettivo è ottenere il massimo throughput e la minima latenza sull'istanza più piccola possibile.

> **Tip**
>
> Dimensiona i cluster Aurora e le istanze database correlate con abbastanza capacità esistente per gestire picchi di traffico improvvisi e imprevedibili. Per i database mission-critical, lasciate almeno il 20% di spazio libero CPU e di memoria.

Esegui test delle prestazioni abbastanza lunghi da misurare le prestazioni del database in uno stato attivo e costante. Potresti dover eseguire il carico di lavoro per molti minuti o anche qualche ora prima di raggiungere questo stato costante. All'inizio di un'esecuzione è normale avere delle variazioni. Queste variazioni avvengono perché ciascuna replica Aurora attiva le proprie cache in base alle query `SELECT` che gestisce.

Aurora offre massime prestazioni con carichi di lavoro transazionali che implicano più utenti e query simultanei. Per accertarti che il carico sia sufficiente per prestazioni ottimali, esegui benchmark che utilizzano multithreading o esegui più istanze dei test prestazionali simultaneamente. Misura le prestazioni con centinaia o anche migliaia di thread client simultanei. Simula il numero di thread simultanei che prevedi nel tuo ambiente di produzione. Potresti anche eseguire ulteriori stress test con più thread per misurare la scalabilità di Aurora.

12. Fare pratica con la disponibilità elevata di Aurora

Molte delle principali funzionalità di Aurora implicano disponibilità elevata. Queste funzionalità includono la replica automatica, il failover automatico, i backup automatici con point-in-time ripristino e la possibilità di aggiungere istanze DB al cluster. La sicurezza e l'affidabilità derivanti da funzionalità come queste sono importanti per le applicazioni mission critical.

La valutazione di tali funzionalità richiede un determinato approccio. Nelle attività precedenti, come la misurazione delle prestazioni, puoi osservare le prestazioni del sistema quando tutto funziona correttamente. Il test della disponibilità elevata richiede di ipotizzare il peggiore comportamento possibile. Occorre considerare vari tipi di errore, anche se tali condizioni sono rare. Potresti creare intenzionalmente dei problemi per accertarti che il sistema effettui il ripristino in modo corretto e veloce.

> **Tip**
>
> Per una dimostrazione del concetto, configura tutte le istanze DB in un cluster Aurora con la AWS stessa classe di istanza. In questo modo è possibile provare la funzionalità di disponibilità di Aurora senza importanti modifiche alle prestazioni e alla scalabilità quando porti le istanze database offline per simulare gli errori.

Consigliamo di utilizzare almeno due istanze in ciascun cluster Aurora. Le istanze DB in un cluster Aurora possono estendersi su un massimo di tre zone di disponibilità (). AZs Posiziona ciascuna delle prime due o tre istanze database in una zona di disponibilità diversa. Quando inizi a utilizzare cluster più grandi, distribuisci le istanze DB in tutta la tua regione. AZs AWS In questo modo, aumenta la tolleranza agli errori. Anche se un problema colpisce un'intera zona di disponibilità, Aurora esegue il failover su un'istanza database in un'altra zona di disponibilità. Se gestisci un cluster con più di tre istanze, distribuisci le istanze DB nel modo più uniforme possibile su tutte e tre. AZs

> **Tip**
>
> Lo storage di un cluster Aurora è indipendente dalle istanze database. Lo storage per ogni cluster Aurora si estende sempre su tre. AZs
> Quando testi le funzionalità di disponibilità elevata, utilizza sempre istanze database con capacità identica nel cluster del test. Ciò permette di evitare cambiamenti imprevisti di prestazioni, latenza e così via quando un'istanza database prende il posto di un'altra.

Per capire come simulare condizioni di errore per testare le funzionalità di disponibilità elevata, consulta Test di Amazon Aurora My SQL using Fault Injection.

Come parte dell' proof-of-concept esercizio, uno degli obiettivi è quello di trovare il numero ideale di istanze DB e la classe di istanze ottimale per tali istanze DB. Ciò richiede il bilanciamento dei requisiti di disponibilità elevata e delle prestazioni.

Per Aurora, più istanze database sono presenti in un cluster, maggiori sono i vantaggi per la disponibilità elevata. Inoltre, avere più istanze database migliora la scalabilità delle applicazioni a lettura intensiva. Aurora può distribuire più connessioni per query `SELECT` tra le repliche Aurora di sola lettura.

Dall'altro lato, limitare il numero di istanze database riduce il traffico di replica dal nodo primario. Il traffico di replica consuma larghezza di banda di rete, altro aspetto delle prestazioni complessive e della scalabilità. Pertanto, per OLTP le applicazioni che richiedono un uso intensivo di scrittura, è preferibile disporre di un numero inferiore di istanze DB di grandi dimensioni piuttosto che di molte istanze DB di piccole dimensioni.

In un tipico cluster Aurora, un'istanza DB (l'istanza principale) gestisce tutte le istruzioni DDL andDML. Le altre istanze database (repliche Aurora) gestiscono solo le istruzioni `SELECT`. Sebbene le istanze database non svolgano esattamente la stessa quantità di lavoro, consigliamo di utilizzare la stessa classe di istanza per tutte le istanze database nel cluster. In questo modo, se si verifica un errore e Aurora promuove una delle istanze database di sola lettura a nuova istanza primaria, l'istanza primaria ha la stessa capacità di prima.

Se devi utilizzare istanze database con capacità diverse nello stesso cluster, imposta i tier di failover per le istanze database. Questi tier determinano l'ordine in cui le repliche Aurora vengono promosse dal meccanismo di failover. Le istanze database che sono molto più grandi o più piccole delle altre vanno collocate in un tier di failover inferiore. Ciò assicura che verranno scelte per ultime per la promozione.

Sfrutta le funzionalità di ripristino dei dati di Aurora, come il point-in-time ripristino automatico, le istantanee e il ripristino manuali e il backtracking del cluster. Se necessario, copia le istantanee in altre AWS regioni e ripristinale in altre AWS regioni per simulare gli scenari di DR.

Analizza i requisiti della tua organizzazione per quanto riguarda l'obiettivo del tempo di ripristino (RTO), l'obiettivo del punto di ripristino (RPO) e la ridondanza geografica. La maggior parte delle organizzazioni raggruppa questi elementi nell'ampia categoria del disaster recovery. Valuta le funzionalità di alta disponibilità di Aurora descritte in questa sezione nel contesto del processo di disaster recovery per assicurarti che RTO i tuoi RPO requisiti siano soddisfatti.

13. Cosa fare in seguito

Al termine di un proof-of-concept processo di successo, confermi che Aurora è la soluzione adatta a te in base al carico di lavoro previsto. Durante il processo precedente, hai verificato il funzionamento di Aurora in un ambiente operativo realistico e lo hai valutato a fronte dei tuoi criteri di successo.

Dopo aver impostato e avviato l'ambiente di database con Aurora, puoi procedere a una valutazione più dettagliata, che porta alla migrazione finale e alla distribuzione della produzione. A seconda della situazione, questi altri passaggi potrebbero essere inclusi o meno nel processo. proof-of-concept Per i dettagli sulle attività di migrazione e porting, consulta il AWS white paper Aurora migration handbook.

In un altro passaggio successivo, prendi in considerazione le configurazioni di sicurezza pertinenti per il tuo carico di lavoro e concepite per rispondere ai requisiti di sicurezza in un ambiente di produzione. Pianifica i controlli da predisporre per proteggere l'accesso alle credenziali degli utenti master del cluster Aurora. Definisci i ruoli e le responsabilità degli utenti del database per controllare l'accesso ai dati archiviati nel cluster Aurora. Prendi in considerazione i requisiti di accesso al database per le applicazioni, gli script e gli strumenti o i servizi di terze parti. Esplora AWS servizi e funzionalità come l'autenticazione and (). AWS Secrets Manager AWS Identity and Access Management IAM

A questo punto, le procedure e le best practice per l'esecuzione di test di benchmark con Aurora dovrebbero essere chiare. Potresti dover effettuare un'ulteriore ottimizzazione delle prestazioni. Per i dettagli, consulta Gestione delle prestazioni e del dimensionamento dei cluster DB Aurora, Miglioramenti alle prestazioni di Amazon Aurora MySQL, Prestazioni e scalabilità per Amazon Aurora PostgreSQL e Monitoraggio del carico DB con Performance Insights su Amazon Aurora. Se effettui un'ulteriore ottimizzazione, assicurati di conoscere bene i parametri che hai raccolto durante il proof of concept. Come passaggio successivo, potresti creare nuovi cluster con diverse scelte per le impostazioni di configurazione, il motore di database e la versione del database. In alternativa, potresti creare tipi di cluster Aurora per rispondere ai requisiti di specifici casi d'uso.

Ad esempio, puoi esplorare i cluster di query parallele Aurora per applicazioni ibride di transazione/elaborazione analitica (). HTAP Se un'ampia distribuzione geografica è fondamentale per il disaster recovery o per ridurre al minimo la latenza, puoi esaminare Aurora Global Database. Se il tuo carico di lavoro è intermittente o se utilizzi Aurora in uno scenario di sviluppo/test, puoi esaminare i cluster Aurora Serverless.

I cluster di produzione potrebbero inoltre dover gestire elevati volumi di connessioni in entrata. Per apprendere queste tecniche, consulta il AWS white paper Aurora SQL My database administrator's handbook — Connection management.

Se, dopo il proof of concept, stabilisci che il tuo caso d'uso non è adatto per Aurora, valuta questi altri sevizi AWS :

- Per i casi d'uso puramente analitici, i carichi di lavoro traggono vantaggio da un formato di archiviazione colonnare e da altre funzionalità più adatte ai carichi di lavoro. OLAP AWS i servizi che si occupano di tali casi d'uso includono quanto segue:
  - Amazon Redshift
  - Amazon EMR
  - Amazon Athena
- Molti carichi di lavoro traggono vantaggio da una combinazione di Aurora con uno o più di questi servizi. Puoi spostare i dati tra questi servizi utilizzando quanto segue:
  - AWS Glue
  - AWS DMS
  - Importazione da Amazon S3, come descritto nella Guida per l'utente di Amazon Aurora
  - Esportazione verso Amazon S3, come descritto nella Guida per l'utente di Amazon Aurora
  - Molti altri ETL strumenti popolari

Sicurezza in Amazon Aurora

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per informazioni sui programmi di conformità che si applicano ad Aurora (Aurora), AWS consulta i servizi rientranti nell'ambito del programma di conformità.
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda, nonché le leggi e le normative applicabili.

Questa documentazione ti aiuta a capire come applicare il modello di responsabilità condivisa quando usi . I seguenti argomenti mostrano come configurare Aurora per soddisfare i tuoi obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon Aurora.

Puoi gestire l'accesso alle tue risorse Amazon Aurora e ai tuoi database su un cluster di istanze . Il metodo utilizzato per gestire l'accesso dipende dal tipo di attività che l'utente deve eseguire con :

- Esegui il tuo cluster di DB in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC per il massimo controllo possibile degli accessi alla rete. Per ulteriori informazioni sulla creazione di un cluster di DB in un VPC, vedere. Amazon VPC e Aurora
- Utilizza le policy AWS Identity and Access Management (IAM) per assegnare autorizzazioni che determinano chi è autorizzato a gestire le risorse Amazon . Ad esempio, puoi utilizzare IAM per determinare chi è autorizzato a creare, descrivere, modificare ed eliminare DB, cluster, taggare, risorse o modificare gruppi di sicurezza.

  Per esaminare gli esempi di policy IAM, consulta.
- Usa i gruppi di sicurezza per controllare quali indirizzi IP o EC2 istanze Amazon possono connettersi ai tuoi database su un cluster di DB. Quando crei per la prima volta un cluster di DB, il

relativo firewall impedisce l'accesso al database se non tramite regole specificate da un gruppo di sicurezza associato.

- Usa connessioni Secure Socket Layer (SSL) o Transport Layer Security (TLS) con cluster di DB che eseguono Aurora MySQL o Aurora PostgreSQL. Per ulteriori informazioni sull'utilizzo di SSL/TLS con un cluster DB, consulta. Utilizzo SSL/TLS per crittografare una connessione a un'
- Usa la crittografia (Amazon Aurora) per proteggere le istanze DB cluster di database e gli snapshot inattivi. Per ulteriori informazioni, consulta Crittografia delle risorse Amazon Aurora.
- Queste funzionalità funzionano come se il database si trovasse sulla rete locale.

  Per informazioni sulla sicurezza con Aurora MySQL, consulta. Sicurezza con Amazon Aurora MySQL Per informazioni sulla sicurezza con Aurora PostgreSQL, consulta. Sicurezza con Amazon Aurora PostgreSQL

Aurora fa parte del servizio di database gestito Amazon Relational Database Service (Amazon RDS). Amazon RDS è un servizio Web che semplifica la configurazione, l'uso e il dimensionamento dei database relazionali nel cloud. Se è la prima volta che utilizzi Amazon RDS, consulta la Guida per l'utente di Amazon RDS.

Aurora include un sottosistema di storage ad alte prestazioni. I relativi motori di database compatibili con MySQL e PostgreSQL sono personalizzati per beneficiare dello storage distribuito e rapido. Aurora inoltre automatizza ed esegue la standardizzazione del clustering e della replica di database, ovvero due aspetti in genere tra i più impegnativi nell'ambito della configurazione e dell'amministrazione dei database.

Sia per Amazon RDS che per Aurora, puoi accedere all'API RDS a livello di codice e puoi utilizzarla per accedere all'API RDS in modo AWS CLI interattivo. Alcune operazioni API RDS e comandi AWS CLI si applicano sia a Amazon RDS che ad Aurora, mentre altri si applicano a uno o all'altro. Per informazioni sulle operazioni API RDS, consulta Documentazione di riferimento dell'API Amazon RDS. Per ulteriori informazioni su AWS CLI, consulta il AWS Command Line Interface riferimento per Amazon RDS.

> **Note**
>
> Devi configurare la sicurezza solo per i tuoi casi d'uso. Non devi configurare l'accesso di sicurezza per i processi gestiti da Amazon Aurora. Sono inclusi la creazione di backup, il failover automatico e altri processi.

Per ulteriori informazioni sulla gestione dell'accesso alle risorse Amazon Aurora e ai database su un cluster di istanze , consulta i seguenti argomenti.

Argomenti

- Autenticazione del database con Amazon Aurora
- Gestione delle password con Amazon Aurora e AWS Secrets Manager
- Protezione dei dati in Amazon RDS
- Gestione delle identità e degli accessi per Amazon Aurora
- Registrazione e monitoraggio in Amazon Aurora
- Convalida della conformità per
- Resilienza in Amazon Aurora
- Sicurezza dell'infrastruttura in Aurora
- API Amazon RDS ed endpoint VPC dell'interfaccia (AWS PrivateLink)
- Best practice relative alla sicurezza di Amazon Aurora
- Controllo dell'accesso con i gruppi di sicurezza
- Privilegi dell'account utente master
- Utilizzo di ruoli collegati ai servizi per Amazon Aurora
- Amazon VPC e Aurora

Autenticazione del database con Amazon Aurora

Amazon Aurora supporta diversi modi per autenticare gli utenti del database.

L'autenticazione con password è disponibile per impostazione predefinita per tutti i cluster database. Per Aurora MySQL e Aurora PostgreSQL, puoi aggiungere sia l'autenticazione del database IAM che l'autenticazione Kerberos per lo stesso cluster di database.

L'autenticazione con password, Kerberos e del database IAM utilizzano diversi metodi di autenticazione nel database. Pertanto, un utente specifico può accedere a un database utilizzando un solo metodo di autenticazione.

Per PostgreSQL, utilizza solo una delle seguenti impostazioni di ruolo per un utente di un database specifico:

- Per utilizzare l'autenticazione del database IAM, assegna all'utente il ruolo `rds_iam`.

- Per utilizzare l'autenticazione Kerberos, assegna all'utente il ruolo `rds_ad`.
- Per utilizzare l'autenticazione con password, non assegnare i ruoli `rds_iam` o `rds_ad`.

Non assegnare entrambi i ruoli `rds_iam` e `rds_ad` a un utente di un database PostgreSQL direttamente o indirettamente mediante l'accesso di concessione nidificato. Se all'utente master, viene aggiunto il ruolo `rds_iam`, l'autenticazione IAM ha la precedenza sull'autenticazione con password, quindi l'utente master dovrà accedere come utente IAM.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

Argomenti

- Autenticazione password
- Autenticazione del database IAM
- Autenticazione Kerberos

Autenticazione password

Con autenticazione con password il database esegue tutta l'amministrazione degli account utente. È possibile creare utenti con istruzioni SQL come `CREATE USER`, con la clausola appropriata richiesta dal motore DB per specificare le password. Ad esempio, in MySQL l'istruzione `CREATE USER` *`name`* `IDENTIFIED BY` *`password`* è, mentre in PostgreSQL l'istruzione è. `CREATE USER` *`name`* `WITH PASSWORD` *`password`*

Con l'autenticazione con password, il database controlla e autentica gli account utente. Se un motore DB dispone di potenti funzionalità di gestione delle password, può migliorare la sicurezza. L'autenticazione del database potrebbe essere più semplice da amministrare utilizzando l'autenticazione con password quando si dispone di comunità di utenti di piccole dimensioni. Poiché in questo caso vengono generate password con testo non crittografato, l'integrazione con può migliorare la sicurezza. AWS Secrets Manager

Per informazioni sull'uso di Secrets Manager con Amazon Aurora, consulta Creazione di un segreto di base e Rotating secret per i database Amazon RDS supportati nella Guida per l'utente.AWS

Secrets Manager Per informazioni sul recupero a livello di programmazione dei segreti nelle applicazioni personalizzate, consulta Recupero del valore segreto nella Guida per l'utente di AWS Secrets Manager .

Autenticazione del database IAM

Puoi autenticarti nel tuo cluster di DB utilizzando l'autenticazione del database AWS Identity and Access Management (IAM). Con questo metodo di autenticazione, non è necessario utilizzare una password quando ci si connette a un cluster di DB. Utilizzi invece un token di autenticazione.

Per ulteriori informazioni sull'autenticazione del database IAM, incluse le informazioni sulla disponibilità per motori DB specifici, consultaAutenticazione del database IAM .

Autenticazione Kerberos

Amazon Aurora Amazon supporta l'autenticazione esterna degli utenti del database tramite Kerberos e Microsoft Active Directory. Kerberos è un protocollo di autenticazione di rete che utilizza ticket e crittografia a chiave simmetrica eliminando la necessità di scambiare password sulla rete. È stato integrato in Microsoft Active Directory ed è progettato per autenticare gli utenti su risorse di rete, ad esempio i database.

Amazon Aurora Il supporto Amazon per Kerberos e Active Directory offre i vantaggi del single sign-on e dell'autenticazione centralizzata degli utenti del database. Puoi mantenere le tue credenziali utente in Active Directory.

Per utilizzare le credenziali dell'Active Directory autogestito, è necessario impostare una relazione di trust con AWS Directory Service Microsoft Active Directory a cui è unito il cluster DB dell' DB.

di trust tra foreste unidirezionali e bidirezionali con autenticazione a livello di foresta o autenticazione selettiva.

In alcuni scenari, è possibile configurare l'autenticazione Kerberos tramite una relazione di trust esterna. Ciò richiede che l'Active Directory autogestito disponga di impostazioni aggiuntive. Ciò include, a titolo esemplificativo ma non esaustivo, Kerberos Forest Search Order.

Aurora supporta l'autenticazione Kerberos per i cluster di database Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, vedere Utilizzo dell'autenticazione Kerberos per Aurora My SQL e. Utilizzo di Autenticazione Kerberos con Aurora PostgreSQL

Gestione delle password con Amazon Aurora e AWS Secrets Manager

Amazon Aurora si integra con Secrets Manager per gestire le password degli utenti master per .

Argomenti

- Disponibilità di regioni e versioni
- Limitazioni per l'integrazione di Secrets Manager con Amazon Aurora
- Panoramica della gestione delle password degli utenti principali con AWS Secrets Manager
- Vantaggi della gestione delle password degli utenti master con Secrets Manager
- Autorizzazioni necessarie per l'integrazione di Secrets Manager
- Applicazione della gestione della password dell'utente principale in AWS Secrets Manager
- Gestione della password dell'utente master per un cluster database con Secrets Manager
- Rotazione del segreto della password dell'utente master per un cluster database
- Visualizzazione dei dettagli di un segreto per un cluster database

Disponibilità di regioni e versioni

Il supporto varia a seconda delle versioni specifiche di ciascun motore di database e a seconda delle Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e regioni con l'integrazione di Secrets Manager con Amazon Aurora, consulta Regioni supportate e motori Aurora DB per l'integrazione di Secrets Manager.

Limitazioni per l'integrazione di Secrets Manager con Amazon Aurora

La gestione delle password degli utenti master con Secrets Manager non è supportata per le seguenti funzionalità:

- Implementazioni blu/verde di Amazon RDS
- Cluster database che fanno parte di un database globale Aurora
- Cluster di database Aurora Serverless v1
- Aurora ha letto le repliche

Panoramica della gestione delle password degli utenti principali con AWS Secrets Manager

Con AWS Secrets Manager, puoi sostituire le credenziali codificate nel codice, incluse le password del database, con una chiamata API a Secrets Manager per recuperare il segreto a livello di codice. Per ulteriori informazioni su Secrets Manager, consultare la Guida per l'utente di AWS Secrets Manager.

Quando memorizzi i segreti del database in Secrets Manager, ti vengono Account AWS addebitati dei costi. Per informazioni sui prezzi, consulta Prezzi di AWS Secrets Manager.

Puoi specificare che Aurora gestisca la password dell'utente master in Secrets Manager per un database Amazon Aurora quando esegui una delle seguenti operazioni:

- Crea un cluster DB
-
- Ripristina un cluster DB da Amazon S3 (solo Aurora MySQL)

Quando specifichi che Aurora gestisce la password dell'utente master in Secrets Manager, Aurora genera la password e la memorizza in Secrets Manager. Puoi interagire direttamente con il segreto per recuperare le credenziali dell'utente master. Puoi anche specificare una chiave gestita dal cliente per crittografare il segreto o utilizzare la chiave KMS fornita da Secrets Manager.

Aurora gestisce le impostazioni del segreto e lo ruota ogni sette giorni per impostazione predefinita. È possibile modificare alcune impostazioni, ad esempio il programma di rotazione. Se si elimina un cluster database che gestisce un segreto in Secrets Manager, vengono eliminati anche il segreto e i metadati associati.

Per connetterti a con le credenziali in un segreto, puoi recuperare il segreto da Secrets Manager. Per ulteriori informazioni, consulta Recupera segreti da AWS Secrets Manager e Connettiti a un database SQL con credenziali in un AWS Secrets Manager segreto nella Guida per l'AWS Secrets Manager utente.

Vantaggi della gestione delle password degli utenti master con Secrets Manager

La gestione delle password degli utenti master Aurora con Secrets Manager offre i seguenti vantaggi:

- Aurora genera automaticamente le credenziali del database.
- Aurora archivia e gestisce automaticamente le credenziali del database in. AWS Secrets Manager
- Aurora ruota regolarmente le credenziali del database, senza richiedere modifiche all'applicazione.
- Secrets Manager protegge le credenziali del database dall'accesso umano e dalla visualizzazione in testo normale.
- Secrets Manager consente il recupero delle credenziali del database nei segreti per le connessioni al database.
- Secrets Manager consente un controllo dettagliato dell'accesso alle credenziali del database nei segreti utilizzando IAM.
- Facoltativamente, puoi separare la crittografia del database dalla crittografia delle credenziali con chiavi KMS diverse.
- Puoi eliminare la gestione manuale e la rotazione delle credenziali del database.
- Puoi monitorare facilmente le credenziali del database con AWS CloudTrail Amazon CloudWatch.

Per ulteriori informazioni sui vantaggi di Secrets Manager, consulta la Guida per l'utente di AWS Secrets Manager.

Autorizzazioni necessarie per l'integrazione di Secrets Manager

Gli utenti devono disporre delle autorizzazioni necessarie per eseguire le operazioni relative all'integrazione di Secrets Manager. Puoi creare le policy IAM che concedono l'autorizzazione per eseguire operazioni API specifiche sulle risorse indicate necessarie. Puoi quindi collegare tali policy ai ruoli o ai set di autorizzazioni IAM che richiedono le autorizzazioni. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon Aurora.

Per le operazioni di creazione, modifica o ripristino, l'utente che specifica che Aurora gestisce la password dell'utente master in Secrets Manager deve disporre delle autorizzazioni per eseguire le seguenti operazioni:

- `kms:DescribeKey`
- `secretsmanager:CreateSecret`
- `secretsmanager:TagResource`

L'`kms:DescribeKey`autorizzazione è necessaria per accedere alla chiave gestita dal cliente `MasterUserSecretKmsKeyId` e per descriverla. `aws/secretsmanager`

Per le operazioni di creazione, modifica o ripristino, l'utente che specifica la chiave gestita dal cliente per crittografare il segreto in Secrets Manager deve disporre delle autorizzazioni per eseguire le seguenti operazioni:

- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:CreateGrant`

Per le operazioni di modifica, l'utente che ruota la password dell'utente master in Secrets Manager deve disporre delle autorizzazioni per eseguire la seguente operazione:

- `secretsmanager:RotateSecret`

Applicazione della gestione della password dell'utente principale in AWS Secrets Manager

È possibile utilizzare le chiavi di condizione IAM per implementare la gestione da parte di Aurora della password dell'utente master in AWS Secrets Manager. La seguente politica non consente agli utenti di creare o ripristinare istanze DB o cluster DB o a meno che la password dell'utente principale non sia gestita da RDS in Secrets Manager.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

> **Note**
>
> Questa politica applica la gestione delle password al momento della creazione. AWS Secrets Manager Tuttavia, puoi comunque disabilitare l'integrazione di Secrets Manager e impostare manualmente una password master modificando il cluster.
> Per evitare questa procedura, includi `rds:ModifyDBInstance`, `rds:ModifyDBCluster` nel blocco operazione della policy. Tieni presente che in tal modo impedisci all'utente di applicare ulteriori modifiche ai cluster esistenti in cui non è abilitata l'integrazione di Secrets Manager.

Per ulteriori informazioni sull'utilizzo delle chiavi di condizione nelle policy IAM, consulta Chiavi relative alle condizioni delle politiche per Aurora e Policy di esempio: Utilizzo di chiavi di condizione.

Gestione della password dell'utente master per un cluster database con Secrets Manager

È possibile configurare la gestione Aurora della password dell'utente master in Secrets Manager eseguendo le seguenti operazioni:

- Creazione di un cluster database Amazon Aurora
- Modifica di un cluster database Amazon Aurora
- La migrazione di dati da un database MySQL esterno a un cluster database Amazon Aurora MySQL

È possibile utilizzare la console RDS AWS CLI, o l'API RDS per eseguire queste azioni.

Console

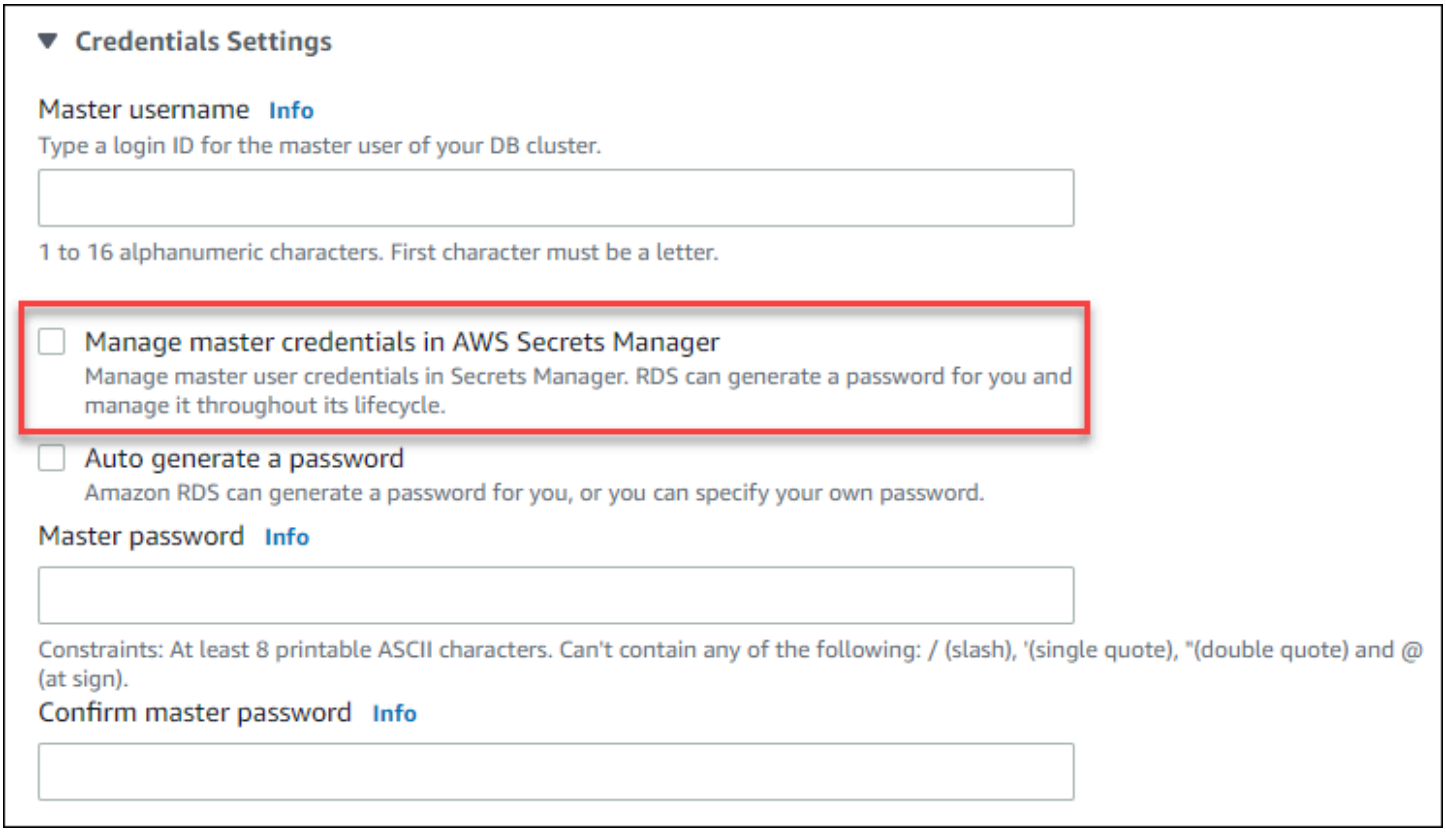
Segui le istruzioni per creare o modificare un cluster database con la console RDS:

- Creazione di un cluster di database
- Modifica di un'istanza database in un cluster database

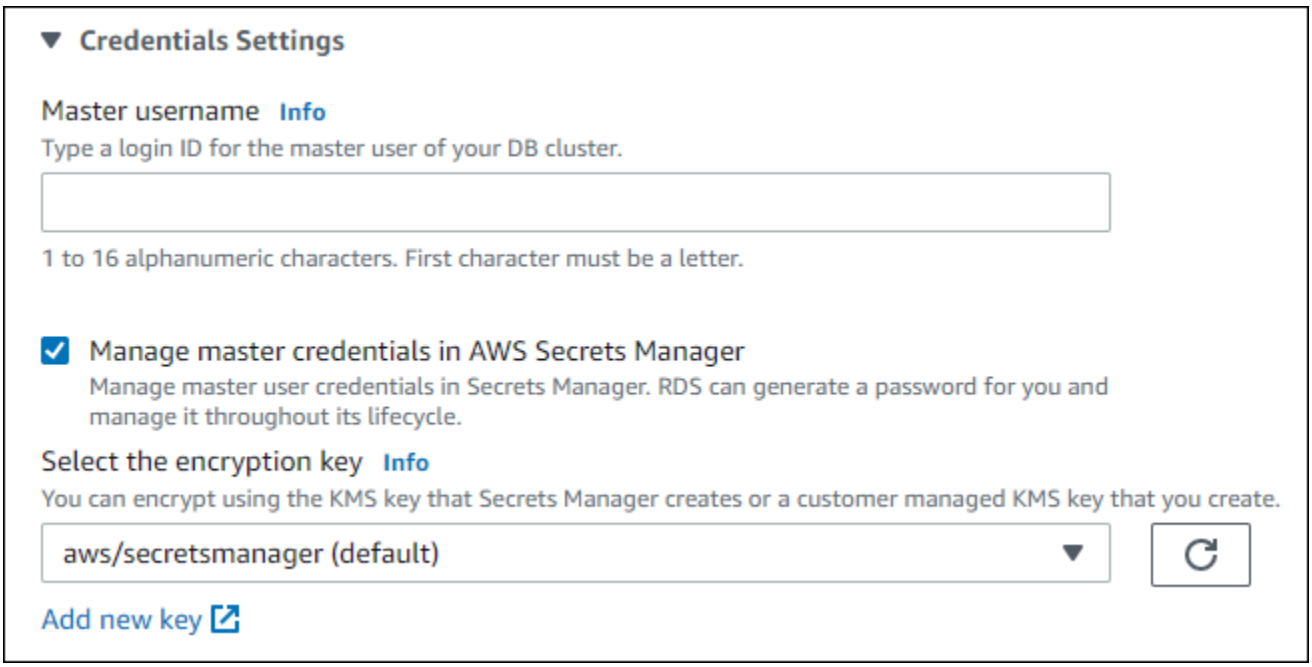
  Nella console RDS puoi modificare qualsiasi istanza database per specificare le impostazioni di gestione della password dell'utente master per l'intero cluster database.
- Ripristino di un cluster di database Amazon Aurora MySQL da un bucket Amazon S3

Quando usi la console RDS per eseguire una di queste operazioni, è possibile specificare che la password dell'utente master sia gestita da Aurora in Secrets Manager. A tale scopo durante la creazione o il ripristino di un cluster database, seleziona Manage master credentials in AWS Secrets Manager (Gestione credenziali master in AWS Secrets Manager) in Credential settings (Impostazioni credenziali). Quando modifichi un cluster database, seleziona Manage master credentials in AWS Secrets Manager (Gestione credenziali master in AWS Secrets Manager) in Settings (Impostazioni).

L'immagine seguente è un esempio di impostazione Manage master credentials in AWS Secrets Manager (Gestione credenziali master in AWS Secrets Manager) durante la creazione o il ripristino di un cluster database.



Quando selezioni questa opzione, Aurora genera la password dell'utente master e la gestisce per tutto il suo ciclo di vita in Secrets Manager.



Puoi scegliere di crittografare il segreto con una chiave KMS fornita da Secrets Manager o con una chiave gestita dal cliente creata da te. Dopo che Aurora gestisce le credenziali del database per un cluster database, non puoi modificare la chiave KMS utilizzata per crittografare il segreto.

Puoi scegliere altre impostazioni per soddisfare le tue esigenze.

Per ulteriori informazioni sulle impostazioni disponibili per la creazione di un cluster database, consulta Impostazioni per cluster di database Aurora. Per ulteriori informazioni sulle impostazioni disponibili per la modifica di un cluster database, consulta Impostazioni per Amazon Aurora.

AWS CLI

Per specificare che Aurora gestisce la password dell'utente master in Secrets Manager, imposta l'opzione `--manage-master-user-password` in uno dei seguenti comandi:

- create-db-cluster
- modify-db-cluster
- restore-db-cluster-from-s3

Quando si specifica l'opzione `--manage-master-user-password` in questi comandi, Aurora genera la password dell'utente master e la gestisce per tutto il suo ciclo di vita in Secrets Manager.

Per crittografare il segreto, è possibile specificare una chiave gestita dal cliente o utilizzare la chiave KMS predefinita fornita da Secrets Manager. Per specificare la chiave gestita dal cliente usa l'opzione

`--master-user-secret-kms-key-id`. L'identificatore della chiave AWS KMS è l'ARN della chiave, l'ID chiave, l'alias ARN o il nome alias per la chiave KMS. Per utilizzare una chiave KMS in un'altra chiave Account AWS, specifica la chiave ARN o l'alias ARN. Dopo che Aurora gestisce le credenziali del database per un cluster database, non puoi modificare la chiave KMS utilizzata per crittografare il segreto.

Puoi scegliere altre impostazioni per soddisfare le tue esigenze.

Per ulteriori informazioni sulle impostazioni disponibili per la creazione di un cluster database, consulta Impostazioni per cluster di database Aurora. Per ulteriori informazioni sulle impostazioni disponibili per la modifica di un cluster database, consulta Impostazioni per Amazon Aurora.

Questo esempio crea un cluster database e specifica che Aurora gestisce la password in Secrets Manager. Il segreto viene crittografato utilizzando la chiave KMS fornita da Secrets Manager.

Example

Per LinuxmacOS, oUnix:

```
aws rds create-db-cluster \
     --db-cluster-identifier sample-cluster \
     --engine aurora-mysql \
     --engine-version 8.0 \
     --master-username admin \
     --manage-master-user-password
```

Per Windows:

```
aws rds create-db-cluster ^
     --db-cluster-identifier sample-cluster ^
     --engine aurora-mysql ^
     --engine-version 8.0 ^
     --master-username admin ^
     --manage-master-user-password
```

API RDS

Per specificare che Aurora gestisce la password dell'utente master in Secrets Manager, imposta il parametro `ManageMasterUserPassword` su `true` in una delle seguenti operazioni:

- CreaDBCluster

- ModificaDBCluster
- Ripristina DBCluster da S3

Quando imposti il parametro `ManageMasterUserPassword` su `true` in una di queste operazioni, Aurora genera la password dell'utente master e la gestisce per tutto il suo ciclo di vita in Secrets Manager.

Per crittografare il segreto, è possibile specificare una chiave gestita dal cliente o utilizzare la chiave KMS predefinita fornita da Secrets Manager. Per specificare la chiave gestita dal cliente usa il parametro `MasterUserSecretKmsKeyId`. L'identificatore della chiave AWS KMS è l'ARN della chiave, l'ID chiave, l'alias ARN o il nome alias per la chiave KMS. Per usate una chiave KMS in un Account AWS diverso, specifica l'ARN della chiave o dell'alias. Dopo che Aurora gestisce le credenziali del database per un cluster database, non puoi modificare la chiave KMS utilizzata per crittografare il segreto.

Rotazione del segreto della password dell'utente master per un cluster database

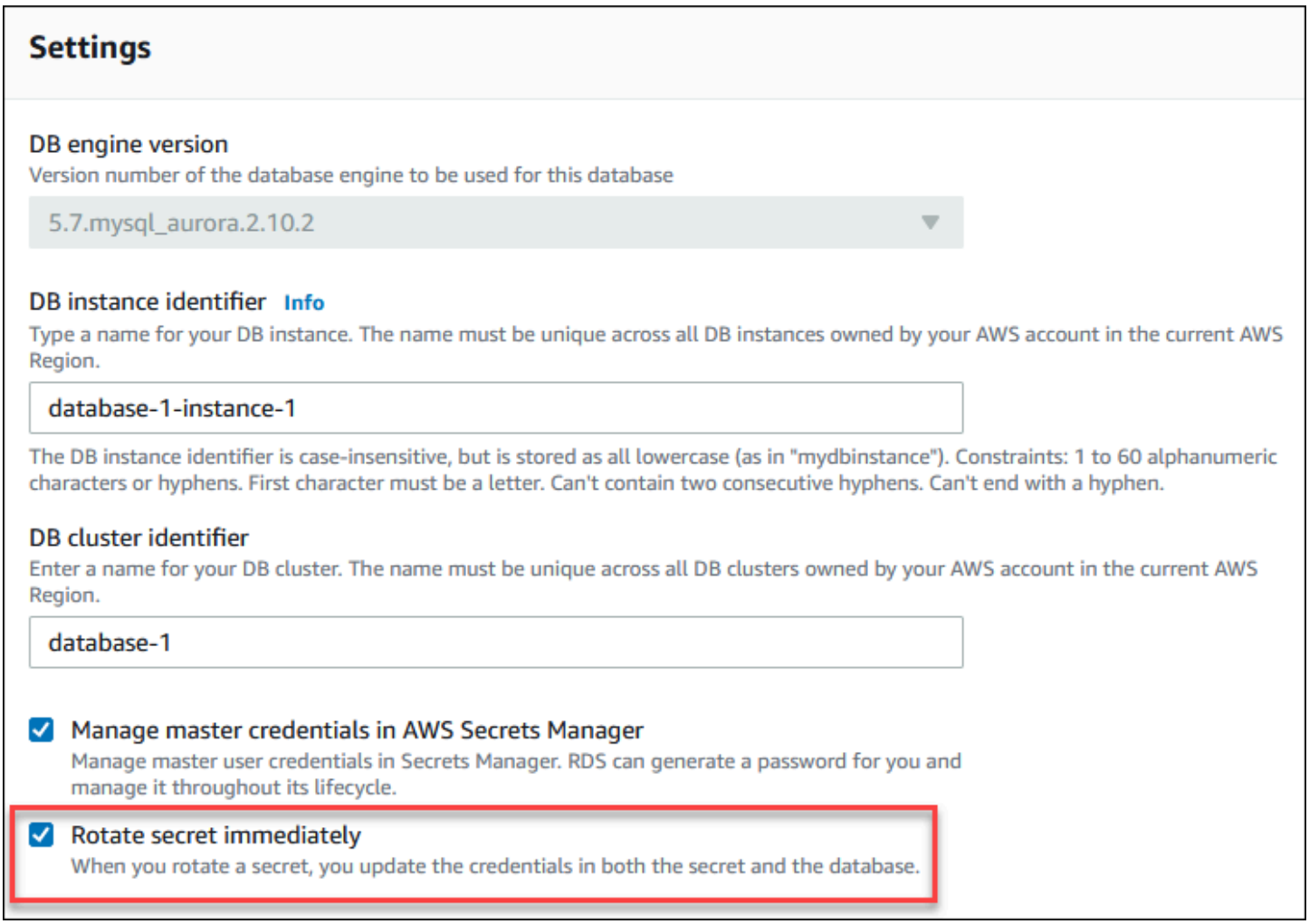
Quando Aurora ruota il segreto della password di un utente master, Secrets Manager genera una nuova versione del segreto esistente. La nuova versione del segreto contiene la nuova password dell'utente master. Amazon Aurora modifica la password dell'utente master per il cluster database in modo che corrisponda alla password per la nuova versione del segreto.

Puoi ruotare un segreto immediatamente invece di aspettare la rotazione programmata. Per ruotare il segreto della password dell'utente master in Secrets Manager, modifica il cluster database . Per informazioni sulla modifica di un cluster database, consulta Modifica di un cluster database Amazon Aurora.

È possibile ruotare immediatamente una password utente principale segreta con la console RDS AWS CLI, l'o l'API RDS. La nuova password è sempre lunga 28 caratteri e contiene almeno un carattere maiuscolo e minuscolo, un numero e una punteggiatura.

Console

Per ruotare il segreto della password dell'utente master utilizzando la console RDS, modifica il cluster database e seleziona Rotate secret immediately (Ruota il segreto immediatamente) in Settings (Impostazioni).



Per modificare un cluster database con la console RDS segui le istruzioni presenti in Modifica del cluster di database tramite la console, la CLI e l'API. È necessario scegliere Apply immediately (Applica immediatamente) nella pagina di conferma.

AWS CLI

Per ruotare una password utente principale segreta utilizzando il AWS CLI, usa il comando e specifica l'modify-db-clusteropzione. `--rotate-master-user-password` È necessario specificare l'opzione `--apply-immediately` quando si ruota la password master.

Questo esempio ruota il segreto della password dell'utente master.

Example

Per LinuxmacOS, oUnix:

```
aws rds modify-db-cluster \
```

```
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Per Windows:

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

API RDS

È possibile ruotare la password segreta di un utente principale utilizzando l'DBClusteroperazione Modifica e impostando il `RotateMasterUserPassword` parametro su`true`. È necessario impostare il parametro `ApplyImmediately` su `true` quando si ruota la password master.

Visualizzazione dei dettagli di un segreto per un cluster database

Puoi recuperare i tuoi segreti usando la console (https://console.aws.amazon.com/secretsmanager/) o il comando AWS CLI (get-secret-valueSecrets Manager).

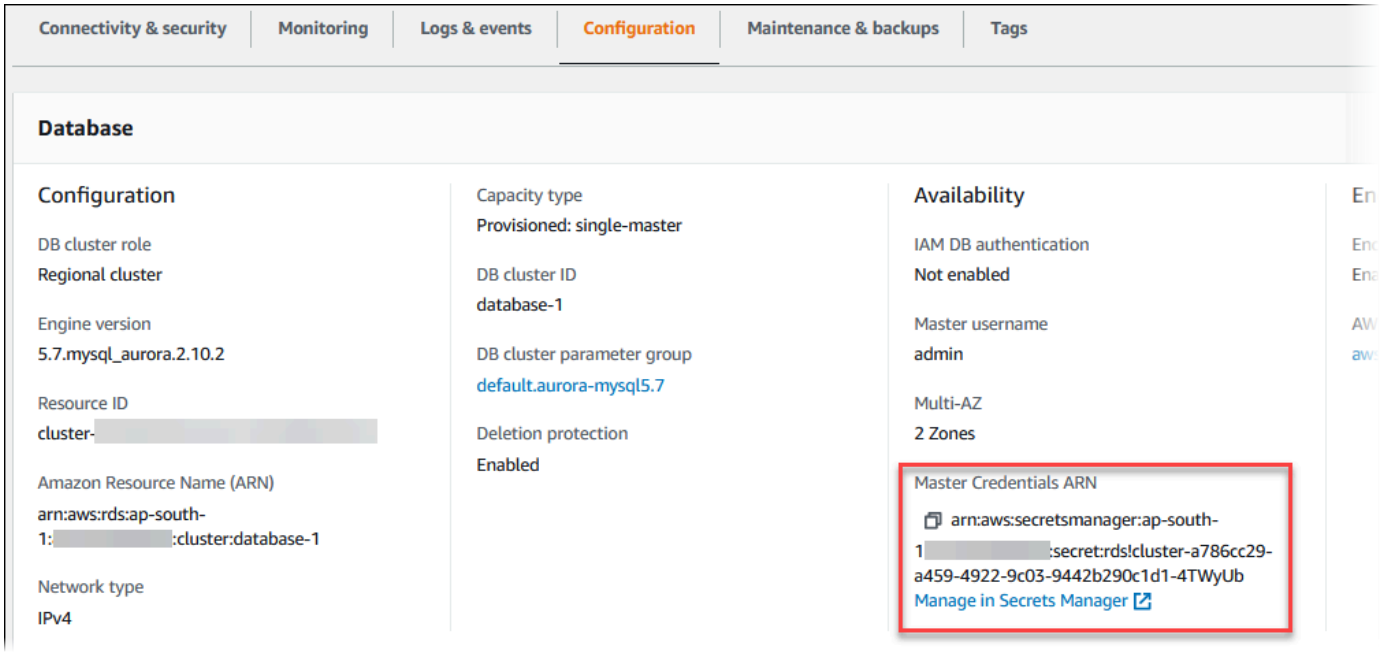
Puoi trovare l'Amazon Resource Name (ARN) di un segreto gestito da Aurora in Secrets Manager con la console RDS AWS CLI, o l'API RDS.

Console

Per visualizzare i dettagli di un segreto gestito da Aurora in Secrets Manager

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegli il nome del cluster database per visualizzarne i dettagli.
4. Scegli la scheda Configurazione.

   In Master Credentials ARN (ARN credenziali master), puoi visualizzare l'ARN del segreto.



Puoi selezionare il collegamento Manage in Secrets Manager (Gestisci in Secrets Manager) per visualizzare e gestire il segreto nella console di Secrets Manager.

AWS CLI

È possibile utilizzare il AWS CLI [describe-db-clusters](#)comando RDS per trovare le seguenti informazioni su un segreto gestito da in Secrets Manager:

- `SecretArn`: l'ARN del segreto
- `SecretStatus`: lo stato del segreto

  I valori possibili per lo stato sono:

  - `creating`: il segreto è in fase di creazione.
  - `active`: il segreto è disponibile per l'uso normale e la rotazione.
  - `rotating`: il segreto è in fase di rotazione.
  - `impaired`: il segreto può essere utilizzato per accedere alle credenziali del database, ma non può essere ruotato. Un segreto può avere questo stato se, ad esempio, le autorizzazioni vengono modificate in modo che RDS non può più accedere al segreto o alla chiave KMS del segreto.

    Quando un segreto ha questo stato, puoi correggere la condizione che lo ha causato. Se correggi la condizione che ha causato lo stato, lo stato rimane `impaired` fino alla rotazione

successiva. In alternativa, è possibile modificare il cluster database per disattivare la gestione automatica delle credenziali del database e quindi modificare nuovamente il cluster database per attivare la gestione automatica delle credenziali del database. Per modificare il cluster DB, utilizzare l'`--manage-master-user-password`opzione nel comando. modify-db-cluster

- `KmsKeyId`: l'ARN della chiave KMS utilizzata per crittografare il segreto

Specifica l'opzione `--db-cluster-identifier` per mostrare l'output per un cluster database specifico. Questo esempio mostra l'output di un segreto utilizzato da un cluster database.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

L'esempio seguente mostra l'output di un segreto:

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

Quando si dispone dell'ARN segreto, è possibile visualizzare i dettagli sul segreto utilizzando il comando get-secret-valueSecrets Manager CLI.

Questo esempio mostra i dettagli del segreto nell'output di esempio precedente.

Example

PerLinux, omacOS: Unix

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Per Windows:

```
aws secretsmanager get-secret-value ^
```

```
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

API RDS

È possibile visualizzare l'ARN, lo stato e la chiave KMS per un segreto gestito da RDS in Secrets Manager utilizzando l'operazione DBClusters Descrivi RDS e `DBClusterIdentifier` impostando il parametro su un identificatore di cluster DB. I dettagli del segreto sono inclusi nell'output.

Quando si dispone dell'ARN segreto, è possibile visualizzare i dettagli sul segreto utilizzando l'operazione GetSecretValueSecrets Manager.

Protezione dei dati in Amazon RDS

Il modello di responsabilità AWS condivisa modello di di si applica alla protezione dei dati in Amazon Relational Database Service. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS .

Ai fini della protezione dei dati, consigliamo di proteggere Account AWS le credenziali e configurare i singoli utenti con AWS IAM Identity Center or AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Usa SSL/TLS per comunicare con le risorse. AWS È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con. AWS CloudTrail Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.

- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Amazon RDS o altro Servizi AWS utilizzando la console, l'API o AWS SDKs. AWS CLI I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Argomenti

- Protezione dei dati tramite crittografia
- Riservatezza del traffico Internet

Protezione dei dati tramite crittografia

Puoi abilitare la crittografia per le risorse di database.

Argomenti

- Crittografia delle risorse Amazon Aurora
- AWS KMS key gestione
- Utilizzo SSL/TLS per crittografare una connessione a un'
- Rotazione del certificato SSL/TLS

Crittografia delle risorse Amazon Aurora

Amazon Aurora può crittografare i cluster database di Amazon Aurora. I dati che vengono crittografati quando sono inattivi includono lo storage sottostante per le cluster database, i backup automatici, le repliche di lettura e gli snapshot.

I istanze database Amazon Aurora crittografate utilizzano l'algoritmo di crittografia AES-256 standard del settore per crittografare i dati sul server che ospita i istanze database Amazon Aurora. Una volta crittografati i dati, Amazon Aurora gestisce l'autenticazione dell'accesso e la decrittografia dei dati

in modo trasparente con un impatto minimo sulle prestazioni. Non è quindi necessario modificare le applicazioni client di database per utilizzare la crittografia.

> **Note**
>
> Per i cluster di DB crittografati e non crittografati, i dati in transito tra le repliche di origine e quelle di lettura vengono crittografati, anche durante la replica tra regioni. AWS

Argomenti

- Panoramica della crittografia delle risorse Amazon Aurora
- Creazione di un cluster di database Amazon Aurora
- Determinare se la crittografia è attivata per un cluster database
- Disponibilità della crittografia Amazon Aurora
- Crittografia in transito
- Limiti relativi a istanze database crittografate Amazon Aurora

Panoramica della crittografia delle risorse Amazon Aurora

I cluster database Amazon Aurora crittografati offrono un livello aggiuntivo di sicurezza dei dati proteggendoli dagli accessi non autorizzati nello storage sottostante. Puoi utilizzare la crittografia Amazon Aurora per aumentare la protezione dei dati delle applicazioni che vengono distribuite nel cloud e per soddisfare i requisiti di conformità per la crittografia dei dati inattivi. Per un cluster database Amazon Aurora crittografato, vengono crittografati tutti i log, i backup e le snapshot. Per ulteriori informazioni sulla disponibilità e sui limiti della crittografia, consulta Disponibilità della crittografia Amazon Aurora e. Limiti relativi a istanze database crittografate Amazon Aurora

Amazon Aurora utilizza una AWS Key Management Service chiave per crittografare queste risorse. AWS KMS combina hardware e software sicuri e ad alta disponibilità per fornire un sistema di gestione delle chiavi scalabile per il cloud. È possibile utilizzare o creare chiavi gestite dal cliente. Chiave gestita da AWS

Quando crei un cluster di database crittografato, puoi scegliere una chiave gestita dal cliente o la Chiave gestita da AWS per Amazon Aurora per la crittografia del cluster di database. Se non specifichi l'identificatore di chiave per una chiave gestita dal cliente, Amazon Aurora lo utilizza per Chiave gestita da AWS il tuo nuovo cluster DB. Amazon Aurora crea una pagina per Amazon Aurora

Chiave gestita da AWS per il tuo account. AWS Il tuo AWS account ha un nome diverso Chiave gestita da AWS per Amazon Aurora per ogni AWS regione.

Per gestire le chiavi gestite dal cliente utilizzate per crittografare e decrittografare le risorse Amazon Aurora, usi il ().AWS Key Management ServiceAWS KMS

Utilizzando AWS KMS, puoi creare chiavi gestite dal cliente e definire le politiche per controllare l'uso di queste chiavi gestite dal cliente. AWS KMS supporta CloudTrail, in modo da poter controllare l'utilizzo delle chiavi KMS per verificare che le chiavi gestite dal cliente vengano utilizzate in modo appropriato. Puoi utilizzare le chiavi gestite dai clienti con Amazon Aurora e AWS servizi supportati come Amazon S3, Amazon EBS e Amazon Redshift. Per un elenco dei servizi integrati con AWS KMS, consulta Service Integration.AWS Alcune considerazioni sull'utilizzo delle chiavi KMS:

- Una volta creata un'istanza database crittografata, non potrai più modificare la chiave KMS utilizzata da quell'istanza database. Pertanto, assicurati di determinare i requisiti della chiave KMS prima di creare la tua istanza database crittografata.

  Se devi modificare la chiave di crittografia per il tuo cluster DB, crea un'istantanea manuale del cluster e abilita la crittografia durante la copia dello snapshot. Per ulteriori informazioni, consulta l'articolo di re:POST Knowledge.
- Se copi uno snapshot crittografata, puoi utilizzare una chiave KMS diversa per crittografare la snapshot di destinazione rispetto a quella utilizzata per crittografare la snapshot di origine.
- Non puoi condividere un'istantanea che è stata crittografata utilizzando l' Chiave gestita da AWS AWS account che ha condiviso l'istantanea.
- Ogni istanza database nel cluster database viene crittografata utilizzando la stessa chiave KMS del cluster di database.
- Puoi anche crittografare una replica di lettura di un cluster crittografato con Amazon Aurora.

> **Important**
>
> Amazon Aurora può perdere l'accesso alla chiave KMS per un cluster DB quando disabiliti la chiave KMS. In questi casi, il cluster di database crittografato entra nello stato `inaccessible-encryption-credentials-recoverable`. Il cluster DB rimane in questo stato per sette giorni, durante i quali l'istanza viene interrotta. Le chiamate API effettuate al cluster DB durante questo periodo potrebbero non avere esito positivo. Per ripristinare il cluster DB, abilita la chiave KMS e riavvia questo cluster DB. Abilita la

chiave KMS dall'API AWS Management Console AWS CLI, o RDS. Riavviare il cluster DB utilizzando il AWS CLI comando start-db-clustero. AWS Management Console
Lo `inaccessible-encryption-credentials-recoverable` stato si applica solo ai cluster DB che possono arrestarsi. Questo stato ripristinabile non è applicabile alle istanze che non possono essere interrotte, come i cluster con repliche di lettura interregionali. Per ulteriori informazioni, consulta the section called “Restrizioni”.
Se il cluster DB non viene ripristinato entro sette giorni, passa allo stato terminale. `inaccessible-encryption-credentials` In questo stato, il cluster DB non è più utilizzabile ed è possibile ripristinare il cluster DB solo da un backup. Ti consigliamo vivamente di attivare sempre i backup per i cluster DB crittografati per evitare la perdita di dati crittografati nei database.
Durante la creazione di un cluster DB, Aurora verifica se il principale chiamante ha accesso alla chiave KMS e genera una concessione dalla chiave KMS che utilizza per l'intera durata del cluster DB. La revoca dell'accesso del principale chiamante alla chiave KMS non influisce su un database in esecuzione. Quando si utilizzano le chiavi KMS in scenari che coinvolgono più account, ad esempio per copiare un'istantanea su un altro account, la chiave KMS deve essere condivisa con l'altro account. Se crei un cluster DB dalla snapshot senza specificare una chiave KMS diversa, il nuovo cluster utilizza la chiave KMS dell'account di origine.
La revoca dell'accesso alla chiave dopo aver creato il cluster DB non influisce sul cluster. Tuttavia, la disabilitazione della chiave influisce su tutti i cluster DB crittografati con quella chiave. Per evitare che ciò accada, specificate una chiave diversa durante l'operazione di copia dell'istantanea.

Per ulteriori informazioni sulle chiavi KMS, consulta AWS KMS keys nella Guida per sviluppatori di AWS Key Management Service e AWS KMS key gestione.

Creazione di un cluster di database Amazon Aurora

Per crittografare un nuovo cluster di database, scegliere Enable encryption (Abilita crittografia) nella console. Per ulteriori informazioni sulla creazione di un cluster database, consulta Creazione di un cluster database Amazon Aurora.

Se utilizzate il create-db-cluster AWS CLI comando per creare un cluster DB crittografato, impostate il parametro. `--storage-encrypted` Se utilizzate l'operazione Create DBCluster API, impostate il `StorageEncrypted` parametro su true.

Una volta creato un cluster di database crittografato, non potrai più modificare la chiave KMS utilizzata da quel cluster di database. Assicurati quindi di determinare i requisiti della chiave KMS prima di creare il cluster di database crittografato.

Se utilizzi il AWS CLI `create-db-cluster` comando per creare un cluster DB crittografato con una chiave gestita dal cliente, imposta il `--kms-key-id` parametro su qualsiasi identificatore di chiave per la chiave KMS. Se utilizzi la funzionalità `CreateDBInstance` dell'API Amazon RDS, imposta il parametro `KmsKeyId` su un qualsiasi identificatore chiave per la chiave KMS. Per utilizzare una chiave gestita dal cliente in un altro AWS account, specificare l'ARN della chiave o l'alias ARN.

Determinare se la crittografia è attivata per un cluster database

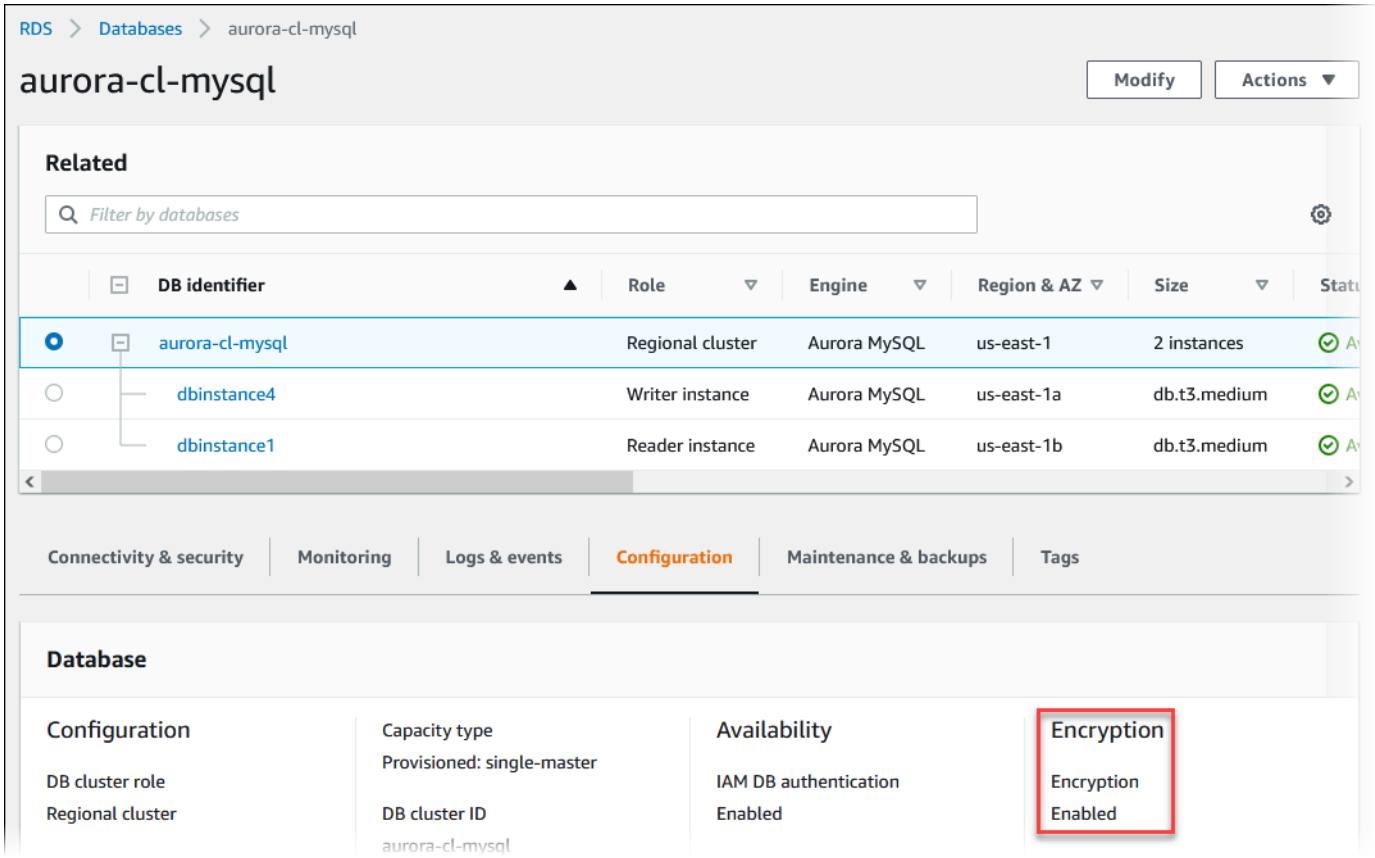
Puoi utilizzare l'API AWS Management Console AWS CLI, o RDS per determinare se la crittografia a riposo è attivata per un cluster DB.

Console

Per determinare se la crittografia a riposo è attivata per un cluster database

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database).
3. Scegliere il nome del cluster database da controllare per visualizzarne i dettagli.
4. Selezionare la casella Configurazione e controllare il valore Crittografia.

   Mostra Enabled (Abilitato) o Non abilitato.



AWS CLI

Per determinare se la crittografia a riposo è attivata per un cluster DB utilizzando il AWS CLI, chiama il describe-db-clusterscomando con la seguente opzione:

- `--db-cluster-identifier`: il nome del cluster di database.

Nell'esempio seguente viene utilizzata una query per restituire `TRUE` o `FALSE` per quanto riguarda la crittografia inattiva per il cluster database `mydb`.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

API RDS

Per determinare se la crittografia a riposo è attivata per un cluster DB utilizzando l'API Amazon RDS, chiama l'DBClustersoperazione Descrivi con il seguente parametro:

- `DBClusterIdentifier`: il nome del cluster di database.

Disponibilità della crittografia Amazon Aurora

La crittografia Amazon Aurora è attualmente disponibile per tutti i motori di database e i tipi di storage.

> **Note**
>
> La crittografia Amazon Aurora non è disponibile per la classe di istanza database db.t2.micro.

Crittografia in transito

Crittografia a livello fisico

Tutti i dati che fluiscono attraverso la Regioni AWS rete AWS globale vengono automaticamente crittografati a livello fisico prima di lasciare le strutture AWS protette. Tutto il traffico AZs intercorrente è crittografato. Ulteriori livelli di crittografia, inclusi quelli elencati in questa sezione, possono fornire protezioni aggiuntive.

Crittografia fornita dal peering Amazon VPC e dal peering transregionale Transit Gateway

Tutto il traffico tra regioni che utilizza il peering Amazon VPC e Transit Gateway viene automaticamente crittografato in massa quando esce da una regione. Un ulteriore livello di crittografia viene fornito automaticamente a livello fisico per tutto il traffico prima che lasci le strutture protette. AWS

Crittografia tra istanze

AWS fornisce una connettività sicura e privata tra istanze DB di tutti i tipi. Inoltre, alcuni tipi di istanza utilizzano le funzionalità di offload dell'hardware Nitro System sottostante per crittografare automaticamente il traffico in transito tra le istanze. Questa crittografia utilizza algoritmi AEAD (Authenticated Encryption with Associated Data), con crittografia a 256 bit. Non vi è alcun impatto sulle prestazioni della rete. Per supportare questa crittografia aggiuntiva del traffico in transito tra istanze, è necessario soddisfare i seguenti requisiti:

- Le istanze utilizzano i seguenti tipi di istanza:
  - Scopo generale: M6i, M6id, M6in, M6idn, M7g
  - Memoria ottimizzata: R6i, R6id, R6in, R6idn, R7g, X2idn, X2iEdn, X2iEzn
- Le Regione AWS istanze sono le stesse.

- Le istanze si trovano nello stesso VPC o VPCs peered e il traffico non passa attraverso un dispositivo o un servizio di rete virtuale, come un sistema di bilanciamento del carico o un gateway di transito.

Limiti relativi a istanze database crittografate Amazon Aurora

Esistono le seguenti limitazioni per le istanze database crittografate Amazon Aurora:

- Non puoi disattivare la crittografia di un cluster database crittografato.
- Non puoi creare uno snapshot crittografata per un cluster database non crittografato.
- Una snapshot di un cluster database crittografato deve essere crittografata utilizzando la stessa chiave KMS del cluster database.
- Non è possibile convertire un cluster database non crittografato in uno crittografato. Tuttavia, puoi ripristinare uno snapshot di un cluster database non crittografato in un cluster database Aurora crittografato. Per eseguire questa operazione, specifica una chiave KMS quando ripristini dalla snapshot non crittografata.
- Non è possibile creare una replica Aurora crittografata da un cluster database Aurora non crittografato. Non è possibile creare una replica Aurora non crittografata da un cluster database Aurora crittografato.
- Per copiare un'istantanea crittografata da una AWS regione all'altra, è necessario specificare la chiave KMS nella regione di destinazione. AWS Questo perché le chiavi KMS sono specifiche della AWS regione in cui vengono create.

  La snapshot di origine resta crittografata nel processo di copia. Amazon Aurora utilizza la crittografia envelope per proteggere i dati durante il processo di copia. Per ulteriori informazioni sulla crittografia envelope, consulta Crittografia envelope nella Guida per sviluppatori di AWS Key Management Service .
- Non è possibile decrittografare un cluster database crittografato. Tuttavia, puoi esportare i dati da un cluster database crittografato e importarli in un cluster database non crittografato.

AWS KMS key gestione

Amazon Aurora si integra automaticamente AWS Key Management Service con AWS KMS() per la gestione delle chiavi. Amazon Aurora utilizza la crittografia a busta. Per ulteriori informazioni sulla crittografia envelope, consulta Crittografia envelope nella Guida per sviluppatori di AWS Key Management Service .

- Per avere il pieno controllo su una chiave KMS, devi creare una chiave gestita dal cliente. Per ulteriori informazioni sulle chiavi gestite dal cliente, consulta Chiavi gestite dal cliente nella Guida per gli sviluppatori di AWS Key Management Service .
- Chiavi gestite da AWSsono chiavi KMS presenti nel tuo account che vengono create, gestite e utilizzate per tuo conto da un AWS servizio integrato con. AWS KMS Per impostazione predefinita, per la crittografia viene utilizzato RDS Chiave gestita da AWS (`aws/rds`). Non è possibile gestire, ruotare o eliminare l'RDS. Chiave gestita da AWS Per ulteriori informazioni Chiavi gestite da AWSin merito Chiavi gestite da AWS, consulta la Guida per gli AWS Key Management Service sviluppatori.

Per gestire le chiavi KMS utilizzate per i cluster di istanze crittografate Amazon Aurora, usa AWS Key Management Service il AWS KMS() nella console, AWS CLI l'o AWS KMS l'API. AWS KMS Puoi visualizzare i log di controllo di ogni operazione eseguita con una chiave gestita da AWS o dal cliente utilizzando AWS CloudTrail. Per ulteriori informazioni sulla rotazione delle chiavi, consulta Rotazione delle chiavi AWS KMS.

Autorizzazione dell'uso di una chiave gestita dal cliente

Quando Aurora utilizza una chiave gestita dal cliente nelle operazioni crittografiche, agisce per conto dell'utente che sta creando o modificando la risorsa Aurora.

Per creare una risorsa Aurora utilizzando una chiave gestita dal cliente, un utente deve disporre delle autorizzazioni per richiamare le seguenti operazioni sulla chiave gestita dal cliente:

- `kms:CreateGrant`
- `kms:DescribeKey`

Puoi specificare queste autorizzazioni necessarie in una policy chiave o in una policy IAM se la policy chiave lo consente.

> **Tip**
>
> Per seguire il principio del privilegio minimo, non consentire l'accesso completo a `kms:CreateGrant`. Utilizza invece la chiave kms: ViaService condition per consentire all'utente di creare concessioni sulla chiave KMS solo quando la concessione viene creata per conto dell'utente da un servizio. AWS

Esistono diversi modi per rendere la policy IAM più efficace. Ad esempio, se desideri consentire l'utilizzo della chiave gestita dal cliente solo per le richieste che hanno origine in Aurora, utilizza kms:ViaService la chiave di condizione con `rds.<region>.amazonaws.com` il valore. Puoi inoltre usare le chiavi o i valori nel Contesto di crittografia di Amazon RDS come condizione per utilizzare la chiave gestita dal cliente per la crittografia.

Per ulteriori informazioni, consulta Autorizzazione per gli utenti in altri account di utilizzare una chiave KMS nella Guida per gli sviluppatori di AWS Key Management Service e Policy delle chiavi in AWS KMS.

Contesto di crittografia di Amazon RDS

Quando Aurora utilizza la tua chiave KMS o quando Amazon EBS utilizza la chiave KMS per conto di Aurora, il servizio specifica un contesto di crittografia. Il contesto di crittografia è costituito da dati autenticati aggiuntivi (AAD) utilizzati per garantire l'integrità dei dati. AWS KMS Quando viene specificato un contesto di crittografia per un'operazione di crittografia, il servizio deve specificare lo stesso contesto di crittografia per l'operazione di decrittografia. In caso contrario, la decrittografia ha esito negativo. Il contesto di crittografia viene scritto nei log AWS CloudTrail per aiutarti a comprendere perché è stata utilizzata una determinata chiave KMS. CloudTrail I registri potrebbero contenere molte voci che descrivono l'uso di una chiave KMS, ma il contesto di crittografia in ogni voce di registro può aiutarti a determinare il motivo di quel particolare utilizzo.

Come minimo, Aurora utilizza sempre l'ID del cluster dell'istanza per il contesto di crittografia, come nel seguente esempio in formato JSON:

```
{ "aws:rds:dbc-id": "cluster-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

Questo contesto di crittografia può aiutarti a identificare il cluster di DB per il quale è stata utilizzata la tua chiave KMS.

Quando la chiave KMS viene utilizzata per uno specifico cluster di DB e un volume Amazon EBS specifico, per il contesto di crittografia vengono utilizzati sia l'ID del cluster di DB che l'ID del volume Amazon EBS, come nel seguente esempio in formato JSON:

```
{
  "aws:rds:dbc-id": "cluster-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

Utilizzo SSL/TLS per crittografare una connessione a un'

Puoi utilizzare Secure Socket Layer (SSL) o Transport Layer Security (TLS) dall'applicazione per crittografare una connessione a un cluster di database che esegue Aurora MySQL o Aurora PostgreSQL.

Facoltativamente, la SSL/TLS connessione può eseguire la verifica dell'identità del server convalidando il certificato del server installato nel database. Per richiedere la verifica dell'identità del server, esegui questa procedura generale:

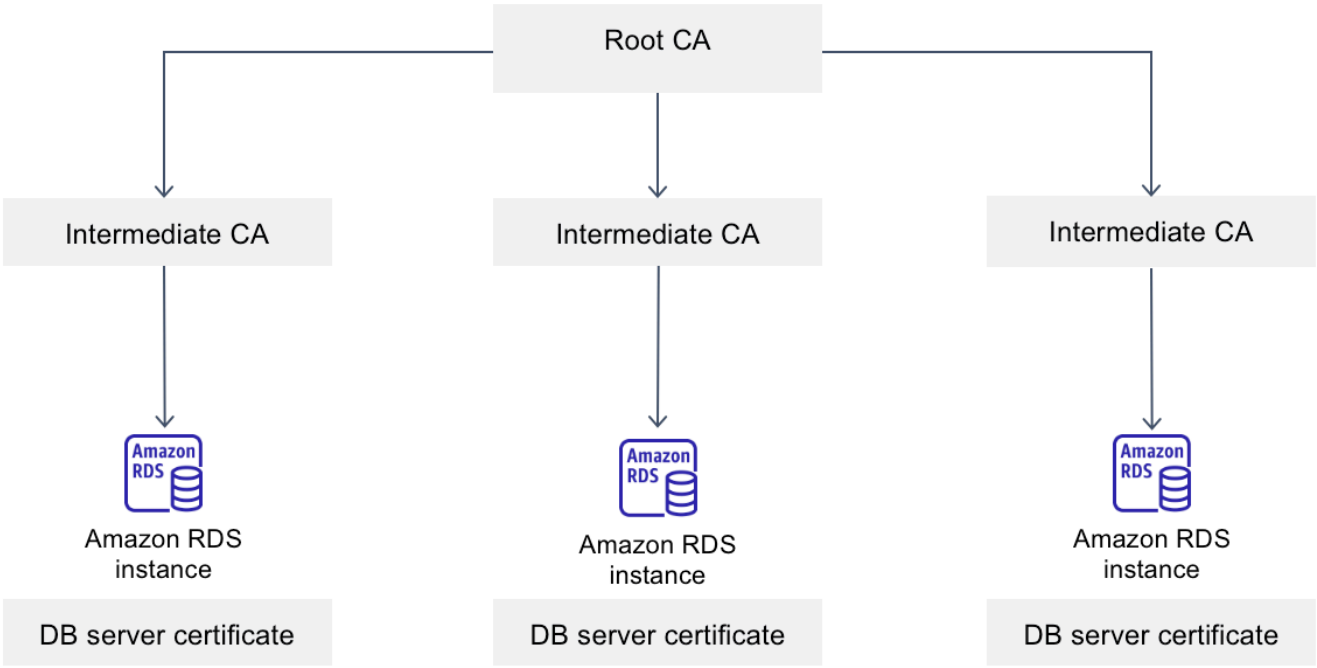
1. Scegli l'autorità di certificazione (CA) che firma il certificato del server di database per il database. Per ulteriori informazioni sulle autorità di certificazione, vedere. Autorità di certificazione
2. Scarica un bundle di certificati da utilizzare quando ti connetti al database. Pacchetti di certificati di Regione AWS.

   > **Note**
   >
   > Tutti i certificati sono disponibili solo per il download tramite connessioni SSL/TLS.

3. Connettiti al database utilizzando il processo del tuo motore DB per l'implementazione delle SSL/TLS connessioni. Ogni motore DB ha il proprio processo di implementazione SSL/TLS. To learn how to implement SSL/TLS per il tuo database, segui il link che corrisponde al tuo motore DB:
   - Sicurezza con Amazon Aurora MySQL
   - Sicurezza con Amazon Aurora PostgreSQL

Autorità di certificazione

L'autorità di certificazione (CA) è il certificato che identifica la CA root della catena di certificati. La CA firma il certificato del server di database, che è il certificato del server installato su ogni istanza database. Il certificato del server di database identifica l'istanza database come server attendibile.



Amazon RDS fornisce quanto segue CAs per firmare il certificato del server DB per un database.

Autorità di certificazione (CA)	Descrizione	Common name (CN) (Nome comune)
rds-ca-rsa2048-g1	Utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 2048 e algoritmo di firma nella maggior parte dei casi. SHA256 Regioni AWS Nel AWS GovCloud (US) Regions, questa CA utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 2048 e algoritmo di firma. SHA384 supporta la rotazione automatica dei certificati del server.	Amazon RDS *region-id entifier* RSA2 048 G1
rds-ca-rsa4096-g1	Utilizza un'autorità di certificazione con algoritmo a chiave privata RSA 4096 e	Amazon RDS *region-id*

Autorità di certificazione (CA)	Descrizione	Common name (CN) (Nome comune)
	algoritmo di firma. SHA384 supporta la rotazione automatica dei certificati del server.	*entifier* RSA4096 G1
rds-ca-ecc384-g1	Utilizza un'autorità di certificazione con algoritmo a chiave privata ECC 384 e algoritmo di firma. SHA384 supporta la rotazione automatica dei certificati del server.	Amazon RDS G1 *region-identifier* ECC384

> **Note**
>
> Se utilizzi il AWS CLI, puoi vedere le validità delle autorità di certificazione sopra elencate utilizzando describe-certificates.

Questi certificati CA sono inclusi nel bundle di certificati regionali e globali. Quando si utilizza la CA rds-ca-rsa 2048-g1, rds-ca-rsa 4096-g1 o rds-ca-ecc 384-g1 con un database, RDS gestisce il certificato del server DB sul database. RDS esegue automaticamente la rotazione del certificato del server di database prima della scadenza.

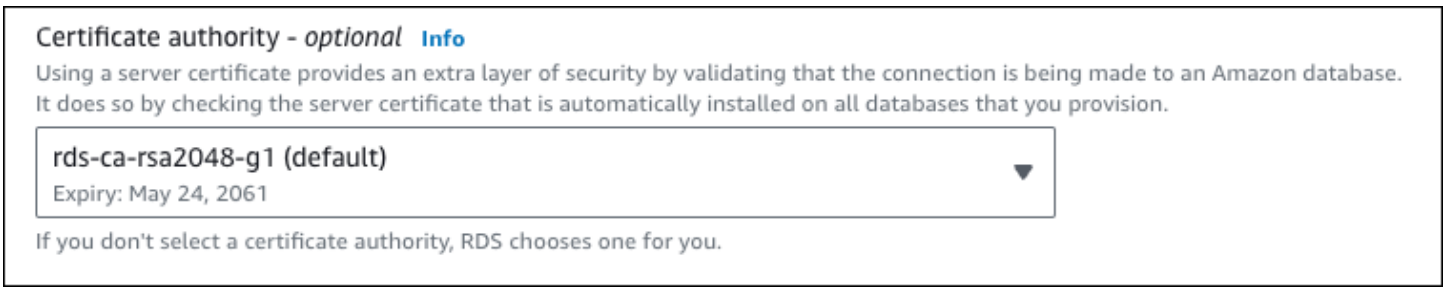
Impostazione della CA per il database

Puoi impostare la CA per un database quando esegui le seguenti attività:

- Crea un cluster Aurora DB: puoi impostare la CA per un'istanza DB in un cluster Aurora quando crei la prima istanza DB nel cluster DB utilizzando l' AWS CLI API o RDS. Attualmente, non puoi impostare la CA per istanze database in un cluster database durante la creazione del cluster database usando la console RDS. Per istruzioni, consulta. Creazione di un cluster database Amazon Aurora
- Modifica di un'istanza database: puoi impostare la CA per un'istanza database in un cluster database modificandola. Per istruzioni, vedereModifica di un'istanza database in un cluster database.

> **Note**
>
> La CA predefinita è impostata su rds-ca-rsa 2048-g1. È possibile sovrascrivere la CA predefinita per il proprio Account AWS utilizzando il comando modify-certificates.

Le opzioni disponibili CAs dipendono dal motore DB e dalla versione del motore DB. Quando si utilizza il AWS Management Console, è possibile scegliere la CA utilizzando l'impostazione dell'autorità di certificazione, come mostrato nell'immagine seguente.



La console mostra solo CAs le versioni disponibili per il motore DB e la versione del motore DB. Se si utilizza il AWS CLI, è possibile impostare la CA per un'istanza DB utilizzando il modify-db-instancecomando create-db-instanceor.

Se utilizzi il AWS CLI, puoi vedere quello disponibile CAs per il tuo account utilizzando il comando describe-certificates. Questo comando mostra nell'output anche la data di scadenza per ogni CA in `ValidTill`. Puoi trovare CAs quelli disponibili per uno specifico motore DB e una versione del motore DB utilizzando il comando. describe-db-engine-versions

L'esempio seguente mostra la CAs versione del motore DB RDS per PostgreSQL disponibile per la versione predefinita.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

L'output è simile a quello riportato di seguito. I disponibili sono elencati in. CAs `SupportedCACertificateIdentifiers` L'output mostra anche se la versione del motore di database supporta la rotazione del certificato senza riavvio in `SupportsCertificateRotationWithoutRestart`.

```
{
    "DBEngineVersions": [
        {
```

```
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
            "SupportedCACertificateIdentifiers": [
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
}
```

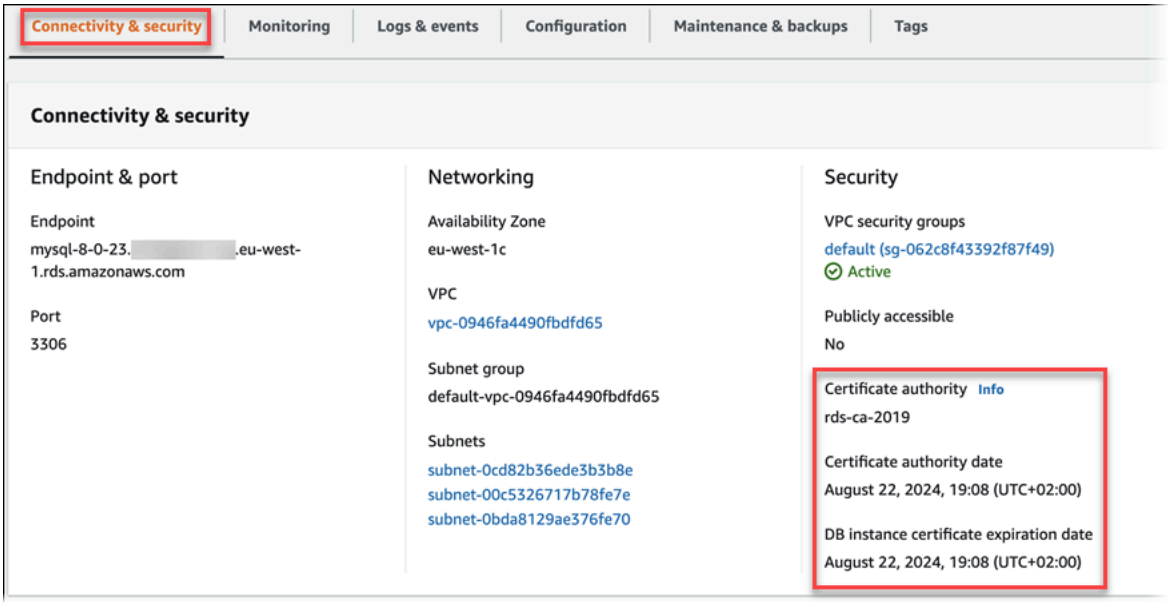
Validità dei certificati del server di database

La validità del certificato del server di database dipende dal motore di database e dalla versione del motore di database. Se la versione del motore di database supporta la rotazione del certificato senza riavvio, la validità del certificato del server di database è di 1 anno. In caso contrario, la validità è di 3 anni.

Per ulteriori informazioni sulla rotazione dei certificati del server DB, vedereRotazione automatica dei certificati del server.

Visualizzazione della CA per l'istanza DB

È possibile visualizzare i dettagli sulla CA di un database visualizzando la scheda Connettività e sicurezza nella console, come nell'immagine seguente.



Se si utilizza il AWS CLI, è possibile visualizzare i dettagli sulla CA per un'istanza DB utilizzando il describe-db-instancescomando.

Scarica i pacchetti di certificati per

Quando ti connetti al database con SSL o TLS, l'istanza del database richiede un certificato di attendibilità di Amazon RDS. Seleziona il link appropriato nella tabella seguente per scaricare il pacchetto corrispondente al Regione AWS luogo in cui ospiti il database.

Pacchetti di certificati di Regione AWS

I pacchetti di certificati per tutte le regioni Regioni AWS e GovCloud (Stati Uniti) contengono i seguenti certificati CA root:

- `rds-ca-rsa2048-g1`
- `rds-ca-rsa4096-g1`
- `rds-ca-ecc384-g1`

I `rds-ca-ecc384-g1` certificati `rds-ca-rsa4096-g1` and non sono disponibili nelle seguenti regioni:

- Asia Pacifico (Mumbai)
- Asia Pacifico (Melbourne)
- Canada occidentale (Calgary)

- Europa (Zurigo)
- Europa (Spagna)
- Israele (Tel Aviv)

L'application trust store deve solo registrare il certificato CA principale.

> **Note**
>
> Il proxy e Aurora Serverless v1 l'uso di Amazon RDS i certificati di AWS Certificate Manager (ACM). Se utilizzi RDS Proxy, non è necessario scaricare certificati Amazon RDS o aggiornare applicazioni che utilizzano connessioni proxy RDS. Per ulteriori informazioni, consulta Utilizzo diTLS/SSLcon Proxy RDS.
> Se utilizziAurora Serverless v1, non è necessario scaricare i certificati Amazon RDS. Per ulteriori informazioni, consulta Usare TLS/SSL con Aurora Serverless v1.

Per scaricare un pacchetto di certificati per un Regione AWS, seleziona il link Regione AWS che ospita il database nella tabella seguente.

AWS Region	Bundle di certificati (PEM)	Pacchetto di certificati () PKCS7
Qualsiasi pubblicità Regione AWS	global-bundle.pem	global-bundle.p7b
Stati Uniti orientali (Virginia settentrionale)	us-east-1-bundle.pem	us-east-1-bundle.p7b
US East (Ohio)	us-east-2-bundle.pem	us-east-2-bundle.p7b
US West (N. California)	us-west-1-bundle.pem	us-west-1-bundle.p7b
US West (Oregon)	us-west-2-bundle.pem	us-west-2-bundle.p7b
Africa (Cape Town)	af-south-1-bundle.pem	af-sud-1-bundle.p7b
Asia Pacific (Hong Kong)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b

AWS Region	Bundle di certificati (PEM)	Pacchetto di certificati () PKCS7
Asia Pacific (Hyderabad)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b
Asia Pacifico (Giacarta)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
Asia Pacifico (Malesia)	ap-southeast-5-bundle.pem	ap-southeast-5-bundle.p7b
Asia Pacifico (Melbourne)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
Asia Pacifico (Mumbai)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
Asia Pacific (Osaka)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b
Asia Pacifico (Tailandia)	ap-southeast-7-bundle.pem	ap-southeast-7-bundle.p7b
Asia Pacifico (Tokyo)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
Asia Pacific (Seoul)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
Asia Pacific (Singapore)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
Asia Pacific (Sydney)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b
Canada (Central)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
Canada occidentale (Calgary)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
Europa (Francoforte)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
Europe (Ireland)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b
Europe (London)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
Europe (Milan)	eu-south-1-bundle.pem	eu-sud-1-bundle.p7b
Europe (Paris)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b
Europa (Spagna)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
Europa (Stoccolma)	eu-nord-1-bundle.pem	eu-nord-1-bundle.p7b

AWS Region	Bundle di certificati (PEM)	Pacchetto di certificati () PKCS7
Europa (Zurigo)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
Israele (Tel Aviv)	il-central-1-bundle.pem	il-central-1-bundle.p7b
Messico (centrale)	mx-central-1-bundle.pem	mx-central-1-bundle.p7b
Middle East (Bahrain)	me-sud-1-bundle.pem	me-sud-1-bundle.p7b
Medio Oriente (Emirati Arabi Uniti)	me-central-1-bundle.pem	me-central-1-bundle.p7b
Sud America (San Paolo)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b
Qualsiasi s AWS GovCloud (US) Region	global-bundle.pem	global-bundle.p7b
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1 bundle.pem	us-gov-east-1 pacchetto.p7b
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1 bundle.pem	us-gov-west-1 pacchetto.p7b

Visualizzazione del contenuto del certificato CA

Per verificare il contenuto del bundle di certificati CA, utilizza il comando seguente:

```
keytool -printcert -v -file global-bundle.pem
```

Rotazione del certificato SSL/TLS

I certificati di Amazon RDS Certificate Authority rds-ca-2019 sono scaduti ad agosto 2024. Se utilizzi o prevedi di utilizzare Secure Sockets Layer (SSL) o Transport Layer Security (TLS) con verifica del certificato per connetterti alle tue istanze DB RDS o ai 384-g1. rds-ca-rsa rds-ca-rsa rds-ca-ecc Se attualmente non utilizzi SSL/TLS with certificate verification, you might still have an expired CA certificate and must update them to a new CA certificate if you plan to use SSL/TLS la verifica del certificato per connetterti ai tuoi database RDS.

Amazon RDS fornisce nuovi certificati CA come best practice di AWS sicurezza. Per informazioni sui nuovi certificati e sulle AWS regioni supportate, consultaUtilizzo SSL/TLS per crittografare una connessione a un'.

Per aggiornare il certificato CA per il tuo database, utilizza i seguenti metodi:

- Aggiornamento del certificato CA modificando l'istanza il cluster di database
- Aggiornamento del certificato CA mediante l'applicazione di manutenzione

Prima di aggiornare le istanze DB per utilizzare il nuovo certificato CA, assicurati di aggiornare i client o le applicazioni che si connettono ai database RDS.

Considerazioni sulla rotazione dei certificati

Considerate le seguenti situazioni prima di ruotare il certificato:

- Il proxy e Aurora Serverless v1 l'uso di Amazon RDS i certificati di AWS Certificate Manager (ACM). Se utilizzi il proxy RDS, quando ruoti il certificato SSL/TLS, non devi aggiornare le applicazioni che utilizzano connessioni proxy RDS. Per ulteriori informazioni, consulta Utilizzo diTLS/SSLcon Proxy RDS.
- Se utilizziAurora Serverless v1, non è necessario scaricare i certificati Amazon RDS. Per ulteriori informazioni, consulta Usare TLS/SSL con Aurora Serverless v1.
- Se utilizzi un'applicazione Go versione 1.15 con un'istanza DB creato o aggiornato al certificato rds-ca-2019 prima del 28 luglio 2020, devi aggiornare nuovamente il certificato.

  Utilizzate il `modify-db-instance` comando di certificato CA. `modify-db-cluster` Puoi trovare CAs quelli disponibili per uno specifico motore DB e una versione del motore DB utilizzando il `describe-db-engine-versions` comando.

  Se hai creato il database o aggiornato il relativo certificato dopo il 28 luglio 2020, non è richiesta alcuna azione. Per ulteriori informazioni, consulta il GitHub numero #39568 di Go.

Aggiornamento del certificato CA modificando l'istanza il cluster di database

L'esempio seguente aggiorna il certificato CA da rds-ca-2019 a 2048-g1. rds-ca-rsa Puoi scegliere un certificato diverso. Per ulteriori informazioni, consultaAutorità di certificazione.

Aggiorna l'application trust store per ridurre i tempi di inattività associati all'aggiornamento del certificato CA. Per ulteriori informazioni sui riavvii associati alla rotazione dei certificati CA, consultaRotazione automatica dei certificati del server.

1. Scarica il nuovo certificato SSL/TLS come descritto in. Utilizzo SSL/TLS per crittografare una connessione a un'
2. Aggiornare le applicazioni per utilizzare il nuovo certificato SSL/TLS.

   I metodi per aggiornare le applicazioni per i nuovi SSL/TLS certificates depend on your specific applications. Work with your application developers to update the SSL/TLS certificati per le tue applicazioni.

   Per maggiori informazioni sulla verifica delle connessioni SSL/TLS e sull'aggiornamento delle applicazioni per ciascun motore DB, consulta i seguenti argomenti:

   - Aggiornamento delle applicazioni per la connessione ai cluster database Aurora MySQL utilizzando nuovi certificati TLS
   - Aggiornamento delle applicazioni per la connessione ai cluster DB Aurora PostgreSQL utilizzando nuovi certificati SSL/TLS

   Per uno script di esempio che aggiorna un trust store per un sistema operativo Linux, vediScript di esempio per l'importazione di certificati nel tuo archivio di trust.

   > **Note**
   >
   > Il bundle di certificati contiene certificati per la vecchia e la nuova CA, pertanto puoi aggiornare l'applicazione in modo sicuro e mantenere la connettività durante il periodo di transizione. Se si utilizza il AWS Database Migration Service per migrare un database verso un', si consiglia di utilizzare il pacchetto di certificati per garantire la connettività durante la migrazione.

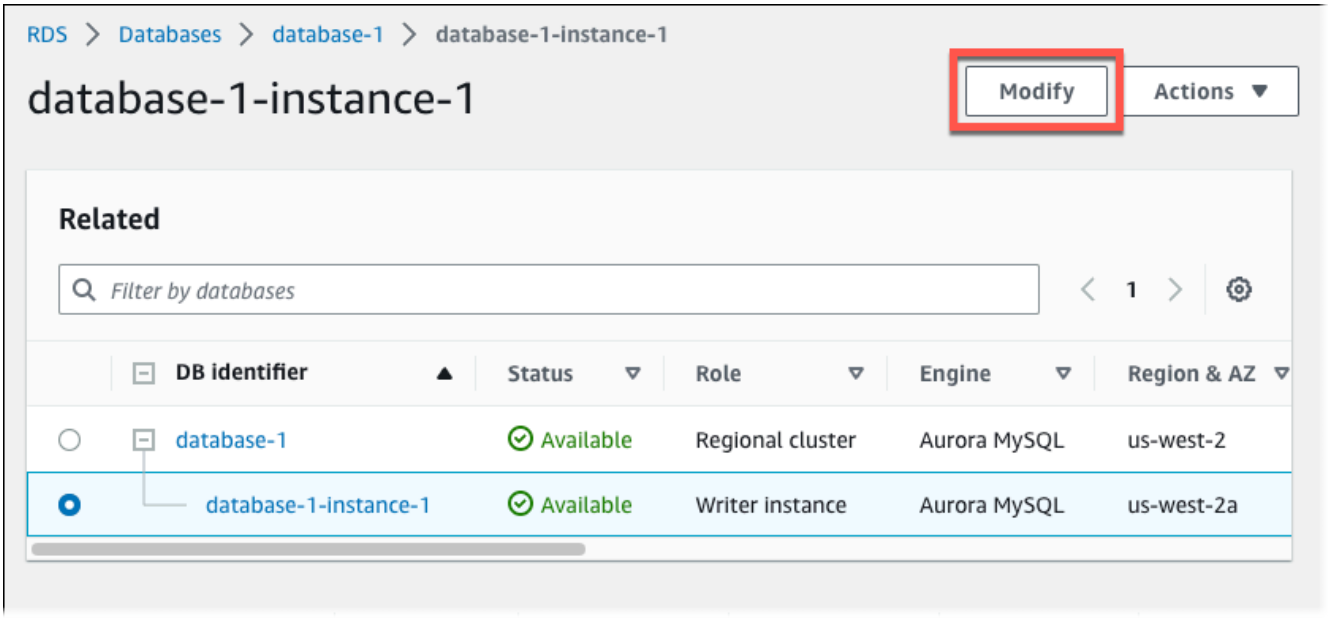
3. Modifica l'istanza DB per cambiare la CA da rds-ca-2019 a 2048-g1. rds-ca-rsa Per verificare se il database richiede un riavvio per aggiornare i certificati CA, utilizza il comando e seleziona il flag. describe-db-engine-versions`SupportsCertificateRotationWithoutRestart`

> **Note**
>
> Riavvia il cluster Babelfish dopo averlo modificato per aggiornare il certificato CA.

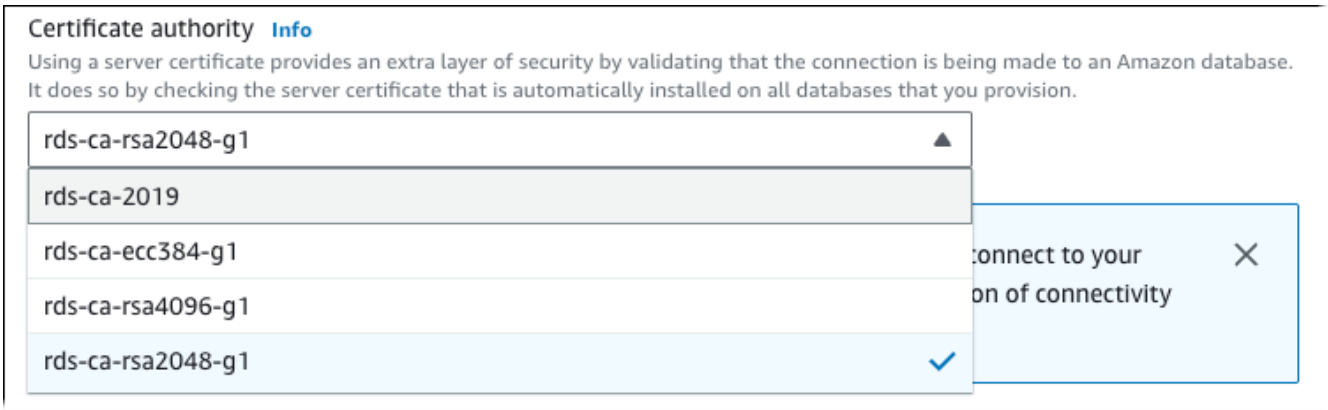
> **Important**
>
> Se si verificano problemi di connettività dopo la scadenza del certificato, utilizzare l'opzione Applica immediatamente specificando Apply immediately (Applica immediatamente) nella console o specificando l'opzione `--apply-immediately` mediante AWS CLI. Per impostazione predefinita, questa operazione è pianificata per l'esecuzione durante la prossima finestra di manutenzione.
> Per impostare un'override per la CA dell' del cluster diversa dalla CA RDS predefinita, utilizza il comando CLI modify-certificates.

Console

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Database, quindi scegli l'istanza DB che desideri modificare.
3. Scegli Modifica.



4. Nella sezione Connettività, scegli rds-ca-rsa2048-g1.



5. Scegliere Continue (Continua) e controllare il riepilogo delle modifiche.
6. Per applicare immediatamente le modifiche, scegliere Apply immediately (Applica immediatamente).
7. Nella pagina di conferma esaminare le modifiche. Se sono corrette, scegli Modifica istanza DB o Modifica per salvare le modifiche.

> **Important**
>
> Quando si pianifica questa operazione, accertarsi di aver aggiornato in anticipo l'archivio di trust lato client.

Oppure scegliere Back (Indietro) per cambiare le modifiche o Cancel (Annulla) per annullare le modifiche.

AWS CLI

Per utilizzare il AWS CLI per modificare la CA da rds-ca-2019 a rds-ca-rsa2048-g1 per un'istanza DB o un cluster DB . modify-db-instancemodify-db-cluster `--ca-certificate-identifier`

Utilizzate il `--apply-immediately` parametro per applicare immediatamente l'aggiornamento. Per impostazione predefinita, questa operazione è pianificata per l'esecuzione durante la prossima finestra di manutenzione.

> **Important**
>
> Quando si pianifica questa operazione, accertarsi di aver aggiornato in anticipo l'archivio di trust lato client.

Example

L'esempio seguente esegue la modifica `mydbinstance` impostando il certificato CA `surds-ca-rsa2048-g1`.

Per LinuxmacOS, oUnix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Per Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> Se l'istanza richiede il riavvio, puoi utilizzare il comando modify-db-instanceCLI e specificare `--no-certificate-rotation-restart` l'opzione.

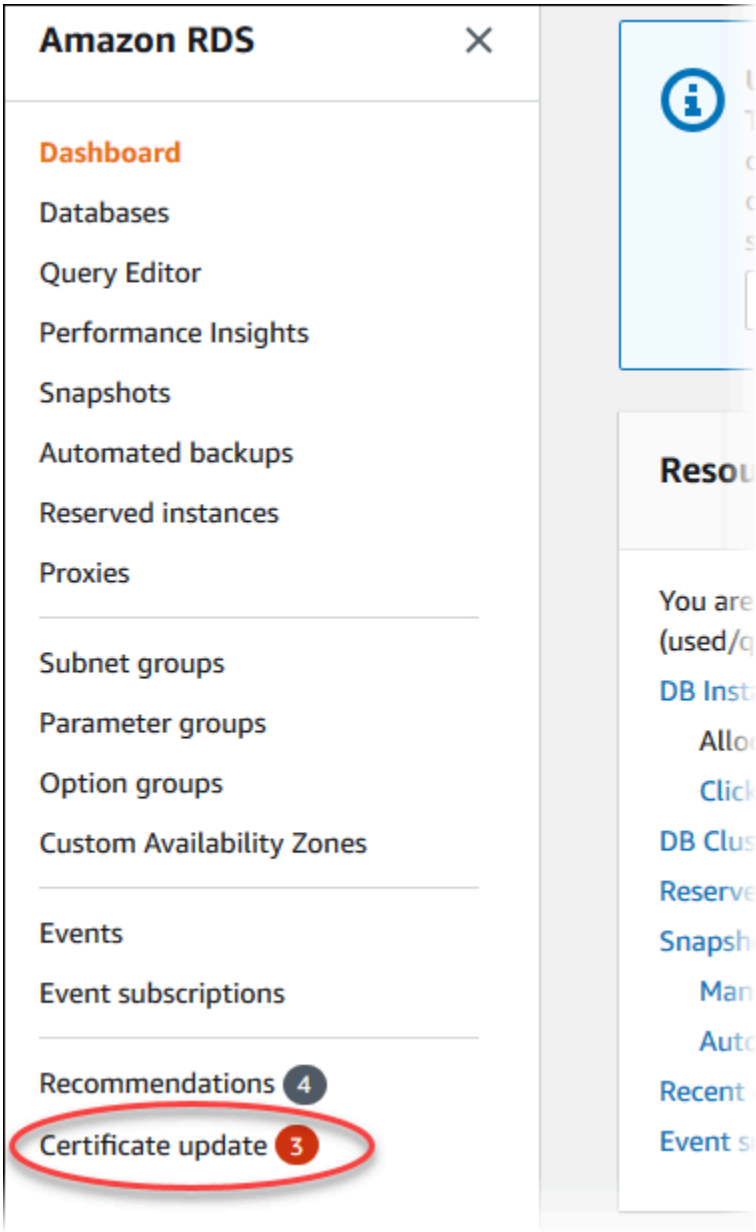
Aggiornamento del certificato CA mediante l'applicazione di manutenzione

Esegui i passaggi seguenti per aggiornare il certificato CA applicando la manutenzione.

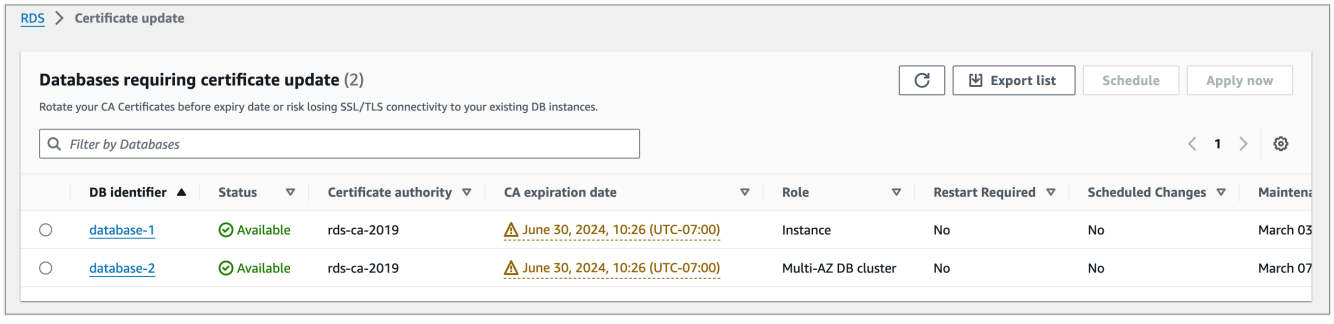
Console

Per aggiornare il certificato CA applicando la manutenzione

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegli Aggiornamento del certificato.



Viene visualizzata la pagina Database con aggiornamento certificati richiesto.



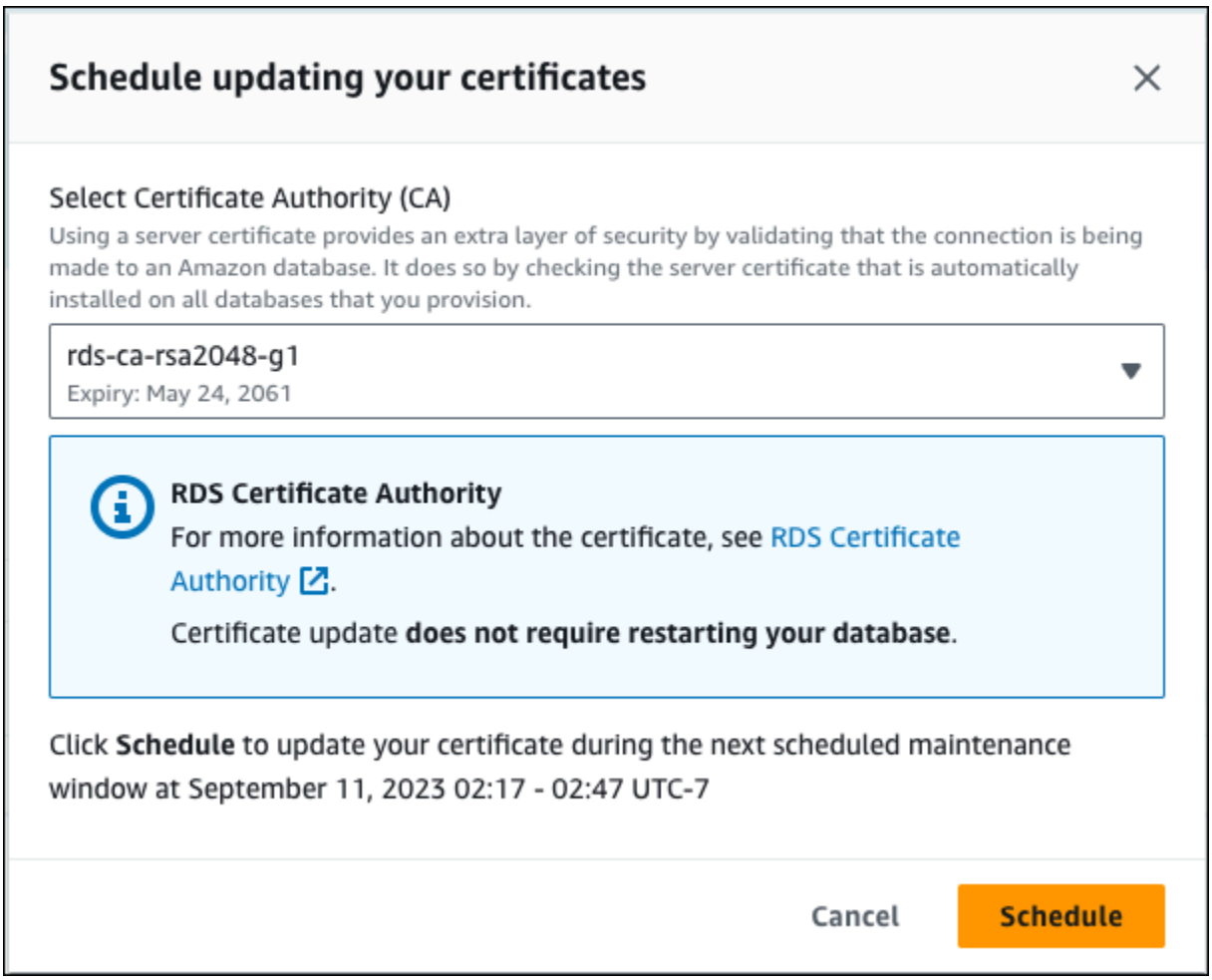
> **Note**
>
> Questa pagina mostra solo le istanze del database corrente. Regione AWS Se disponi di database in più di uno Regione AWS, controlla questa pagina in ciascuno di essi Regione AWS per vedere tutte le istanze DB con vecchi certificati SSL/TLS.

3. Scegli l'istanza DB che desideri aggiornare.

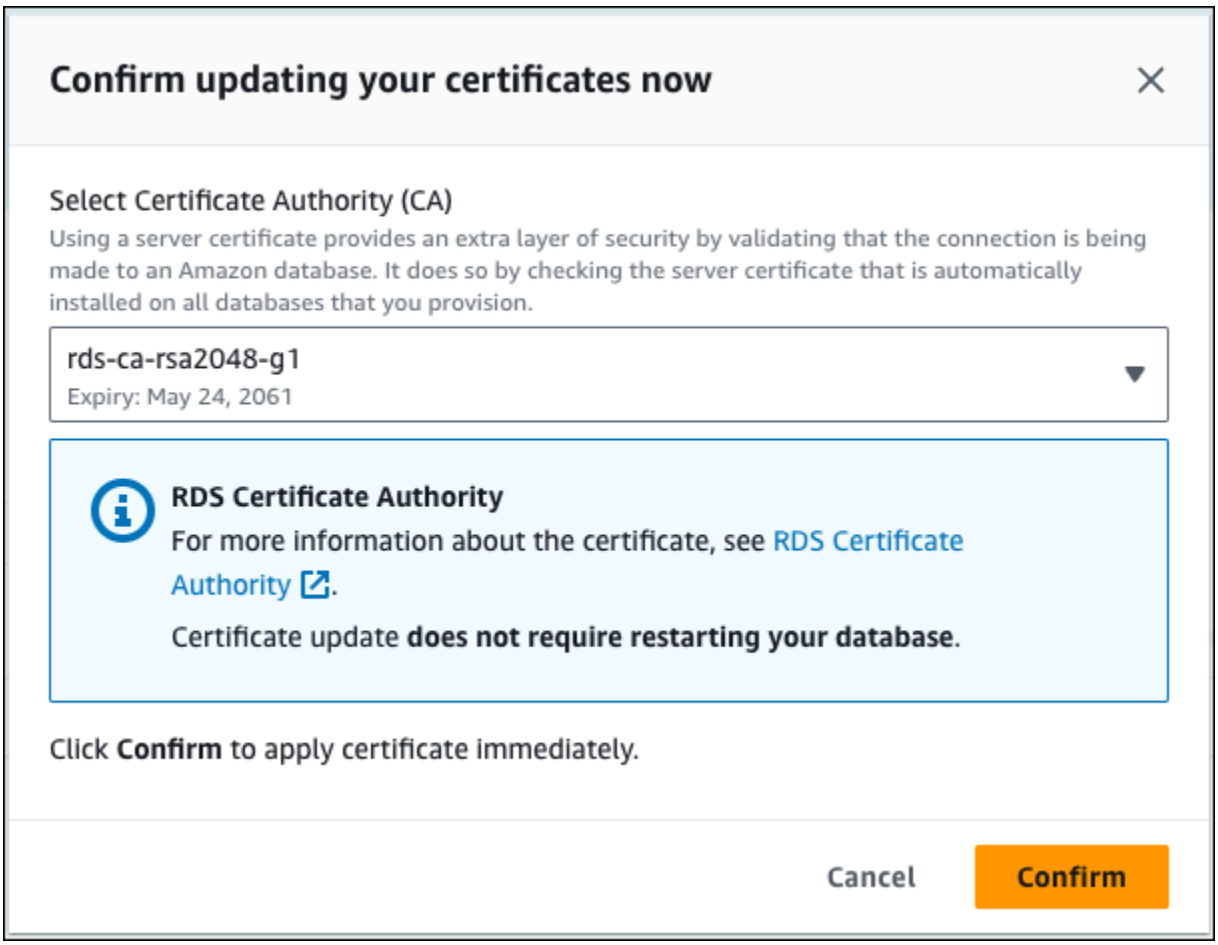
   È possibile pianificare la rotazione dei certificati per la finestra di manutenzione successiva scegliendo Pianifica. Applica immediatamente la rotazione scegliendo Applica ora.

   > **Important**
   >
   > Se si verificano problemi di connettività dopo la scadenza del certificato, utilizza l'opzione Applica ora.

4. a. Se scegli Pianifica, ti viene richiesto di confermare la rotazione dei certificati CA. Nella richiesta viene indicato anche la finestra pianificata per l'aggiornamento.



b. Se scegli Applica ora, ti viene richiesto di confermare la rotazione dei certificati CA.



> **Important**
>
> Prima di pianificare la rotazione del certificato CA nel database, aggiornare tutte le applicazioni client che utilizzano SSL/TLS e il certificato server per connettersi. Questi aggiornamenti sono specifici per il motore DB. Dopo avere aggiornato queste applicazioni client, è possibile confermare la rotazione del certificato CA.

Per continuare, scegliere la casella di controllo e quindi scegliere Confirm (Conferma).

5. Ripeti i passaggi 3 e 4 per ogni istanza DB che desideri aggiornare.

Rotazione automatica dei certificati del server

Se la CA principale supporta la rotazione automatica dei certificati del server, RDS gestisce automaticamente la rotazione del certificato del server DB. Per questa rotazione automatica, RDS

utilizza la stessa CA root e pertanto non è necessario scaricare un nuovo bundle CA. VediAutorità di certificazione.

La rotazione e la validità del certificato del server di database dipendono dal motore di database:

- Se il motore di database supporta la rotazione senza riavvio, RDS esegue automaticamente la rotazione del certificato del server di database senza richiedere alcuna azione da parte dell'utente. RDS tenta di eseguire la rotazione del certificato del server di database nella finestra di manutenzione preferita in corrispondenza della semivita del certificato del server di database. Il nuovo certificato del server di database è valido per 12 mesi.
- Se il motore di database in uso non supporta la rotazione senza riavvio, RDS ti avvisa di un evento di manutenzione almeno 6 mesi prima della scadenza del certificato del server di database. Il nuovo certificato del server di database è valido per 36 mesi.

Usa il describe-db-engine-versionscomando e controlla il `SupportsCertificateRotationWithoutRestart` flag per identificare se la versione del motore DB supporta la rotazione del certificato senza riavvio. Per ulteriori informazioni, consulta Impostazione della CA per il database.

Script di esempio per l'importazione di certificati nel tuo archivio di trust

Di seguito sono riportati script di shell di esempio che importano il bundle di certificati in un archivio di trust.

Ogni script di shell di esempio utilizza keytool, che fa parte del Java Development Kit (JDK). Per informazioni sull'installazione di JDK, consulta la Guida di installazione di JDK.

Linux

Il seguente script è uno script di esempio shell che importa il bundle di certificati in un archivio di trust su un sistema operativo Linux.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"> 
 ${mydir}/global-bundle.pem
```

```
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

Il seguente script è uno script di shell di esempio che importa il bundle di certificati in un archivio di trust su un sistema operativo Linux.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
```

```
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

Per ulteriori informazioni sulle best practice sull'uso di SSL con Amazon RDS, consulta Best practice per connessioni SSL di successo ad Amazon RDS for Oracle.

Riservatezza del traffico Internet

Le connessioni sono protette sia tra Amazon Aurora e le applicazioni locali sia tra Aurora e altre risorse all'interno della stessa regione. AWS

Traffico tra servizio e applicazioni e client locali

Sono disponibili due opzioni di connettività tra la rete privata e: AWS

- Una connessione AWS Site-to-Site VPN. Per ulteriori informazioni, consulta What is AWS Site-to-Site VPN?
- Una AWS Direct Connect connessione. Per ulteriori informazioni, vedi Cos'è AWS Direct Connect?

Puoi accedere ad Amazon Aurora tramite la rete AWS utilizzando operazioni API pubblicate. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Gestione delle identità e degli accessi per Amazon Aurora

AWS Identity and Access Management (IAM) è uno strumento Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS IAMgli amministratori controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Amazon. RDS IAMè un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso con policy
- Come funziona Aurora con IAM
-
- AWS politiche gestite per Amazon RDS
- Aggiornamenti Amazon RDS alle politiche AWS gestite
- Prevenzione del problema "confused deputy" tra servizi
- Autenticazione del database IAM
- Risoluzione dei problemi di identità e accesso in Amazon Aurora

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in .

Utente del servizio: se utilizzi il servizio Aurora per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità di Aurora per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di Aurora, consulta. Risoluzione dei problemi di identità e accesso in Amazon Aurora

Amministratore del servizio: se sei responsabile delle risorse Aurora della tua azienda, probabilmente hai pieno accesso ad Amazon . È tuo compito determinare a quali funzionalità e risorse di Aurora devono accedere i tuoi dipendenti. Devi inviare le richieste all'amministratore per cambiare le

autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere le nozioni di base di IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM Aurora, consulta. Come funziona Aurora con IAM

Amministratore: se sei un amministratore, potresti voler saperne di più su come scrivere policy per gestire l'accesso ad Aurora. Per visualizzare esempi di policy basate sull'identità di Aurora che puoi utilizzare, consulta. IAM

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. È necessario autenticarsi (accedere a AWS) come Utente root dell'account AWS, come IAM utente o assumendo un ruolo. IAM

È possibile accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAMIdentity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando () per firmare crittograficamente le tue richieste utilizzando le tue credenziali. CLI Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta AWS Signature Version 4 per API le richieste nella Guida per l'IAMutente.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'AWS IAM Identity Center utente e Autenticazione a AWS più fattori IAM nella Guida per l'IAMutente.

AWS account utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per l'elenco completo delle attività che richiedono l'accesso come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'IAMutente.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, di un provider di identità Web AWS Directory Service, della directory Identity Center o di qualsiasi utente che accede utilizzando le Servizi AWS credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per informazioni su IAM Identity Center, vedi Cos'è IAM Identity Center? nella Guida AWS IAM Identity Center per l'utente.

IAM users and groups

Un IAMutente è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Laddove possibile, consigliamo di fare affidamento su credenziali temporanee anziché creare IAM utenti con credenziali a lungo termine come password e chiavi di accesso. Tuttavia, se hai casi d'uso specifici che richiedono credenziali a lungo termine con IAM gli utenti, ti consigliamo di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta Ruotare regolarmente le chiavi di accesso per i casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente. IAM

Un gruppo IAM è un'identità che specifica una raccolta di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti

alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile assegnare un nome a un gruppo IAMAdminse concedere a tale gruppo le autorizzazioni per amministrare le risorse. IAM

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Casi d'uso per IAM gli utenti nella Guida per l'IAMutente.

È possibile autenticarsi nel cluster di DB utilizzando l'autenticazione IAM del database.

IAMl'autenticazione del database funziona con Aurora.Per ulteriori informazioni sull'autenticazione tramite il cluster di DB, consulta. IAM Autenticazione del database IAM

Ruoli IAM

Un IAMruolo è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente, ma non è associato a una persona specifica. È possibile assumere temporaneamente un IAM ruolo in AWS Management Console cambiando ruolo. È possibile assumere un ruolo chiamando un' AWS APIoperazione AWS CLI or o utilizzando un'operazione personalizzataURL. Per ulteriori informazioni sui metodi di utilizzo dei ruoli, vedere Utilizzo IAM dei ruoli nella Guida per l'IAMutente.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Autorizzazioni utente temporanee: un utente può assumere un IAM ruolo per acquisire temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per informazioni sui ruoli per la federazione, consulta Creare un ruolo per un provider di identità di terze parti (federazione) nella Guida per l'IAMutente. Se utilizzi IAM Identity Center, configuri un set di autorizzazioni. Per controllare a cosa possono accedere le identità dopo l'autenticazione, IAM Identity Center correla il set di autorizzazioni a un ruolo in. IAM Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (principale attendibile) di un account diverso di accedere alle risorse nel tuo account. I ruoli sono lo strumento

principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per conoscere la differenza tra ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta In che modo IAM i ruoli differiscono dalle politiche basate sulle risorse nella Guida per l'utente. IAM

- Accesso tra servizi: alcuni utilizzano funzionalità in altri. Servizi AWS Servizi AWS Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato: quando utilizzi un IAM utente o un ruolo per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'azione che poi avvia un'altra azione in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. FASle richieste vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un IAMruolo che un servizio assume per eseguire azioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio da IAM. Per ulteriori informazioni, consulta Creare un ruolo per delegare le autorizzazioni a un utente Servizio AWS nella Guida per l'IAMutente.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare, ma non modificare le autorizzazioni dei ruoli collegati ai servizi.
- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un IAM ruolo per gestire le credenziali temporanee per le applicazioni in esecuzione su un'EC2istanza e che effettuano AWS CLI o effettuano AWS API richieste. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un AWS ruolo a un'EC2istanza e renderlo disponibile per tutte le sue applicazioni, crei un profilo di istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzare un IAM ruolo per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella Guida per l'IAMutente.

Per sapere se utilizzare IAM i ruoli, consulta Quando creare un IAM ruolo (anziché un utente) nella Guida per l'IAMutente.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e associandole a IAM identità o AWS risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un'entità (utente root, utente o IAM ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come JSON documenti. Per ulteriori informazioni sulla struttura e il contenuto dei documenti relativi alle JSON politiche, vedere Panoramica delle JSON politiche nella Guida per l'IAMutente.

Un amministratore può utilizzare le policy per specificare chi ha accesso alle AWS risorse e quali azioni può eseguire su tali risorse. Ogni IAM entità (set di autorizzazioni o ruolo) inizia senza autorizzazioni. Ovvero, di default, gli utenti non possono eseguire alcuna operazione, neppure modificare la propria password. Per autorizzare un utente a eseguire operazioni, un amministratore deve allegare una policy di autorizzazioni a tale utente. In alternativa, l'amministratore può aggiungere l'utente a un gruppo che dispone delle autorizzazioni desiderate. Quando un amministratore fornisce le autorizzazioni a un gruppo, le autorizzazioni vengono concesse a tutti gli utenti in tale gruppo.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, indipendentemente dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo da AWS Management Console, da o da. AWS CLI AWS API

Policy basate su identità

Le politiche basate sull'identità sono documenti relativi alle politiche di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un set di autorizzazioni o un ruolo. Tali policy definiscono le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una politica basata sull'identità, consulta Creazione di politiche nella Guida per l'utente. IAM IAM

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo set di autorizzazioni o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più set di autorizzazioni e ruoli nel

tuo account. AWS Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una politica gestita o una politica in linea, consulta Scelta tra politiche gestite e politiche in linea nella Guida per l'IAMutente.

Per informazioni sulle politiche AWS gestite specifiche di Aurora, consulta. AWS politiche gestite per Amazon RDS

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite di autorizzazioni è una funzionalità avanzata in cui si impostano le autorizzazioni massime che una politica basata sull'identità può concedere a un'entità (set di autorizzazioni o ruolo). IAM È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i suoi limiti delle autorizzazioni. Le policy basate su risorse che specificano il set di autorizzazioni o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità nella Guida per l'utente. IAM IAM
- Politiche di controllo del servizio (SCPs): SCPs sono JSON politiche che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più AWS account di proprietà dell'azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. SCPLimita le autorizzazioni per le entità negli account dei membri, inclusa ciascuna Utente root dell'account AWS. Per ulteriori informazioni su Organizations andSCPs, consulta How SCPs work nella AWS Organizations User Guide.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate sull'identità del set di autorizzazioni o del ruolo e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle politiche nella Guida per l'IAMutente.

Come funziona Aurora con IAM

Prima di utilizzare IAM per gestire l'accesso ad Aurora, è necessario comprendere quali IAM funzionalità sono disponibili per l'uso con Amazon .

La tabella seguente elenca IAM le funzionalità che puoi utilizzare con Amazon Aurora:

Caratteristica IAM	Supporto per Amazon Aurora
Policy basate su identità	Sì
Policy basate su risorse	No
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
ACLs	No
Controllo degli accessi basato sugli attributi (ABAC) (tag nelle politiche)	Sì
Credenziali temporanee	Sì
Sessioni di accesso diretto	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	Sì

Per una panoramica generale del funzionamento di Aurora e di AWS altri servizi, AWS consulta i servizi che funzionano IAM nella Guida per IAMl'utente. IAM

Argomenti

- Policy basate sull'identità di Aurora
-
- Azioni politiche per Aurora
- Risorse relative alle policy per Aurora
- Chiavi relative alle condizioni delle politiche per Aurora
- Elenchi di controllo degli accessi (ACLs) in Aurora
-
- Utilizzo di credenziali temporanee con
- Sessioni di accesso diretto per Aurora
- Ruoli di servizio per Aurora
-

Policy basate sull'identità di Aurora

Supporta politiche basate sull'identità: Sì.

Le politiche basate sull'identità sono documenti relativi alle politiche di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un IAM utente, un gruppo di utenti o un ruolo. Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una politica basata sull'identità, consulta Definire le IAM autorizzazioni personalizzate con le politiche gestite dal cliente nella Guida per l'utente. IAM

Con le policy IAM IAM basate su identità, puoi specificare operazioni e risorse consentite o rifiutate, nonché le condizioni in base alle quali le operazioni sono consentite o rifiutate. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per maggiori informazioni su tutti gli elementi che puoi utilizzare in una JSON policy, consulta il riferimento agli elementi della IAM JSON policy nella Guida per l'utente. IAM

Per visualizzare esempi di politiche basate sull'identità di Aurora, consulta.

Supporta politiche basate sulle risorse: No.

Le politiche basate sulle risorse sono documenti di JSON policy allegati a una risorsa. Esempi di politiche basate sulle risorse sono le policy di trust dei IAM ruoli e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso a più account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata su risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando il principale e la risorsa sono diversi Account AWS, un IAM amministratore dell'account fidato deve inoltre concedere all'entità principale (utente o ruolo) l'autorizzazione ad accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta la sezione Cross Account Resource Access IAM nella Guida IAM per l'utente.

Azioni politiche per Aurora

Supporta azioni politiche: Sì.

Gli amministratori possono utilizzare AWS JSON le policy per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'`Action`elemento di una JSON policy descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a una policy. Le azioni politiche in genere hanno lo stesso nome dell' AWS APIoperazione associata. Esistono alcune eccezioni, come le azioni basate solo sulle autorizzazioni che non hanno un'operazione corrispondente. API Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Le azioni politiche in Aurora utilizzano il seguente prefisso prima dell'azione:. `rds:` Ad esempio, per concedere a qualcuno l'autorizzazione a descrivere le istanze DB con l'RDS`DescribeDBInstances`APIoperazione Amazon, includi l'`rds:DescribeDBInstances`azione nella sua politica. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. Aurora definisce il proprio set di azioni che descrivono le attività che puoi eseguire con questo servizio.

Per specificare più operazioni in una singola istruzione, separarle con una virgola come mostrato di seguito.

```
"Action": [
      "rds:action1",
      "rds:action2"
```

Puoi specificare più operazioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le operazioni che iniziano con la parola `Describe`, includi la seguente operazione.

```
"Action": "rds:Describe*"
```

Per visualizzare un elenco di azioni di Aurora, consulta Actions Defined by Amazon RDS nel Service Authorization Reference.

Risorse relative alle policy per Aurora

Supporta risorse politiche: Sì.

Gli amministratori possono utilizzare AWS JSON le policy per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Resource` JSON policy specifica l'oggetto o gli oggetti a cui si applica l'azione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il relativo Amazon Resource Name (ARN). È possibile eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

La risorsa dell'istanza DB ha il seguente Amazon Resource Name (ARN).

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

Per ulteriori informazioni sul formato diARNs, consulta Amazon Resource Names (ARNs) e AWS service namespaces.

Ad esempio, per specificare l'istanza `dbtest` DB nella tua dichiarazione, usa quanto segue. ARN

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

Per specificare tutte le istanze database che appartengono a un account specifico, utilizza il carattere jolly (*).

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

Alcune RDS API operazioni, come quelle per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, utilizza il carattere jolly (*).

```
"Resource": "*"
```

Molte RDS API operazioni di Amazon coinvolgono più risorse. Ad esempio, `CreateDBInstance` crea un'istanza database. Puoi specificare che un utente deve utilizzare un gruppo specifico di sicurezza e un gruppo di parametri quando crea un'istanza database. Per specificare più risorse in un'unica istruzione, separale ARNs con virgole.

```
"Resource": [
      "resource1",
      "resource2"
```

Per visualizzare un elenco dei tipi di risorse di Aurora e relativiARNs, consulta Resources Defined by Amazon RDS nel Service Authorization Reference. Per sapere con quali azioni puoi specificare le caratteristiche ARN di ogni risorsa, consulta Azioni definite da Amazon RDS.

Chiavi relative alle condizioni delle politiche per Aurora

Supporta le chiavi delle condizioni delle politiche specifiche del servizio: Sì.

Gli amministratori possono utilizzare AWS JSON le policy per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition`(o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition`è facoltativo. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition`in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND`logica. Se si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica`OR`. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

È possibile anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi concedere a un utente IAM l'autorizzazione per accedere a una risorsa solo se è stata taggata con il nome utente IAM. Per ulteriori informazioni, consulta Elementi IAM della politica: variabili e tag nella Guida per l'IAMutente.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'IAMutente.

Aurora definisce il proprio set di chiavi di condizione e supporta anche l'utilizzo di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'IAMutente.

Tutte le RDS API operazioni supportano la chiave di `aws:RequestedRegion` condizione.

Per visualizzare un elenco di chiavi di condizione di Aurora, consulta Condition Keys for Amazon RDS nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Actions Defined by Amazon RDS.

Elenchi di controllo degli accessi (ACLs) in Aurora

Supporta gli elenchi di controllo degli accessi (ACLs): No

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLssono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy. JSON

Supporta i tag di controllo degli accessi (ABAC) basati sugli attributi nelle politiche: Sì

Il controllo degli accessi basato sugli attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, questi attributi sono chiamati tag. È possibile allegare tag a IAM entità (utenti o ruoli) e a molte AWS risorse. L'etichettatura di entità e risorse è il primo

passo diABAC. Quindi si progettano ABAC politiche per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa a cui sta tentando di accedere.

ABACè utile in ambienti in rapida crescita e aiuta in situazioni in cui la gestione delle politiche diventa complicata.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni in meritoABAC, vedere Definizione delle autorizzazioni con ABAC autorizzazione nella Guida per l'IAMutente. Per visualizzare un tutorial con i passaggi per la configurazioneABAC, consulta Use Attribute-based access control (ABAC) nella Guida per l'utente. IAM

Per ulteriori informazioni sull'etichettatura delle risorse di Aurora, consulta. Specifica delle condizioni: Utilizzo di tag personalizzati Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Concedere l'autorizzazione per le operazioni su una risorsa con un tag specifico con due valori diversi.

Utilizzo di credenziali temporanee con

Supporta credenziali temporanee: Sì.

Alcune Servizi AWS non funzionano quando si accede utilizzando credenziali temporanee. Per ulteriori informazioni, incluse quelle che Servizi AWS funzionano con credenziali temporanee, consulta la sezione Servizi AWS relativa alla funzionalità IAM nella Guida per l'IAMutente.

Si utilizzano credenziali temporanee se si accede AWS Management Console utilizzando qualsiasi metodo tranne il nome utente e la password. Ad esempio, quando accedete AWS utilizzando il link Single Sign-on (SSO) della vostra azienda, tale processo crea automaticamente credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sul cambio di ruolo, consulta Passare da un utente a un IAM ruolo (console) nella Guida per l'IAMutente.

È possibile creare manualmente credenziali temporanee utilizzando AWS CLI o AWS API. È quindi possibile utilizzare tali credenziali temporanee per accedere. AWS AWS consiglia di generare

dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, vedere Credenziali di sicurezza temporanee in. IAM

Sessioni di accesso diretto per Aurora

Supporta sessioni di accesso diretto: Sì.

Quando utilizzi un IAM utente o un ruolo per eseguire azioni AWS, sei considerato un preside. Quando si utilizzano alcuni servizi, è possibile eseguire un'azione che quindi avvia un'altra azione in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. FASle richieste vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.

Ruoli di servizio per Aurora

Supporta i ruoli di servizio: Sì.

Un ruolo di servizio è un IAMruolo che un servizio assume per eseguire azioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio da IAM. Per ulteriori informazioni, consulta Creare un ruolo per delegare le autorizzazioni a un utente Servizio AWS nella Guida per l'IAMutente.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità di Aurora. Modifica i ruoli di servizio solo quando Aurora fornisce indicazioni in tal senso.

Supporta ruoli collegati ai servizi: Sì.

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare, ma non modificare le autorizzazioni dei ruoli collegati ai servizi.

Per informazioni dettagliate sull'utilizzo dei ruoli collegati ai servizi di Aurora, consulta. Utilizzo di ruoli collegati ai servizi per Amazon Aurora

Per impostazione predefinita, i set di autorizzazioni e i ruoli non dispongono dell'autorizzazione per creare o modificare Aurora. Inoltre, non possono eseguire attività utilizzando AWS Management Console AWS CLI, o AWS API. Un amministratore deve creare IAM politiche che concedano ai set di autorizzazioni e ai ruoli il permesso di eseguire API operazioni specifiche sulle risorse specificate di cui ha bisogno. L'amministratore deve quindi collegare queste policy ai set di autorizzazioni o ai ruoli che richiedono tali autorizzazioni.

Per informazioni su come creare una politica IAM basata sull'identità utilizzando questi documenti di esempioJSON, consulta Creazione di politiche nella JSON scheda della Guida per l'utente. IAM

Argomenti

- Best practice per le policy
- Utilizzo della console Aurora
- Autorizzazioni necessarie per l'uso della console
- Consentire agli utenti di visualizzare le loro autorizzazioni
-
- Policy di esempio: Utilizzo di chiavi di condizione
- Specifica delle condizioni: Utilizzo di tag personalizzati

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare RDS risorse Amazon nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta le politiche AWS gestite o le politiche AWS gestite per le funzioni lavorative nella Guida per l'IAMutente.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le IAM politiche, concedi solo le autorizzazioni necessarie per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche

come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo per applicare le autorizzazioni, consulta Politiche e autorizzazioni nella Guida IAM per l'utente. IAM IAM

- Utilizza le condizioni nelle IAM politiche per limitare ulteriormente l'accesso: puoi aggiungere una condizione alle tue politiche per limitare l'accesso ad azioni e risorse. Ad esempio, puoi scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzandoSSL. È inoltre possibile utilizzare condizioni per concedere l'accesso alle azioni di servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta Elementi IAM JSON della politica: Condizione nella Guida IAM per l'utente.
- Usa IAM Access Analyzer per convalidare IAM le tue policy e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio delle IAM policy () e alle best practice. JSON IAM IAM Access Analyzer fornisce più di 100 controlli delle policy e consigli pratici per aiutarti a creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle politiche con IAM Access Analyzer nella Guida per l'utente. IAM
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede l'utilizzo di IAM utenti o di un utente root Account AWS, attiva questa opzione MFA per una maggiore sicurezza. Per richiedere MFA quando vengono richiamate API le operazioni, aggiungi MFA delle condizioni alle tue politiche. Per ulteriori informazioni, consulta Secure API access with MFA nella Guida IAM per l'utente.

Per ulteriori informazioni sulle best practice inIAM, consulta la sezione Procedure consigliate in materia di sicurezza IAM nella Guida IAM per l'utente.

Utilizzo della console Aurora

Per accedere alla console Amazon Aurora, devi disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Aurora presenti nel tuo. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso la o la AWS CLI . AWS API Consenti invece l'accesso solo alle azioni che corrispondono all'APIoperazione che stai cercando di eseguire.

Per garantire che tali entità possano ancora utilizzare la console Aurora, allega anche la seguente policy AWS gestita alle entità.

```
AmazonRDSReadOnlyAccess
```

Per ulteriori informazioni, consulta Aggiungere autorizzazioni a un utente nella Guida per l'IAMutente.

Autorizzazioni necessarie per l'uso della console

Affinché un utente possa utilizzare la console, è necessario che disponga di un set minimo di autorizzazioni. Queste autorizzazioni consentono all'utente di descrivere le risorse Aurora per il AWS proprio account e di fornire altre informazioni correlate, tra cui informazioni sulla sicurezza e sulla EC2 rete di Amazon.

Se si crea una policy IAM più restrittiva delle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per gli utenti con tale policy IAM. Per garantire che gli utenti possano continuare a usare la console, collega anche la policy gestita `AmazonRDSReadOnlyAccess` all'utente, come descritto in Gestione dell'accesso con policy.

Non è necessario consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso Amazon AWS CLI o verso Amazon RDSAPI.

La seguente politica garantisce l'accesso completo a tutte le risorse Amazon Aurora per l' AWS account root:

```
AmazonRDSFullAccess
```

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando o a livello di codice. AWS CLI AWS API

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
```

```
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Le seguenti sezioni presentano esempi di politiche di autorizzazione che concedono e limitano l'accesso alle risorse:

Consenti a un utente di creare istanze DB in un account AWS

Di seguito è riportato un esempio di politica che consente all'account con l'ID di `123456789012` creare istanze DB per il tuo AWS account. La policy richiede che il nome della nuova istanza database inizi con `test`. La nuova istanza DB deve inoltre utilizzare il motore di SQL database My e la classe di istanze `db.t2.micro` DB. Inoltre, la nuova istanza database deve utilizzare un gruppo di opzioni e un gruppo di parametri database che inizia con `default`, e deve utilizzare il gruppo di sottoreti `default`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
```

```
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
            }
         }
      }
   ]
}
```

La policy include una singola istruzione che specifica le autorizzazioni seguenti per l'utente :

- La policy consente all'account di creare un'istanza DB utilizzando l'reateDBInstanceAPIoperazione C (ciò vale anche per il create-db-instance AWS CLI comando e il AWS Management Console).
- L'elemento `Resource` specifica che l'utente può eseguire azioni in o con altre risorse. Specifichi le risorse utilizzando un Amazon Resources Name (ARN). Ciò ARN include il nome del servizio a cui appartiene la risorsa (`rds`), la AWS regione (*indica qualsiasi regione in questo esempio), il numero di AWS account (123456789012è il numero di account in questo esempio) e il tipo di risorsa. Per ulteriori informazioni sulla creazioneARNs, vedereNomi di risorse Amazon (ARNs) in Amazon RDS.

  L'elemento `Resource` nell'esempio specifica i seguenti vincoli della policy sulle risorse per l'utente:

  - L'identificatore istanze DB per la nuova istanza database deve iniziare con `test` (per esempio, `testCustomerData1`, `test-region2-data`).
  - Il gruppo di opzioni per la nuova istanza database deve iniziare con `default`.
  - Il gruppo di parametri database per la nuova istanza database deve iniziare con `default`.
  - Il gruppo di sottoreti per la nuova istanza database deve essere il gruppo di sottoreti `default`.
- L'`Condition`elemento specifica che il motore DB deve essere My SQL e la classe dell'istanza DB deve esseredb.t2.micro. L'elemento `Condition` specifica le condizioni quando deve

essere applicata una policy. È possibile aggiungere permessi o restrizioni aggiuntivi usando l'elemento `Condition`. Per ulteriori informazioni su come specificare le condizioni;, consulta Chiavi relative alle condizioni delle politiche per Aurora. Questo esempio specifica le condizioni `rds:DatabaseEngine` e `rds:DatabaseClass`. Per informazioni sui valori delle condizioni validi per`rds:DatabaseEngine`, consultate l'elenco sotto il `Engine` parametro in C. reateDBInstance Per informazioni sui valori di condizione validi per `rds:DatabaseClass`, consulta Motori DB supportati per classi di istanza database.

La policy non specifica l'elemento `Principal` poiché in una policy basata su identità l'entità che ottiene l'autorizzazione non viene specificata. Quando si collega una policy a un utente, quest'ultimo è l'entità implicita. Quando si associa una politica di autorizzazione a un IAM ruolo, il principale identificato nella politica di fiducia del ruolo ottiene le autorizzazioni.

Per visualizzare un elenco di azioni di Aurora, consulta Actions Defined by Amazon RDS nel Service Authorization Reference.

Consenti a un utente di eseguire qualsiasi azione di descrizione su qualsiasi risorsa RDS

La seguente policy di autorizzazione concede a un utente le autorizzazioni per eseguire tutte le operazioni che iniziano con `Describe`. Queste azioni mostrano informazioni su una RDS risorsa, ad esempio un'istanza DB. Il carattere jolly (*) nell'`Resource`elemento indica che le azioni sono consentite per tutte le risorse Aurora di proprietà dell'account.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

Consentire a un utente di creare un'istanza database che utilizza il gruppo parametri del database e il gruppo di sottorete specificati

La seguente policy di autorizzazioni concede le autorizzazioni per consentire a un utente di creare un'istanza database che deve usare il gruppo di parametri database `mydbpg` e il gruppo di sottorete DB `mydbsubnetgroup`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
         ]
      }
   ]
}
```

Concedere l'autorizzazione per le operazioni su una risorsa con un tag specifico con due valori diversi

Puoi utilizzare le condizioni della tua policy basata sull'identità per controllare l'accesso alle risorse di Amazon in base ai tag. La seguente policy consente l'autorizzazione a eseguire l'`CreateDBSnapshot`APIoperazione su istanze DB con il `stage` tag impostato su o. `development` `test`

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
```

```
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La seguente politica consente l'autorizzazione a eseguire l'`ModifyDBInstance`APIoperazione su istanze DB con il `stage` tag impostato su o. `development` `test`

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
```

```
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

Impedire a un utente di eliminare un'istanza database

La seguente policy di autorizzazione assegna le autorizzazioni per impedire a un utente di eliminare un'istanza database specifica. Ad esempio, potresti voler negare la possibilità di eliminare le istanze database di produzione a qualsiasi utente che non sia un amministratore.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

Negare tutti gli accessi a una risorsa

È anche possibile negare esplicitamente l'accesso a una risorsa. I criteri di negazione hanno la precedenza sui criteri di autorizzazione. La policy seguente nega esplicitamente a un utente la possibilità di gestire una risorsa:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
```

```
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
   ]
}
```

Policy di esempio: Utilizzo di chiavi di condizione

Di seguito sono illustrati esempi di come è possibile utilizzare le chiavi di condizione nelle policy di autorizzazioni IAM di Amazon Aurora.

Esempio 1: Concessione dell'autorizzazione per creare un'istanza database che utilizza un motore del database specifico e non è Multi-AZ

La policy seguente utilizza una chiave di condizione RDS e permette a un utente di creare solo istanze database che utilizzano il motore del database MySQL e non utilizzano Multi-AZ. L'elemento `Condition` indica il requisito secondo cui il motore del database deve essere MySQL.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
}
```

Esempio 2: Rifiuto esplicito dell'autorizzazione per creare istanze database per determinate classi di istanze database e per creare istanze database che utilizzano Provisioned IOPS

La policy seguente rifiuta in modo esplicito l'autorizzazione per creare istanze database che utilizzano le classi di istanze database `r3.8xlarge` e `m4.10xlarge`, che sono le istanze database di dimensioni maggiori e più costose. Questa policy impedisce anche gli utenti di creare istanze database che utilizzano Provisioned IOPS, che comporta costi aggiuntivi.

Il rifiuto esplicito di un'autorizzazione prevale su qualsiasi altra autorizzazione concessa. In questo modo si evita che le identità ottengano accidentalmente un'autorizzazione che non desideravi concedere.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
      },
      {
         "Sid": "DenyPIOPSCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "NumericNotEquals": {
               "rds:Piops": "0"
            }
         }
      }
   ]
}
```

Esempio 3: limita il set di chiavi di tag e valori che possono essere utilizzati per aggiungere tag a una risorsa

La policy seguente utilizza una chiave di condizione RDS che consente l'aggiunta di un tag con la chiave `stage` per essere aggiunta alla risorsa con i valori `test`, `qa` e `production`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource": "*",
         "Condition": {
            "streq": {
               "rds:req-tag/stage": [
                  "test",
                  "qa",
                  "production"
               ]
            }
         }
      }
   ]
}
```

Specifica delle condizioni: Utilizzo di tag personalizzati

Amazon Aurora permette di specificare condizioni in una policy IAM utilizzando tag personalizzati.

Ad esempio, supponi di aggiungere alle istanze database un tag denominato `environment` con valori quali `beta`, `staging`, `production` e così via. In questo modo, puoi creare una policy che limita alcuni utenti alle istanze database in base al valore del tag `environment`.

> **Note**
>
> Per gli identificatori di tag personalizzati viene fatta distinzione tra maiuscole e minuscole.

Nella tabella seguente sono elencati gli identificatori di tag RDS che è possibile utilizzare in un elemento `Condition`.

Identificatore di tag RDS	Si applica a
`db-tag`	Istanze database, incluse repliche di lettura
`snapshot-tag`	Snapshot DB
`ri-tag`	Istanze database riservate
`og-tag`	Gruppi di opzioni database
`pg-tag`	Gruppi di parametri database
`subgrp-tag`	Gruppi di sottoreti database
`es-tag`	Abbonamenti a eventi
`cluster-tag`	Cluster database
`cluster-pg-tag`	Gruppi di parametri di cluster database
`cluster-snapshot-tag`	Snapshot cluster database

La sintassi per una condizione di un tag personalizzato è la seguente:

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name":
["value"]} }
```

Ad esempio, l'elemento `Condition` seguente si applica alle istanze database con un tag denominato `environment` e un valore di tag corrispondente a `production`.

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

Per informazioni sulla creazione di tag, consulta Etichettatura di Amazon Aurora e risorse Amazon RDS.

Important

Se gestisci l'accesso alle risorse RDS tramite tagging, è consigliabile proteggere l'accesso ai tag per le risorse RDS. È possibile gestire l'accesso ai tag creando policy per le operazioni `AddTagsToResource` e `RemoveTagsFromResource`. Ad esempio, la policy seguente nega agli utenti la possibilità di aggiungere o rimuovere tag per tutte le risorse. Puoi quindi creare policy per permettere a utenti specifici di aggiungere o rimuovere tag.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

Per visualizzare un elenco di operazioni di Aurora, consulta Operazioni definite da Amazon RDS nella Service Authorization Reference.

Policy di esempio: Utilizzo di tag personalizzati

Di seguito sono illustrati esempi di come è possibile utilizzare i tag personalizzati nelle policy di autorizzazioni IAM di Amazon Aurora. Per ulteriori informazioni sull'aggiunta di tag a una risorsa Amazon Aurora, consulta Nomi di risorse Amazon (ARNs) in Amazon RDS.

Note

Tutti gli esempi utilizzano la regione us-west-2 e contengono account fittizi. IDs

Esempio 1: Concessione dell'autorizzazione per le operazioni su una risorsa con un tag specifico con due valori diversi

La seguente policy concede l'autorizzazione per eseguire l'operazione API `CreateDBSnapshot` sulle istanze database con il tag `stage` impostato su `development` o `test`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La seguente policy concede l'autorizzazione per eseguire l'operazione API `ModifyDBInstance` sulle istanze database con il tag `stage` impostato su `development` o `test`.

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":" [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

Esempio 2: Rifiuto esplicito dell'autorizzazione per creare un'istanza database che utilizza gruppi di parametri database specificati

La policy seguente rifiuta in modo esplicito l'autorizzazione per creare un'istanza database che utilizza gruppi di parametri database con valori di tag specifici. Puoi applicare questa policy se desideri che uno specifico gruppo di parametri database creato dal cliente venga sempre utilizzato nella creazione di istanze database. Le policy che utilizzano `Deny` servono per lo più a limitare l'accesso concesso da una policy più ampia.

Il rifiuto esplicito di un'autorizzazione prevale su qualsiasi altra autorizzazione concessa. In questo modo si evita che le identità ottengano accidentalmente un'autorizzazione che non desideravi concedere.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

Esempio 3: Concessione dell'autorizzazione per le operazioni in un'istanza database con un nome di istanza che ha come prefisso un nome utente

La policy seguente concede l'autorizzazione per chiamare qualsiasi API (ad eccezione di `AddTagsToResource` o `RemoveTagsFromResource`) in un'istanza database con un nome di istanza che ha come prefisso il nome dell'utente e che dispone di un tag denominato `stage` equivalente a `devo` o che non dispone di un tag `stage`.

La riga `Resource` nella policy identifica una risorsa in base al relativo Amazon Resource Name (ARN). Per ulteriori informazioni sull'utilizzo ARNs con le risorse Amazon Aurora, consulta. Nomi di risorse Amazon (ARNs) in Amazon RDS

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
```

```
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

AWS politiche gestite per Amazon RDS

Per aggiungere autorizzazioni ai set di autorizzazioni e ai ruoli, è più facile utilizzare le politiche AWS gestite che scrivere le politiche da soli. Ci vogliono tempo ed esperienza per creare politiche gestite dai IAM clienti che forniscano al team solo le autorizzazioni di cui ha bisogno. Per iniziare rapidamente, puoi utilizzare le nostre politiche AWS gestite. Queste policy coprono i casi d'uso comuni e sono disponibili nel tuo Account AWS. Per ulteriori informazioni sulle politiche AWS gestite, consulta le politiche AWS gestite nella Guida IAM per l'utente.

Servizi AWS mantenere e aggiornare le politiche AWS gestite. Non è possibile modificare le autorizzazioni nelle politiche AWS gestite. I servizi aggiungono occasionalmente autorizzazioni aggiuntive a una policy AWS gestita per supportare nuove funzionalità. Questo tipo di aggiornamento interessa tutte le identità (set di autorizzazioni e ruoli) a cui è collegata la policy. È più probabile che i servizi aggiornino una politica AWS gestita quando viene lanciata una nuova funzionalità o quando diventano disponibili nuove operazioni. I servizi non rimuovono le autorizzazioni da una policy AWS gestita, quindi gli aggiornamenti delle policy non compromettono le autorizzazioni esistenti.

Inoltre, AWS supporta politiche gestite per le funzioni lavorative che si estendono su più servizi. Ad esempio, la policy `ReadOnlyAccess` AWS gestita fornisce l'accesso in sola lettura a tutte le Servizi AWS risorse. Quando un servizio lancia una nuova funzionalità, AWS aggiunge autorizzazioni di sola lettura per nuove operazioni e risorse. Per un elenco e le descrizioni delle politiche relative alle funzioni lavorative, consulta le politiche AWS gestite per le funzioni lavorative nella Guida per l'utente. IAM

Argomenti

- AWS politica gestita: A mazonRDSRead OnlyAccess
- AWS politica gestita: A mazonRDSFull Access
- AWS politica gestita: A mazonRDSData FullAccess
- AWS politica gestita: A mazonRDSEnhanced MonitoringRole
- AWS politica gestita: A mazonRDSPerformance InsightsReadOnly
- AWS politica gestita: A mazonRDSPerformance InsightsFullAccess
- AWS politica gestita: A mazonRDSDirectory ServiceAccess
- AWS politica gestita: A mazonRDSService RolePolicy
- AWS politica gestita: A mazonRDSPreview ServiceRolePolicy
- AWS politica gestita: A mazonRDSBeta ServiceRolePolicy

AWS politica gestita: A mazonRDSRead OnlyAccess

Questa politica consente l'accesso in sola lettura ad Amazon RDS tramite. AWS Management Console

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`— Consente ai responsabili di descrivere RDS le risorse Amazon ed elencare i tag per RDS le risorse Amazon.
- `cloudwatch`— Consente ai mandanti di ottenere statistiche sui CloudWatch parametri di Amazon.
- `ec2`: consente ai principali di descrivere le zone di disponibilità e le risorse di rete.
- `logs`— Consente ai responsabili di descrivere CloudWatch Logs, i flussi di log dei gruppi di log e di ottenere gli eventi di log di Logs. CloudWatch
- `devops-guru`— Consente ai responsabili di descrivere le risorse che hanno una copertura Amazon DevOps Guru, specificata dai nomi degli CloudFormation stack o dai tag delle risorse.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, consulta A mazonRDSRead OnlyAccess nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSFull Access

Questa politica fornisce l'accesso completo ad Amazon RDS tramite AWS Management Console.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`— Consente ai mandanti l'accesso completo ad AmazonRDS.
- `application-autoscaling`: consente ai principali di descrivere e gestire gli obiettivi e le policy di dimensionamento di Application Auto Scaling.
- `cloudwatch`— Consente ai presidi di ottenere statistiche CloudWatch metriche e gestire gli allarmi. CloudWatch
- `ec2`: consente ai principali di descrivere le zone di disponibilità e le risorse di rete.
- `logs`— Consente ai responsabili di descrivere CloudWatch Logs, log, flussi di log dei gruppi di log e ottenere gli eventi di log di Logs. CloudWatch

- `outposts`— Consente ai principali di ottenere i tipi di istanza. AWS Outposts
- `pi`: consente ai principali di ottenere i parametri di Performance Insights.
- `sns`— Consente ai responsabili degli abbonamenti e degli argomenti di Amazon Simple Notification Service (AmazonSNS) e di pubblicare messaggi AmazonSNS.
- `devops-guru`— Consente ai responsabili di descrivere le risorse che hanno una copertura Amazon DevOps Guru, specificata dai nomi degli CloudFormation stack o dai tag delle risorse.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, consulta A mazonRDSFull Access in the AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSData FullAccess

Questa politica consente l'accesso completo all'utilizzo dei dati API e dell'editor di query sui Aurora Serverless cluster in uno specifico Account AWS. Questa politica consente di Account AWS ottenere il valore di un segreto da AWS Secrets Manager.

Puoi allegare la `AmazonRDSDataFullAccess` politica alle tue IAM identità.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `dbqms`: consente ai principali di accedere, creare, eliminare, descrivere e aggiornare le query. Il Database Query Metadata Service (`dbqms`) è un servizio solo interno. Fornisce le tue query recenti e salvate per l'editor di query su più Servizi AWS, incluso AmazonRDS. AWS Management Console
- `rds-data`— Consente ai principali di eseguire SQL istruzioni sui Aurora Serverless database.
- `secretsmanager`— Consente ai principali di ottenere il valore di un segreto da. AWS Secrets Manager

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, vedere A mazonRDSData FullAccess nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSEnhanced MonitoringRole

Questa policy fornisce l'accesso ad Amazon CloudWatch Logs for Amazon RDS Enhanced Monitoring.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `logs`— Consente ai responsabili di creare CloudWatch Logs, gruppi di log e politiche di conservazione e di creare e descrivere i flussi di log dei gruppi di log CloudWatch Logs. Consente inoltre ai responsabili di inserire e ottenere gli eventi di log. CloudWatch

Per ulteriori informazioni su questa politica, incluso il documento JSON sulla politica, vedere A mazonRDSEnhanced MonitoringRole nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSPerformance InsightsReadOnly

Questa policy fornisce l'accesso in sola lettura ad Amazon RDS Performance Insights per istanze Amazon RDS DB e cluster Amazon Aurora DB.

Questa policy ora include `Sid` (ID istruzione) come identificativo per l'istruzione della policy.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`— Consente ai responsabili di descrivere le istanze di Amazon RDS DB e i cluster di database Amazon Aurora.
- `pi`— Consente ai responsabili di effettuare chiamate ad Amazon RDS Performance Insights API e accedere ai parametri di Performance Insights.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, consulta A mazonRDSPerformance InsightsReadOnly nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSPerformance InsightsFullAccess

Questa policy fornisce l'accesso completo ad Amazon RDS Performance Insights per le istanze Amazon RDS DB e i cluster Amazon Aurora DB.

Questa policy ora include `Sid` (ID istruzione) come identificativo per l'istruzione della policy.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `rds`— Consente ai responsabili di descrivere le istanze di Amazon RDS DB e i cluster di database Amazon Aurora.
- `pi`— Consente ai responsabili di effettuare chiamate ad Amazon RDS Performance Insights API e di creare, visualizzare ed eliminare report di analisi delle prestazioni.
- `cloudwatch`— Consente ai responsabili di elencare tutte le CloudWatch metriche di Amazon e ottenere dati e statistiche sui parametri.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, consulta A mazonRDSPerformance InsightsFullAccess nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSDirectory ServiceAccess

Questa politica consente RDS ad Amazon di effettuare chiamate verso AWS Directory Service.

Dettagli dell'autorizzazione

Questa policy include la seguente autorizzazione:

- `ds`— Consente ai responsabili di descrivere le AWS Directory Service directory e di controllare l'autorizzazione alle AWS Directory Service directory.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, vedere A mazonRDSDirectory ServiceAccess nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSService RolePolicy

Non puoi allegare la `AmazonRDSServiceRolePolicy` politica alle tue IAM entità. Questa politica è associata a un ruolo collegato al servizio che consente RDS ad Amazon di eseguire azioni per tuo conto. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora.

AWS politica gestita: A mazonRDSPreview ServiceRolePolicy

Non dovresti attaccarti `AmazonRDSPreviewServiceRolePolicy` alle tue IAM entità. Questa policy è associata a un ruolo collegato al servizio che consente RDS ad Amazon di chiamare AWS i servizi per conto delle tue risorse RDS DB. Per ulteriori informazioni, consulta Ruolo collegato ai servizi per Amazon RDS Preview.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `ec2`- Consente ai responsabili di descrivere le zone di disponibilità e le risorse di rete.
- `secretsmanager`— Consente ai presidi di ottenere il valore di un segreto da. AWS Secrets Manager
- `cloudwatch`, `logs` - Consente RDS ad Amazon di caricare i parametri e i log delle istanze DB CloudWatch tramite CloudWatch agente.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, consulta AmazonRDSData FullAccess nella AWS Managed Policy Reference Guide.

AWS politica gestita: A mazonRDSBeta ServiceRolePolicy

Non dovresti attaccarti `AmazonRDSBetaServiceRolePolicy` alle tue IAM entità. Questa policy è associata a un ruolo collegato al servizio che consente RDS ad Amazon di chiamare AWS i servizi per conto delle tue risorse RDS DB. Per ulteriori informazioni, consulta Autorizzazioni di ruolo collegate ai servizi per Amazon RDS Beta.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `ec2`- Consente RDS ad Amazon di eseguire operazioni di backup sull'istanza DB che fornisce funzionalità di point-in-time ripristino.
- `secretsmanager`- Consente RDS ad Amazon di gestire i segreti specifici delle istanze DB creati da AmazonRDS.
- `cloudwatch`, `logs` - Consente RDS ad Amazon di caricare i parametri e i log delle istanze DB CloudWatch tramite CloudWatch agente.

Per ulteriori informazioni su questa politica, incluso il documento relativo alla JSON policy, consulta AmazonRDSBeta ServiceRolePolicy nella AWS Managed Policy Reference Guide.

Aggiornamenti Amazon RDS alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon RDS da quando questo servizio ha iniziato a tracciare queste modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivere il feed RSS nella pagina di Cronologia dei documenti di Amazon RDS.

Modifica	Descrizione	Data
AWS politica gestita: AmazonRDSPreviewServiceRolePolicy: aggiornamento a policy esistente	Amazon RDS ha rimosso `sns:Publish` l'autorizzazione dal `AmazonRDSPreviewServiceRolePolicy` ruolo collegato al `AWSServiceRoleForRDSPreview` servizio. Per ulteriori informazioni, consulta AWS politica gestita: AmazonRDSPreviewServiceRolePolicy.	7 agosto 2024
AWS politica gestita: AmazonRDSBetaServiceRolePolicy: aggiornamento a policy esistente	Amazon RDS ha rimosso `sns:Publish` l'autorizzazione dal `AmazonRDSBetaServiceRolePolicy` ruolo collegato al `AWSServiceRoleForRDSBeta` servizio. Per ulteriori informazioni, consulta AWS politica gestita: AmazonRDSBetaServiceRolePolicy.	7 agosto 2024
AWS politica gestita: AmazonRDSServiceRolePolicy: aggiornamento a policy esistente	Amazon RDS ha rimosso `sns:Publish` l'autorizzazione dal `AmazonRDSServiceRolePolicy` ruolo collegato al	2 luglio 2024

Modifica	Descrizione	Data
	`AWSServiceRoleForRDS` servizio. Per ulteriori informazioni, consulta AWS politica gestita: A mazonRDSService RolePolicy.	
AWS politiche gestite per Amazon RDS: aggiornamento a policy esistente	Amazon RDS ha aggiunto una nuova autorizzazione al ruolo `AWSServiceRoleForRDSCustom` collegato al servizio per consentire a RDS Custom for SQL Server di modificare il tipo di istanza host del database sottostante. `AmazonRDSCustomServiceRolePolicy` RDS ha inoltre aggiunto l'`ec2:DescribeInstanceTypes` autorizzazione a ottenere informazioni sul tipo di istanza per l'host del database. Per ulteriori informazioni, consulta AWS politiche gestite per Amazon RDS.	8 aprile 2024

Modifica	Descrizione	Data
AWS politiche gestite per Amazon RDS: nuova policy	Amazon RDS ha aggiunto una nuova policy gestita denominata `AmazonRDSCustomInstanceProfileRolePolicy` per consentire a RDS Custom di eseguire azioni di automazione e attività di gestione del database tramite un profilo di EC2 istanza. Per ulteriori informazioni, consulta AWS politiche gestite per Amazon RDS.	27 febbraio 2024
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto una nuova dichiarazione IDs al `AmazonRDSServiceRolePolicy` ruolo collegato ai `AWSServiceRoleForRDS` servizi. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora.	19 gennaio 2024

Modifica	Descrizione	Data
AWS politiche gestite per Amazon RDS: aggiornamento a policy esistenti	Le policy gestite `AmazonRDSPerformanceInsightsReadOnly` e `AmazonRDSPerformanceInsightsFullAccess` ora includono `Sid` (ID istruzione) come identificativo nell'istruzione della policy. Per ulteriori informazioni, consulta AWS politica gestita: A mazonRDSPerformance InsightsReadOnly e AWS politica gestita: A mazonRDSPerformance InsightsFullAccess.	23 ottobre 2023
AWS politiche gestite per Amazon RDS: aggiornamento a policy esistente	Amazon RDS ha aggiunto nuove autorizzazioni alla policy gestita `AmazonRDSFullAccess`. Le autorizzazioni consentono di generare, visualizzare ed eliminare il report di analisi delle prestazioni per un periodo di tempo. Per ulteriori informazioni sulla configurazione delle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights	17 agosto 2023

Modifica	Descrizione	Data
AWS politiche gestite per Amazon RDS: nuova policy e aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuove autorizzazioni alla policy gestita `AmazonRDSPerformanceInsightsReadOnly` e una nuova policy gestita denominata `AmazonRDSPerformanceInsightsFullAccess`. Queste autorizzazioni consentono di analizzare Performance Insights per un periodo di tempo, visualizzare i risultati dell'analisi insieme ai suggerimenti ed eliminare i report. Per ulteriori informazioni sulla configurazione delle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights	16 agosto 2023

Modifica	Descrizione	Data
AWS politiche gestite per Amazon RDS: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei CloudWatch nomi `ListMetrics` Amazon a and. `AmazonRDSFullAccess` `AmazonRDSReadOnlyAccess` Questo spazio dei nomi è necessario affinché Amazon RDS elenchi specifici parametri di utilizzo delle risorse. Per ulteriori informazioni, consulta Panoramica della gestione delle autorizzazioni di accesso alle tue CloudWatch risorse nella Amazon CloudWatch User Guide.	4 aprile 2023

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuove autorizzazioni al ruolo `AWSServiceRoleForRDS` collegato al `AmazonRDSServiceRolePolicy` servizio per l'integrazione con. AWS Secrets Manager RDS richiede l'integrazione con Secrets Manager per la gestione delle password degli utenti master in Secrets Manager. Il segreto utilizza una convenzione di denominazione riservata e limita gli aggiornamenti del cliente. Per ulteriori informazioni, consulta Gestione delle password con Amazon Aurora e AWS Secrets Manager.	22 dicembre 2022

Modifica	Descrizione	Data
AWS politiche gestite per Amazon RDS: aggiornamento a policy esistenti	Amazon RDS ha aggiunto una nuova autorizzazione alle policy `AmazonRDSFullAccess` e alle policy `AmazonRDSReadOnlyAccess` gestite per consentirti di attivare Amazon DevOps Guru nella console RDS. Questa autorizzazione è necessaria per verificare se DevOps Guru è attivo. Per ulteriori informazioni, consulta Configurazione delle politiche di accesso per Guru per IAM DevOps RDS.	19 dicembre 2022
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei CloudWatch nomi Amazon a for. `AmazonRDSPreviewServiceRolePolicy` `PutMetricData` Questo spazio dei nomi è necessario affinché Amazon RDS pubblichi i parametri di utilizzo delle risorse. Per ulteriori informazioni, consulta Usare le chiavi di condizione per limitare l'accesso ai CloudWatch namespace nella Amazon CloudWatch User Guide.	7 luglio 2022

Modifica	Descrizione	Data
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei CloudWatch nomi Amazon a for. `AmazonRDSBetaServiceRolePolicy` `PutMetricData` Questo spazio dei nomi è necessario affinché Amazon RDS pubblichi i parametri di utilizzo delle risorse. Per ulteriori informazioni, consulta Usare le chiavi di condizione per limitare l'accesso ai CloudWatch namespace nella Amazon CloudWatch User Guide.	7 luglio 2022
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto un nuovo spazio dei CloudWatch nomi Amazon a for. `AWSServiceRoleForRDS` `PutMetricData` Questo spazio dei nomi è necessario affinché Amazon RDS pubblichi i parametri di utilizzo delle risorse. Per ulteriori informazioni, consulta Usare le chiavi di condizione per limitare l'accesso ai CloudWatch namespace nella Amazon CloudWatch User Guide.	22 aprile 2022

Modifica	Descrizione	Data
AWS politiche gestite per Amazon RDS: nuova policy	Amazon RDS ha aggiunto una nuova policy gestita denominata `AmazonRDSPerformanceInsightsReadOnly` per consentire ad Amazon RDS di chiamare i AWS servizi per conto delle tue istanze DB. Per ulteriori informazioni sulla configurazione delle policy di accesso per Performance Insights, consulta Configurazione delle policy di accesso per Performance Insights	10 marzo 2022
Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora: aggiornamento a una policy esistente	Amazon RDS ha aggiunto nuovi CloudWatch namespace Amazon a for. `AWSServiceRoleForRDS` `PutMetricData` Questi namespace sono necessari per Amazon DocumentDB (con compatibilità MongoDB) e Amazon Neptune per pubblicare i parametri. CloudWatch Per ulteriori informazioni, consulta Usare le chiavi di condizione per limitare l'accesso ai CloudWatch namespace nella Amazon CloudWatch User Guide.	4 marzo 2022

Modifica	Descrizione	Data
Amazon RDS ha iniziato a monitorare le modifiche	Amazon RDS ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	26 ottobre 2021

Prevenzione del problema "confused deputy" tra servizi

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. In AWS, la rappresentazione cross-service può comportare il problema confused deputy.

La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per aiutarti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse del tuo account. Per ulteriori informazioni, consultare Problema del "confused deputy" nella Guida per l'utente di IAM.

Per limitare le autorizzazioni alle risorse che Amazon RDS fornisce a un altro servizio per una risorsa specifica, si consiglia di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse.

In alcuni casi, il valore `aws:SourceArn` non contiene l'ID dell'account, ad esempio quando utilizzi Amazon Resource Name (ARN) per un bucket Simple Storage Service (Amazon S3). In questi casi, assicurati di utilizzare entrambe le chiavi di contesto delle condizioni globali per limitare le autorizzazioni. In alcuni casi, si utilizzano le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID dell'account. In questi casi, assicurati che il valore `aws:SourceAccount` e l'account in `aws:SourceArn` utilizzano lo stesso ID dell'account quando vengono utilizzati nella stessa dichiarazione di policy. Utilizza `aws:SourceArn` se desideri consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi nell'account AWS specificato all'uso tra servizi.

Assicurati che il valore di `aws:SourceArn` sia un ARN per un tipo di risorsa Amazon RDS. Per ulteriori informazioni, consultare Nomi di risorse Amazon (ARNs) in Amazon RDS.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. In alcuni casi, potresti non conoscere l'ARN completo della risorsa o potresti specificare più risorse. In questi casi, utilizza la chiave di contesto delle condizioni globali `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Un esempio è `arn:aws:rds:*:123456789012:*`.

L'esempio seguente mostra il modo in cui puoi utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` delle condizioni globali in Amazon RDS per prevenire il problema “confused deputy”.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Per altri esempi di policy che utilizzano le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount`, consulta le sezioni seguenti:

- Concessione di autorizzazioni per pubblicare le notifiche in un argomento Amazon SNS
- Configurazione dell'accesso a un bucket Simple Storage Service (Amazon S3) (importazione PostgreSQL)
- Configurazione dell'accesso a un bucket Amazon S3 (esportazione PostgreSQL)

Autenticazione del database IAM

Puoi autenticarti nel tuo cluster di DB utilizzando l'autenticazione del database AWS Identity and Access Management (IAM). L'autenticazione del database IAM funziona con Aurora MySQL e Aurora PostgreSQL. Con questo metodo di autenticazione, non devi utilizzare una password quando esegui la connessione al cluster database. Utilizzi invece un token di autenticazione.

Un token di autenticazione è una stringa univoca di caratteri generata da Amazon Aurora su richiesta. I token di autenticazione vengono generati utilizzando AWS Signature Version 4. Ciascun token ha un ciclo di vita di 15 minuti. Non devi archiviare le credenziali dell'utente nel database, perché l'autenticazione è gestita esternamente utilizzando IAM. Puoi anche utilizzare ancora l'autenticazione standard del database. Il token viene utilizzato solo per l'autenticazione e non influisce sulla sessione dopo che è stato stabilito.

L'autenticazione del database IAM fornisce i seguenti vantaggi:

- Il traffico di rete da e verso il database viene crittografato utilizzando Secure Socket Layer (SSL) o Transport Layer Security (TLS). Per ulteriori informazioni sull'uso di SSL/TLS con Amazon Aurora, consulta Utilizzo SSL/TLS per crittografare una connessione a un'
- Puoi usare IAM per gestire in modo centralizzato l'accesso alle risorse del database invece di gestire l'accesso singolarmente in ogni cluster database.
- Per le applicazioni in esecuzione su Amazon EC2, puoi utilizzare credenziali di profilo specifiche per la tua EC2 istanza per accedere al database anziché una password, per una maggiore sicurezza.

In generale, prendi in considerazione l'utilizzo dell'autenticazione del database IAM quando le applicazioni creano meno di 200 connessioni al secondo e non desideri gestire nomi utente e password direttamente nel codice dell'applicazione.

Il driver JDBC di Amazon Web Services (AWS) supporta l'autenticazione del database IAM. Per ulteriori informazioni, consulta AWS IAM Authentication Plugin nel repository di driver GitHub JDBC di Amazon Web Services (AWS).

Il driver Python di Amazon Web Services (AWS) supporta l'autenticazione del database IAM. Per ulteriori informazioni, consulta AWS IAM Authentication Plugin nel repository Python Driver GitHub di Amazon Web Services (AWS).

Esplora i seguenti argomenti per apprendere il processo di impostazione di IAM per l'autenticazione DB:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM
- Connessione al cluster di database tramite l'autenticazione IAM

Disponibilità di regioni e versioni

La disponibilità e il supporto della funzionalità varia a seconda delle versioni specifiche di ciascun motore di database Aurora e in tutte le Regioni AWS. Per ulteriori informazioni sulla disponibilità di versioni e Regioni per l'autenticazione del database Aurora e IAM, consulta Regioni supportate e motori Aurora DB per IAM l'autenticazione del database.

Per Aurora MySQL, tutte le classi di istanza database supportate supportano l'autenticazione del database IAM, ad eccezione di db.t2.small e db.t3.small. Per informazioni sulle classi di istanza database supportate, consulta Motori DB supportati per classi di istanza database.

Supporto per CLI e SDK

L'autenticazione del database IAM è disponibile per AWS CLIe per le seguenti lingue AWS SDKs:

- AWS SDK per .NET
- AWS SDK per C++
- AWS SDK per Go
- AWS SDK per Java
- AWS SDK per JavaScript
- AWS SDK per PHP
- AWS SDK per Python (Boto3)
- AWS SDK per Ruby

Limitazioni per l'autenticazione database IAM

Quando utilizzi l'autenticazione database IAM, tieni presenti le seguenti limitazioni:

- L'autenticazione del database IAM limita le connessioni a 200 connessioni al secondo.

  Le connessioni che utilizzano lo stesso token di autenticazione non vengono limitate. Ti consigliamo di riutilizzare i token di autenticazione quando possibile.

- Attualmente, l'autenticazione database IAM non supporta nessuna delle chiavi di contesto delle condizioni globali.

  Per ulteriori informazioni sulle chiavi di contesto delle condizioni globali, consulta Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente di IAM.
- Per PostgreSQL, se il ruolo IAM (`rds_iam`) viene aggiunto a un utente (incluso l'utente principale RDS), l'autenticazione IAM ha la precedenza sull'autenticazione tramite password, quindi l'utente deve accedere come un utente IAM.
- Per Aurora PostgreSQL, non è possibile utilizzare l'autenticazione IAM per stabilire una connessione di replica.
- Non è possibile utilizzare un record DNS Route 53 personalizzato anziché l'endpoint del cluster di databaseper generare il token di autenticazione.
- CloudWatch e CloudTrail non registrate l'autenticazione IAM. Questi servizi non tengono traccia delle chiamate `generate-db-auth-token` API che autorizzano il ruolo IAM a abilitare la connessione al database.
- L'autenticazione IAM DB richiede risorse di calcolo sul cluster di del database. È necessario disporre di una memoria aggiuntiva compresa tra 300 e 1000 MiB nel database per una connettività affidabile. Per visualizzare la memoria necessaria per il carico di lavoro, confronta la colonna RES per i processi RDS nell'elenco dei processi di Enhanced Monitoring prima e dopo aver abilitato l'autenticazione IAM DB. Consultare Visualizzazione delle metriche del sistema operativo nella console RDS.

  Se utilizzi un'istanza di classe burstable, evita di esaurire la memoria riducendo la memoria utilizzata da altri parametri come i buffer e la cache della stessa quantità.
- Per , non è possibile utilizzare l'autenticazione basata su password per un utente del database configurato con l'autenticazione IAM.
- L'autenticazione IAM DB non è supportata per RDS on Outposts per nessun motore.

Consigli per l'autenticazione del database IAM

Quando si utilizza l'autenticazione del database IAM, è consigliabile procedere come segue:

- Utilizzare l'autenticazione del database IAM quando l'applicazione richiede meno di 200 nuove connessioni di autenticazione del database IAM al secondo.

I motori di database che funzionano con Amazon Aurora non prevedono limitazioni per i tentativi di autenticazione al secondo. Tuttavia, quando utilizzi un'autenticazione database IAM, l'applicazione deve generare un token di autenticazione. L'applicazione usa quindi il token per connettersi al cluster database. Se eccedi il limite massimo di nuove connessioni al secondo, la gestione extra dell'autenticazione database IAM può causare throttling della connessione.

Valuta la possibilità di utilizzare il pool di connessioni nelle applicazioni per mitigare la creazione continua di connessioni. Questo può ridurre il sovraccarico derivante dall'autenticazione DB IAM e consentire alle applicazioni di riutilizzare le connessioni esistenti. In alternativa, per questi casi d'uso considera l'utilizzo di Server proxy per RDS. Per Server proxy per RDS sono previsti costi aggiuntivi. Consulta i prezzi per Server proxy per RDS.

- La dimensione di un token di autenticazione del database IAM dipende da molti fattori, tra cui il numero di tag IAM, le policy di servizio IAM, la lunghezza del nome della risorsa Amazon (ARN) e altre proprietà IAM e del database. La dimensione minima del token è generalmente di circa 1 KB, ma può essere maggiore. Poiché questo token viene utilizzato come password nella stringa di connessione al database tramite l'autenticazione IAM, è necessario assicurarsi che il driver del database (ad esempio ODBC) e/o qualsiasi strumento non limitino o altrimenti tronchino questo token a causa delle sue dimensioni. Un token troncato causa l'esito negativo della convalida dell'autenticazione effettuata dal database e da IAM.
- Se si utilizzano credenziali temporanee durante la creazione di un token di autenticazione del database IAM, le credenziali temporanee devono essere ancora valide quando si utilizza il token di autenticazione del database IAM per effettuare una richiesta di connessione.

Chiavi di contesto relative alle condizioni AWS globali non supportate

L'autenticazione del database IAM non supporta il seguente sottoinsieme di chiavi di contesto delle condizioni AWS globali.

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:UserAgent`
- `aws:VpcSourceIp`

Per ulteriori informazioni, consultare Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente IAM.

Abilitazione e disabilitazione dell'autenticazione database IAM

Per impostazione predefinita, l'autenticazione database IAM è disabilitata nei cluster database. Puoi abilitare o disabilitare l'autenticazione del database IAM utilizzando AWS Management Console AWS CLI, o l'API.

Puoi abilitare l'autenticazione database IAM quando esegui una delle seguenti operazioni:

- Per creare un nuovo cluster di database con l'autenticazione database IAM abilitata, consulta Creazione di un cluster database Amazon Aurora.
- Per modificare un cluster di database per abilitare l'autenticazione database IAM, consulta Modifica di un cluster database Amazon Aurora.
- Per ripristinare un cluster di database da uno snapshot con l'autenticazione del database IAM abilitata, consulta Ripristino da uno snapshot cluster database.
- Per ripristinare un cluster database a un momento specifico con l'autenticazione del database IAM abilitata, consulta Ripristino di un cluster di database a un determinato momento.

Console

Ogni flusso di lavoro di creazione o modifica include una sezione Database authentication (Autenticazione database) in cui puoi abilitare o disabilitare l'autenticazione database IAM. In questa sezione scegli Password and IAM database authentication (Autenticazione password e database IAM) per abilitare l'autenticazione database IAM.

Per abilitare o disabilitare l'autenticazione database IAM per un cluster di database esistente

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione, scegliere Databases (Database).
3. Scegliere il cluster database che si vuole modificare.

> **Note**
>
> Puoi abilitare l'autenticazione IAM solo se tutte le istanze database nel cluster di database sono compatibili con IAM. Controllare i requisiti di compatibilità in Disponibilità di regioni e versioni.

4. Scegliere Modify (Modifica).
5. Nella sezione Autenticazione database seleziona Autenticazione database con password e IAM per abilitare l'autenticazione del database IAM. Scegli Autenticazione password o Autenticazione di password e Kerberos per disabilitare l'autenticazione IAM.
6. Puoi anche scegliere di abilitare la pubblicazione dei log di autenticazione IAM DB su Logs. CloudWatch In Esportazioni di log, scegli l'opzione iam-db-auth-error log. La pubblicazione dei log in CloudWatch Logs consuma spazio di archiviazione e comporta l'addebito di costi per tale archiviazione. Assicurati di eliminare tutti i CloudWatch log che non ti servono più.
7. Scegli Continue (Continua).
8. Per applicare immediatamente le modifiche, scegli Immediately (Immediatamente) nella sezione Scheduling of modifications (Pianificazione delle modifiche).
9. Scegliere Modify cluster (Modifica cluster).

AWS CLI

Per creare un nuovo cluster DB con autenticazione IAM utilizzando il AWS CLI, usa il `create-db-cluster`comando. Specifica l'opzione `--enable-iam-database-authentication`.

Per aggiornare un cluster di database esistente in modo da abilitare o disabilitare l'autenticazione IAM, utilizzare il comando AWS CLI `modify-db-cluster`. Specifica l'opzione `--enable-iam-database-authentication` o `--no-enable-iam-database-authentication`, come appropriato.

> **Note**
>
> Puoi abilitare l'autenticazione IAM solo se tutte le istanze database nel cluster di database sono compatibili con IAM. Controllare i requisiti di compatibilità in Disponibilità di regioni e versioni.

Per impostazione predefinita, Aurora esegue la modifica durante la finestra di manutenzione successiva. Se desideri sostituire ciò e abilitare l'autenticazione database IAM il prima possibile, utilizza il parametro `--apply-immediately`.

Se stai ripristinando un cluster di DB, usa uno dei seguenti AWS CLI comandi:

- `restore-db-cluster-to-point-in-time`

- `restore-db-cluster-from-db-snapshot`

L'impostazione di autenticazione del database IAM corrisponde a quella della snapshot origine. Per modificare questa impostazione, imposta l'opzione `--enable-iam-database-authentication` or `--no-enable-iam-database-authentication`, come appropriato.

API RDS

Per creare una nuova istanza database con autenticazione IAM tramite l'API, utilizzare l'operazione API `CreateDBCluster`. Imposta il parametro `EnableIAMDatabaseAuthentication` su `true`.

Per aggiornare un cluster di database esistente in modo da abilitare l'autenticazione IAM, utilizzare l'operazione API `ModifyDBCluster`. Imposta il parametro `EnableIAMDatabaseAuthentication` su `true` per abilitare l'autenticazione IAM o su `false` per disabilitarla.

> **Note**
>
> Puoi abilitare l'autenticazione IAM solo se tutte le istanze database nel cluster di database sono compatibili con IAM. Controllare i requisiti di compatibilità in Disponibilità di regioni e versioni.

Se ripristini un cluster database, usa una delle operazioni API seguenti:

- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`

L'impostazione di autenticazione del database IAM corrisponde a quella della snapshot origine. Per modificare questa impostazione, imposta il parametro `EnableIAMDatabaseAuthentication` su `true` per abilitare l'autenticazione IAM o su `false` per disabilitarla.

Creazione e utilizzo di una policy IAM per l'accesso al database IAM

Per permettere a un utente o un ruolo di connettersi al cluster database, devi creare una policy IAM. Dopo questa operazione, collega la policy a un set di autorizzazioni o un ruolo.

> **Note**
>
> Per ulteriori informazioni sulle policy IAM, consulta Gestione delle identità e degli accessi per Amazon Aurora.

La policy nell'esempio seguente permette a un utente di connettersi a un cluster database tramite l'autenticazione database IAM.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:1234567890:dbuser:cluster-ABCDEFGHIJKL01234/
db_user"
         ]
      }
   ]
}
```

> **Important**
>
> Un utente con le autorizzazioni di amministratore può accedere ai cluster database senza una policy IAM esplicita. Se vuoi limitare l'accesso come amministratore a cluster di , puoi creare un ruolo IAM con le autorizzazioni appropriate, con privilegi minori e assegnarlo all'amministratore.

> **Note**
>
> Non confondere il prefisso `rds-db:` con altri prefissi di operazione API RDS che iniziano con `rds:`. Utilizzi il prefisso `rds-db:` e l'operazione `rds-db:connect` solo per l'autenticazione database IAM. Non sono validi in altri contesti.

La policy di esempio include una singola istruzione con i seguenti elementi:

- `Effect` – Specifica `Allow` per concedere l'accesso al cluster database. Se non si concede esplicitamente l'accesso, questo viene rifiutato per impostazione predefinita.
- `Action` – Specifica `rds-db:connect` per consentire le connessioni al cluster database.
- `Resource` – Specifica un Amazon Resource Name (ARN) che descrive un account database in un cluster database. Di seguito è riportato il formato ARN.

```
arn:aws:rds-db:region:account-id:dbuser:DbClusterResourceId/db-user-name
```

  In questo formato, sostituisci quanto segue:

  - *`region`*è la AWS regione per il cluster di DB. Nella politica di esempio, la AWS regione è`us-east-2`.
  - *`account-id`*è il numero di AWS account per il cluster di DB. Nella policy di esempio, il numero di account è `1234567890`. L'utente deve essere nello stesso account dell'account del cluster di database.

    Per eseguire l'accesso multi-account, crea un ruolo IAM con la policy mostrata in precedenza nell'account per il cluster di database e consenti all'altro account di assumere il ruolo.
  - *`DbClusterResourceId`* è l'identificatore del cluster database. Questo identificatore è unico per una AWS regione e non cambia mai. Nella policy di esempio, l'identificatore è `cluster-ABCDEFGHIJKL01234`.

    Per trovare un ID di risorsa del AWS Management Console cluster di DB in Amazon Aurora, scegli il cluster di istanze per visualizzarne i dettagli. Quindi seleziona la scheda Configuration (Configurazione). Resource ID (ID risorsa) è visualizzato nella sezione Configuration (Configurazione).

    In alternativa, puoi utilizzare il AWS CLI comando per elencare gli identificatori e le risorse IDs per tutto il cluster di DB nella AWS regione corrente, come illustrato di seguito.

```
aws rds describe-db-clusters --query "DBClusters[*].
[DBClusterIdentifier,DbClusterResourceId]"
```

> **Note**
>
> Se si sta effettuando la connessione a un database tramite proxy RDS, specificare l'ID risorsa proxy, ad esempio `prx-ABCDEFGHIJKL01234`. Per informazioni sull'utilizzo dell'autenticazione del database IAM con proxy RDS, vedere Connessione a un proxy mediante autenticazione IAM.

- *`db-user-name`* è il nome dell'account database da associare all'autenticazione IAM. Nella policy di esempio, l'account database è `db_user`.

È possibile crearne altri ARNs per supportare vari modelli di accesso. La policy seguente permette l'accesso a due diversi account database in un cluster database.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
mary_roe"
         ]
      }
   ]
}
```

La seguente politica utilizza il carattere «*» per abbinare tutte le DB, i cluster e gli account di database per un account e una regione particolari AWS . AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:1234567890:dbuser:*/*"
            ]
        }
    ]
}
```

La seguente politica corrisponde a tutti i cluster di DB per un account e una regione particolari AWS . AWS Tuttavia, la policy concede l'accesso solo a cluster database che hanno un account database `jane_doe`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

L'utente o il ruolo ha accesso solo a quei database ai quali l'utente ha accesso. Supponiamo, ad esempio, che il cluster database abbia un database denominato dev e un altro database denominato test. Se l'utente del database `jane_doe` ha accesso solo a dev, anche qualsiasi utente o ruolo che accede al cluster database con l'utente `jane_doe` ha accesso solo a dev. Questa restrizione dell'accesso è anche valida per altri oggetti database, come tabelle, visualizzazioni e così via.

Un amministratore deve creare policy IAM che concedono alle entità l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore deve quindi collegare queste policy ai set di autorizzazioni o ai ruoli che richiedono tali autorizzazioni. Per esempi di policy, consulta .

Collegamento di una policy IAM a un set di autorizzazioni o un ruolo

Dopo aver creato una policy IAM per consentire l'autenticazione del database, devi collegare la policy a un set di autorizzazioni o un ruolo. Per un tutorial su questo argomento, consulta Creare e collegare la tua prima policy gestita dal cliente nella Guida per l'utente IAM.

Durante il tutorial, puoi utilizzare uno degli esempi di policy indicati in questa sezione come punto di partenza e personalizzarli in base alle tue esigenze. Al termine del tutorial, avrai un set di autorizzazioni con una policy collegata che può utilizzare l'operazione `rds-db:connect`.

> **Note**
>
> Puoi mappare più set di autorizzazioni o ruoli allo stesso account utente del database. Ad esempio, supponiamo che la policy IAM abbia specificato la seguente risorsa ARN.
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> Se colleghi la policy agli utenti Jane, Bob e Diego, ciascuno di quegli utenti potrà connettersi al cluster di database specificato utilizzando l'account di database `jane_doe`.

Creazione di un account database tramite l'autenticazione IAM

Con l'autenticazione database IAM, non devi assegnare password di database agli account utente che crei. Se rimuovi un utente mappato a un account database, devi anche rimuovere l'account database con l'istruzione `DROP USER`.

> **Note**
>
> Il nome utente utilizzato per l'autenticazione IAM deve avere lo stesso formato maiuscolo/minuscolo del nome utente nel database.

Argomenti

- Utilizzo dell'autenticazione IAM con Aurora MySQL
- Utilizzo dell'autenticazione IAM con Aurora PostgreSQL

Utilizzo dell'autenticazione IAM con Aurora MySQL

Con per autenticare gli utenti. Connettiti al cluster database come utente master o altro utente che può creare utenti e concedere privilegi. Dopo la connessione, immetti l'istruzione `CREATE USER`, come mostrato nell'esempio seguente.

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

La clausola `IDENTIFIED WITH` permette a Aurora MySQL di utilizzare `AWSAuthenticationPlugin` per autenticare l'account database (`jane_doe`). La clausola `AS 'RDS'` fa riferimento al metodo di autenticazione. Assicurarsi che il nome utente del database specificato sia lo stesso di una risorsa nella policy IAM per l'accesso al database IAM. Per ulteriori informazioni, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM.

> **Note**
>
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
> Per correggere questo errore, assicurati di usare una configurazione supportata e di aver abilitato l'autenticazione database IAM nel cluster database. Per ulteriori informazioni, consulta Disponibilità di regioni e versioni e Abilitazione e disabilitazione dell'autenticazione database IAM.

Dopo aver creato un account utilizzando `AWSAuthenticationPlugin`, lo gestisci nello stesso modo di altri account database. Ad esempio, puoi modificare i privilegi di account con le istruzioni `GRANT` e `REVOKE` o modificare vari attributi di account con l'istruzione `ALTER USER`.

Il traffico di rete del database viene crittografato utilizzando SSL/TLS quando si utilizza IAM. Per consentire le connessioni SSL, modifica l'account utente con il comando seguente.

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

Utilizzo dell'autenticazione IAM con Aurora PostgreSQL

Per utilizzare l'autenticazione IAM con Aurora PostgreSQL, connettiti al cluster database come utente master o altro utente che può creare utenti e concedere privilegi. Dopo la connessione, crea gli utenti di database e autorizza il ruolo `rds_iam`, come mostrato nell'esempio seguente.

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

Assicurarsi che il nome utente del database specificato sia lo stesso di una risorsa nella policy IAM per l'accesso al database IAM. Per ulteriori informazioni, consulta Creazione e utilizzo di una policy IAM per l'accesso al database IAM. È necessario concedere il `rds_iam` ruolo per utilizzare l'autenticazione IAM. Puoi anche utilizzare le appartenenze annidate o le concessioni indirette del ruolo.

Tieni presente che un utente del database PostgreSQL può utilizzare l'autenticazione IAM o Kerberos ma non entrambe, quindi questo utente non può avere anche il ruolo `rds_ad`. Questo vale anche per le appartenenze nidificate. Per ulteriori informazioni, consulta Fase 7: creazione di utenti PostgreSQL per i principali Kerberos .

Connessione al cluster di database tramite l'autenticazione IAM

Con l'autenticazione database IAM devi usare un token di autenticazione per la connessione al cluster di database. Un token di autenticazione è una stringa unica di caratteri che utilizzi invece di una password. Trascorsi 15 minuti dalla sua creazione, un token di autenticazione scade. Se cerchi di eseguire la connessione utilizzando un token scaduto la richiesta di connessione viene negata.

Ogni token di autenticazione deve essere accompagnato da una firma valida, utilizzando AWS Signature Version 4. (Per ulteriori informazioni, vedere Processo di firma Signature Version 4 in Riferimenti generali di AWS.) AWS CLI E un AWS SDK, come AWS SDK per Java o AWS SDK per Python (Boto3), possono firmare automaticamente ogni token creato.

Puoi utilizzare un token di autenticazione quando ti connetti ad Amazon Aurora da AWS un altro servizio, ad esempio. AWS Lambda Utilizzando un token, eviti di inserire una password nel codice. In alternativa, puoi utilizzare un AWS SDK per creare e firmare programmaticamente un token di autenticazione.

Quando hai un token di autenticazione IAM firmato, puoi connetterti a un cluster di database Aurora. Di seguito, puoi scoprire come eseguire questa operazione utilizzando uno strumento da riga di comando o un AWS SDK, come o. AWS SDK per Java AWS SDK per Python (Boto3)

Per ulteriori informazioni, consulta il seguente post sul blog:

- Uso dell'autenticazione IAM per la connessione con SQL Workbench/J a Aurora MySQL o Amazon RDS for MySQL.

- Utilizzo dell'autenticazione IAM per connettersi con pgAdmin Amazon Aurora PostgreSQL o Amazon RDS for PostgreSQL

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Argomenti

- Connessione al cluster di DB utilizzando l'autenticazione IAM con i AWS driver
- Connessione al cluster di DB utilizzando l'autenticazione IAM dalla riga di comando: AWS CLI e il client mysql
- Connessione al cluster di database tramite Autenticazione IAM dalla riga di comando: AWS CLI e client psql
- Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per .NET
- Connessione al cluster di DB utilizzando IAM l'autenticazione e il AWS SDK per Go
- Connessione al cluster di DB utilizzando l'autenticazione IAM e AWS SDK per Java
- Connessione al cluster di DB utilizzando IAM l'autenticazione e il AWS SDK per Python (Boto3)

Connessione al cluster di DB utilizzando l'autenticazione IAM con i AWS driver

La AWS suite di driver è stata progettata per fornire supporto per tempi di switchover e failover più rapidi e l'autenticazione con AWS Secrets Manager, AWS Identity and Access Management (IAM) e Federated Identity. I AWS driver si basano sul monitoraggio dello stato dell' e sulla conoscenza della topologia dell' del cluster per determinare il nuovo writer. Questo approccio riduce i tempi di switchover e failover a secondi a una cifra, rispetto alle decine di secondi dei driver open source.

Per ulteriori informazioni sui AWS driver, consulta il driver di lingua corrispondente per il cluster Aurora MySQL o Aurora PostgreSQL DB.

Connessione al cluster di DB utilizzando l'autenticazione IAM dalla riga di comando: AWS CLI e il client mysql

Puoi connetterti dalla riga di comando a un cluster Aurora con AWS CLI lo strumento da riga di comando `mysql` and come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

> **Note**
>
> Per informazioni sulla connessione al database tramite SQL Workbench/J con autenticazione IAM, consulta il post del blog Utilizzo dell'autenticazione IAM per connettersi con SQL Workbench/J a Aurora MySQL o Amazon RDS for MySQL.

Argomenti

- Generazione di un token di autenticazione IAM
- Connessione a un cluster database

Generazione di un token di autenticazione IAM

L'esempio seguente mostra come ottenere un token di autenticazione firmato utilizzando AWS CLI.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
   --port 3306 \
   --region us-west-2 \
   --username jane_doe
```

Nell'esempio, i parametri sono come segue:

- `--hostname` – Nome host del cluster database cui vuoi accedere.
- `--port` – Numero di porta usato per la connessione al dell'istanza database
- `--region`— La AWS regione in cui è in esecuzione il cluster di DB
- `--username` – L'account database cui vuoi accedere.

I primi caratteri del token hanno il seguente aspetto.

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Connessione a un cluster database

Il formato generale per la connessione è visualizzato di seguito.

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

I parametri sono i seguenti:

- `--host` – Nome host del cluster database cui vuoi accedere.
- `--port` – Numero di porta usato per la connessione al dell'istanza database
- `--ssl-ca` – Il percorso completo del file del certificato SSL che contiene la chiave pubblica

  Per ulteriori informazioni, consulta Connessioni TLS ai cluster Aurora MySQL DB.

  Per scaricare un certificato SSL consulta Utilizzo SSL/TLS per crittografare una connessione a un'
- `--enable-cleartext-plugin` – Valore che specifica che per la connessione deve essere usato `AWSAuthenticationPlugin`.

  Se si utilizza un client MariaDB, l'opzione `--enable-cleartext-plugin` non è richiesta.

- `--user` – L'account database cui vuoi accedere.
- `--password` – Token di autenticazione IAM firmato.

Il token di autenticazione consiste in diverse centinaia di caratteri. Può essere macchinoso nella riga di comando. Una soluzione è di salvare il token in una variabile di ambiente e utilizzare quella variabile durante la connessione. L'esempio seguente mostra un modo per eseguire questa soluzione. Nell'esempio,*/sample_dir/* è il percorso completo del file del certificato SSL che contiene la chiave pubblica.

```
RDSHOST="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

Quando si esegue la connessione utilizzando `AWSAuthenticationPlugin`, la connessione viene protetta utilizzando SSL. Per verificare ciò, digita quanto segue al prompt del comando `mysql>`.

```
show status like 'Ssl%';
```

Le righe seguenti nell'output mostrano più dettagli.

```
+---------------+-------------+
| Variable_name | Value

                                                                    |
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                    |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                    |
| ...           | ...
+-----------------------------+
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Connessione al cluster di database tramite Autenticazione IAM dalla riga di comando: AWS CLI e client psql

Puoi eseguire la connessione dalla riga di comando a un cluster database Aurora PostgreSQL con AWS CLI e lo strumento a riga di comando psql come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

> **Note**
>
> Per informazioni sulla connessione al database tramite pgAdmin con autenticazione IAM, consulta il post sul blog Utilizzo dell'autenticazione IAM per connettersi con PGadmin Amazon Aurora PostgreSQL o Amazon RDS for PostgreSQL.

Argomenti

- Generazione di un token di autenticazione IAM
- Connessione a un cluster Aurora PostgreSQL

Generazione di un token di autenticazione IAM

Il token di autenticazione è costituito da diverse centinaia di caratteri, quindi può essere complesso nella riga di comando. Una soluzione è di salvare il token in una variabile di ambiente e utilizzare quella variabile durante la connessione. Il seguente esempio mostra come usare l'AWS CLI per ottenere un token di autenticazione firmato utilizzando il comando `generate-db-auth-token` e archiviarlo in una variabile di ambiente `PGPASSWORD`.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
```

```
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

Nell'esempio, i parametri per il comando `generate-db-auth-token` sono i seguenti:

- `--hostname` – Nome host del cluster (endpoint del cluster) di database cui desideri accedere.
- `--port` – Numero di porta usato per la connessione al dell'istanza database
- `--region`: la regione AWS in cui è in esecuzione il cluster database
- `--username` – L'account database cui vuoi accedere.

I primi caratteri del token generato hanno il seguente aspetto.

```
mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Connessione a un cluster Aurora PostgreSQL

Di seguito è mostrato il formato generale per l'utilizzo di psql per la connessione.

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

I parametri sono i seguenti:

- `host` – Nome host del cluster (endpoint del cluster) di database cui desideri accedere.
- `port` – Numero di porta usato per la connessione al dell'istanza database
- `sslmode` – Modalità SSL da utilizzare.

Quando si utilizza `sslmode=verify-full`, la connessione SSL verifica l'endpoint del cluster database rispetto a quello nel certificato SSL.

- `sslrootcert` – Il percorso completo del file del certificato SSL che contiene la chiave pubblica

  Per ulteriori informazioni, consulta Sicurezza dei dati Aurora PostgreSQL con SSL/TLS.

  Per scaricare un certificato SSL consulta Utilizzo SSL/TLS per crittografare una connessione a un'
- `dbname` – Database a cui accedere.
- `user` – L'account database cui vuoi accedere.
- `password` – Token di autenticazione IAM firmato.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

L'esempio seguente mostra l'utilizzo di psql per la connessione. Nell'esempio, psql utilizza la variabile d'ambiente `RDSHOST` per l'host e la variabile d'ambiente `PGPASSWORD` per il token generato. Inoltre, */sample_dir/* è il percorso completo al file del certificato SSL che contiene la chiave pubblica.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"

psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Connessione al dell'istanza database tramite Autenticazione IAM e AWS SDK per .NET

Puoi connetterti a un cluster di database Aurora MySQL o Aurora PostgreSQL con la AWS SDK per .NET come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Esempi

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Per eseguire questo codice di esempio è necessario AWS SDK per .NET, disponibile sul sito AWS. I pacchetti `AWSSDK.CORE` e `AWSSDK.RDS` sono obbligatori. Per connetterti a un cluster database, utilizza il connettore di database .NET per il motore di database, ad esempio MySQLConnector per MariaDB o MySQL o Npgsql per PostgreSQL.

Questo codice si connette a un cluster di database Aurora MySQL. Modifica i valori delle variabili seguenti in base alle esigenze.

- `server`: l'endpoint del cluster database cui vuoi accedere
- `user` – L'account database cui vuoi accedere.
- `database` – Database a cui accedere.
- `port` – Numero di porta usato per la connessione al dell'istanza database
- `SslMode` – Modalità SSL da utilizzare.

  Quando si utilizza `SslMode=Required`, la connessione SSL verifica l'endpoint del cluster database rispetto a quello nel certificato SSL.
- `SslCa` - Percorso completo del certificato SSL per Amazon Aurora

  Per scaricare un certificato, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new
 MySqlConnection($"server=mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);

      // Execute a query
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
```

```
      conn.Close();
    }
  }
}
```

Questo codice si connette a un cluster di database Aurora PostgreSQL.

Modifica i valori delle variabili seguenti in base alle esigenze.

- `Server`: l'endpoint del cluster database cui vuoi accedere
- `User ID` – L'account database cui vuoi accedere.
- `Database` – Database a cui accedere.
- `Port` – Numero di porta usato per la connessione al dell'istanza database
- `SSL Mode` – Modalità SSL da utilizzare.

  Quando si utilizza `SSL Mode=Required`, la connessione SSL verifica l'endpoint del cluster database rispetto a quello nel certificato SSL.
- `Root Certificate` - Percorso completo del certificato SSL per Amazon Aurora

  Per scaricare un certificato, consultare Utilizzo SSL/TLS per crittografare una connessione a un'.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

```
using System;
using Npgsql;
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;User Id=jane_doe;Password={pwd};Database=mydb;SSL
 Mode=Require;Root Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Connessione al cluster di DB utilizzando IAM l'autenticazione e il AWS SDK per Go

È possibile connettersi a un cluster Aurora My o Aurora con SQL la procedura descritta di seguito SQL. AWS SDK per Go

Prerequisiti

IAM

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Esempi

Per eseguire questi esempi di codice, è necessario il AWS SDK per Gofile, trovato sul AWS sito.

Modifica i valori delle variabili seguenti in base alle esigenze.

- `dbName` – Database a cui accedere.
- `dbUser` – L'account database cui vuoi accedere.
- `dbHost`: l'endpoint del cluster database cui vuoi accedere

> **Note**
>
> Non è possibile utilizzare un DNS record Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster di DB per generare il token di autenticazione.

- `dbPort` – Numero di porta usato per la connessione al dell'istanza database
- `region`— La AWS regione in cui è in esecuzione il cluster di DB

Inoltre, assicurarsi che le librerie importate nel codice di esempio esistano nel sistema.

> **Important**
>
> Negli esempi riportati in questa sezione viene utilizzato il codice seguente per fornire credenziali che accedono a un database da un ambiente locale:
> `creds := credentials.NewEnvCredentials()`
> Se accedi a un database da un AWS servizio, come Amazon EC2 o AmazonECS, puoi sostituire il codice con il seguente codice:
> `sess := session.Must(session.NewSession())`
> `creds := sess.Config.Credentials`
> Se si apporta questa modifica, assicurarsi di aggiungere la seguente importazione:
> `"github.com/aws/aws-sdk-go/aws/session"`

Argomenti

- Connessione tramite IAM autenticazione e AWS SDK per Go V2
- Connessione tramite IAM autenticazione e V1. AWS SDK per Go

Connessione tramite IAM autenticazione e AWS SDK per Go V2

È possibile connettersi a un cluster di DB utilizzando IAM l'autenticazione e la AWS SDK per Go V2.

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Questo codice si connette a un cluster Aurora My SQL DB.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
```

```
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Questo codice si connette a un cluster Aurora SQL Postgre DB.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
```

```
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Connessione tramite IAM autenticazione e V1. AWS SDK per Go

È possibile connettersi a un cluster di DB utilizzando IAM l'autenticazione e la AWS SDK per Go V1

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Questo codice si connette a un cluster Aurora My SQL DB.

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)
```

```
func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Questo codice si connette a un cluster Aurora SQL Postgre DB.

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)
```

```
func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Connessione al cluster di DB utilizzando l'autenticazione IAM e AWS SDK per Java

È possibile connettersi a un cluster come descritto di seguito. AWS SDK per Java

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione alcluster di DB utilizzando l'autenticazione IAM:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM

- Creazione di un account database tramite l'autenticazione IAM
- Configurare l' AWS SDK per Java

Per esempi su come utilizzare l'SDK per Java 2.x, consulta Esempi di Amazon RDS con SDK for Java 2.x. Puoi anche usare Advanced JDBC Wrapper, consulta la AWS documentazione di Advanced JDBC Wrapper.AWS

Argomenti

- Generazione di un token di autenticazione IAM
- Creazione manuale di un token di autenticazione IAM
- Connessione a un cluster database

Generazione di un token di autenticazione IAM

Se state scrivendo programmi utilizzando il AWS SDK per Java, potete ottenere un token di autenticazione firmato utilizzando la classe. `RdsIamAuthTokenGenerator` L'utilizzo di questa classe richiede l'immissione di AWS credenziali. A tale scopo, create un'istanza della `DefaultAWSCredentialsProviderChain` classe. `DefaultAWSCredentialsProviderChain`utilizza la prima chiave di AWS accesso e la chiave segreta che trova nella catena di provider di credenziali predefinita. Per ulteriori informazioni sulle chiavi di accesso AWS , consulta Gestione delle chiavi di accesso per gli utenti IAM.

> **Note**
>
> Non è possibile utilizzare un record DNS Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster database per generare il token di autenticazione.

Dopo aver creato un'istanza di `RdsIamAuthTokenGenerator`, puoi chiamare il metodo `getAuthToken` per ottenere un token firmato. Specifica la regione AWS , il nome host, il numero di porta e il nome utente. Il seguente esempio di codice dimostra come eseguire questa operazione.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
```

```
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {

    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());

     return authToken;
    }

}
```

Creazione manuale di un token di autenticazione IAM

In Java, il modo più semplice di generare un token di autenticazione è di utilizzare `RdsIamAuthTokenGenerator`. Questa classe crea un token di autenticazione per te, quindi lo firma utilizzando la versione 4 AWS della firma. Per ulteriori informazioni, consulta la pagina relativa al processo di firma Signature Version 4 nella Riferimenti generali di AWS.

Tuttavia, puoi anche creare e firmare un token di autenticazione manualmente, come visualizzato nel seguente esempio di codice.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
 SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
```

```
       DefaultAWSCredentialsProviderChain creds = new
DefaultAWSCredentialsProviderChain();
    String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
    String awsSecretKey = creds.getCredentials().getAWSSecretKey();
       String expiryMinutes = "900";

       System.out.println("Step 1:  Create a canonical request:");
       String canonicalString = createCanonicalString(username, awsAccessKey, date,
dateTimeStamp, region, expiryMinutes, instanceName, port);
       System.out.println(canonicalString);
       System.out.println();

       System.out.println("Step 2:  Create a string to sign:");
       String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
awsAccessKey, date, region);
       System.out.println(stringToSign);
       System.out.println();

       System.out.println("Step 3:  Calculate the signature:");
       String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
       System.out.println(signature);
       System.out.println();

       System.out.println("Step 4:  Add the signing info to the request");

       System.out.println(appendSignature(signature));
       System.out.println();

   }

   //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
should be in format YYYYMMDDTHHMMSSZ
   public static String createCanonicalString(String user, String accessKey, String
date, String dateTime, String region, String expiryPeriod, String hostName, String
port) throws Exception {
       canonicalQueryParameters.put("Action", action);
       canonicalQueryParameters.put("DBUser", user);
       canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
       canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
       canonicalQueryParameters.put("X-Amz-Date", dateTime);
       canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
       canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
```

```
        String canonicalQueryString = "";
        while(!canonicalQueryParameters.isEmpty()) {
            String currentQueryParameter = canonicalQueryParameters.firstKey();
            String currentQueryParameterValue =
 canonicalQueryParameters.remove(currentQueryParameter);
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
 currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));

    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
```

```
                        SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                    String dateStamp, String regionName, String
serviceName) {
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                        SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
```

```
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
        }
    }
}
```

Connessione a un cluster database

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster eseguendo Aurora MySQL.

Per eseguire questo esempio di codice, è necessario il AWS SDK per Javafile, trovato sul AWS sito. Inoltre, hai bisogno di quanto segue:

- MySQL Connector/J. Questo esempio di codice è stato testato con `mysql-connector-java-5.1.33-bin.jar`.
- Un certificato intermedio per Amazon Aurora specifico per una regione. AWS Per ulteriori informazioni, consult Utilizzo SSL/TLS per crittografare una connessione a un'.) Durante il runtime, il loader della classe cerca un certificato nella stessa directory di questo esempio di codice Java, per permettere al loader della classe di trovarlo.
- Modifica i valori delle variabili seguenti in base alle esigenze.
  - `RDS_INSTANCE_HOSTNAME`: il nome host del cluster database cui vuoi accedere.
  - `RDS_INSTANCE_PORT`: il numero di porta usato per la connessione al cluster database PostgreSQL.
  - `REGION_NAME`— La AWS regione in cui è in esecuzione il cluster di DB.
  - `DB_USER`: l'account database cui vuoi accedere.
  - `SSL_CERTIFICATE`— Un certificato SSL per Amazon Aurora specifico per una regione. AWS

    Per scaricare un certificato per la regione AWS , consulta Utilizzo SSL/TLS per crittografare una connessione a un'. Inserisci il certificato SSL nella stessa directory di questo file di programma Java, per permettere al loader di classe di trovare il certificato durante il runtime.

Questo esempio di codice ottiene AWS le credenziali dalla catena di provider di credenziali predefinita.

> **Note**
>
> Specifica una password per `DEFAULT_KEY_STORE_PASSWORD` diversa da quella mostrata qui come best practice di sicurezza.

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
```

```
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
 Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
```

```
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
     * as the password. It also specifies that SSL verification is required.
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
     * @return
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
 AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
 AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

 .hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
 and password:
```

```
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
DEFAULT_KEY_STORE_PASSWORD);
    }

    /**
     * This method returns the path of the Key Store File needed for the SSL
verification during the IAM Database Authentication to
     * the db instance.
     * @return
     * @throws Exception
     */
    private static String createKeyStoreFile() throws Exception {
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
Exception {
```

```
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
 KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }

    /**
     * This method clears the SSL properties.
     * @throws Exception
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Connessione al cluster di DB utilizzando IAM l'autenticazione e il AWS SDK per Python (Boto3)

È possibile connettersi a un cluster Aurora Postgre DB con SQL AWS SDK per Python (Boto3) come descritto di seguito.

Prerequisiti

Di seguito sono riportati i prerequisiti per la connessione al cluster di DB tramite IAM l'autenticazione:

- Abilitazione e disabilitazione dell'autenticazione database IAM
- Creazione e utilizzo di una policy IAM per l'accesso al database IAM
- Creazione di un account database tramite l'autenticazione IAM

Inoltre, assicurarsi che le librerie importate nel codice di esempio esistano nel sistema.

Esempi

Gli esempi di codice utilizzano i profili per le credenziali condivise. Per informazioni sulla specificazione delle credenziali, consulta Credenziali nella AWS SDK per Python (Boto3) documentazione.

Il seguente esempio di codice mostra come generare un token di autenticazione e utilizzarlo per eseguire la connessione a un cluster del database.

Per eseguire questo esempio di codice, è necessario AWS SDK per Python (Boto3), trovato su AWS sito.

Modifica i valori delle variabili seguenti in base alle esigenze.

- `ENDPOINT`: l'endpoint del cluster di database cui vuoi accedere
- `PORT` – Numero di porta usato per la connessione al dell'istanza database
- `USER` – L'account database cui vuoi accedere.
- `REGION`— Il AWS Regione in cui è in esecuzione il cluster di DB
- `DBNAME` – Database a cui accedere.
- `SSLCERTIFICATE`— Il percorso completo del SSL certificato per Aurora

  Per`ssl_ca`, specifica un SSL certificato. Per scaricare un SSL certificato, consultaUtilizzo SSL/TLS per crittografare una connessione a un'.

> **Note**
>
> Non è possibile utilizzare un DNS record Route 53 personalizzato o un endpoint personalizzato Aurora anziché l'endpoint del cluster di DB per generare il token di autenticazione.

Questo codice si connette a un cluster Aurora My SQL DB.

Prima di eseguire questo codice, installa il PyMy SQL driver seguendo le istruzioni nel Python Package Index.

```
import pymysql
import sys
import boto3
```

```
import os

ENDPOINT="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Questo codice si connette a un cluster Aurora SQL Postgre DB.

Prima di eseguire questo codice, installa`psycopg2`seguendo le istruzioni in Documentazione di Psycopg.

```
import psycopg2
import sys
import boto3
import os

ENDPOINT="postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
```

```
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Se desideri connetterti a un cluster di database tramite un proxy, consulta Connessione a un proxy mediante autenticazione IAM.

Risoluzione dei problemi per l'autenticazione IAM DB

Di seguito, puoi trovare idee per la risoluzione di alcuni problemi comuni di autenticazione IAM DB e informazioni su CloudWatch log e metriche per l'autenticazione IAM DB.

Esportazione dei log degli errori di autenticazione IAM DB in Logs CloudWatch

I log degli errori di autenticazione IAM DB sono archiviati sull'host del database e puoi esportare questi log nell'account Logs. CloudWatch Utilizza i log e i metodi di correzione in questa pagina per risolvere i problemi di autenticazione IAM DB.

Puoi abilitare le esportazioni dei log in CloudWatch Logs dalla console e dall'API RDS. AWS CLI Per le istruzioni sulla console, consulta. Pubblicazione dei log del database su Amazon CloudWatch Logs

Per esportare i log degli errori di autenticazione IAM DB in CloudWatch Logs durante la creazione di un cluster DB da AWS CLI, usa il seguente comando:

```
aws rds create-db-cluster --db-cluster-identifier mydbinstance \
```

```
--region us-east-1 \
--engine postgres \
--engine-version 16 \
--master-username master \
--master-user-password password \
--publicly-accessible \
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

Per esportare i log degli errori di autenticazione IAM DB in CloudWatch Logs quando modifichi un cluster DB di da AWS CLI, usa il seguente comando:

```
aws rds modify-db-cluster --db-instance-identifier mydbcluster \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

Per verificare se il cluster DB dell'istanza DB sta esportando i log di autenticazione IAM DB in CloudWatch Logs, controlla se il `EnabledCloudwatchLogsExports` parametro è impostato su `iam-db-auth-error` nell'output del comando. `describe-db-instances`

```
aws rds describe-db-cluster --region us-east-1 --db-cluster-identifier mydbcluster
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

Metriche di autenticazione IAM DB CloudWatch

Amazon Aurora fornisce parametri quasi in tempo reale sull'autenticazione IAM DB al tuo account Amazon. CloudWatch La tabella seguente elenca i parametri di autenticazione IAM DB disponibili utilizzando: CloudWatch

Parametro	Descrizione
`IamDbAuthConnectionRequests`	Numero totale di richieste di connessione effettuate con l'autenticazione IAM DB.

Parametro	Descrizione
`IamDbAuthConnectionSuccess`	Numero totale di richieste di autenticazione IAM DB riuscite.
`IamDbAuthConnectionFailure`	Numero totale di richieste di autenticazione IAM DB non riuscite.
`IamDbAuthConnectionFailureInvalidToken`	Numero totale di richieste di autenticazione IAM DB non riuscite a causa di un token non valido.
`IamDbAuthConnectionFailureInsufficientPermissions`	Numero totale di richieste di autenticazione IAM DB non riuscite a causa di policy o autorizzazioni errate.
`IamDbAuthConnectionFailureThrottling`	Numero totale di richieste di autenticazione IAM DB non riuscite a causa della limitazione dell'autenticazione IAM DB.
`IamDbAuthConnectionFailureServerError`	Numero totale di richieste di autenticazione IAM DB non riuscite a causa di un errore interno del server nella funzionalità di autenticazione IAM DB.

Problemi e soluzioni comuni

È possibile riscontrare i seguenti problemi quando si utilizza l'autenticazione IAM DB. Utilizza i passaggi di riparazione indicati nella tabella per risolvere i problemi:

Errore	Metrica/e	Causa	Soluzione

`[ERROR] Failed to authenticate the connection request for user` *`db_user`* `because the provided token`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectio`
		Il token di autenticazione IAM DB nella richiesta di connessione non è un token SigV4A valido o non è formattato correttamente.	Controlla la tua strategia di generazione di token nella tua applicazione. In alcuni casi, assicuratevi di passare il token con una formattazione

Errore	Metrica/e	Causa	Soluzione
`is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidToken)`	`nFailureInvalidToken`		valida. Il tronco del token (o la formattazione errata della stringa) renderà il token non valido.

`[ERROR] Failed to authenticate the connection request for user` *`db_user`* `because the token age is longer than 15 minutes. (Status Code: 400, Error Code:ExpiredToken)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInvalidToken`
		Il token di autenticazione IAM DB è scaduto. I token sono validi solo per 15 minuti.	Controlla la logica di memorizzazione nella cache dei token e/o di riutilizzo dei token nell'applicazione. Non dovresti riutilizzare token più vecchi di 15 minuti.

Errore	Metrica/e	Causa	Soluzione

`[ERROR] Failed to authorize the connection request for user` *`db_user`* `because the IAM policy assumed by the caller 'arn:aws:sts::123456789012:assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform `rds-db:connect` on the DB instance. (Status Code: 403, Error Code:NotAuthorized)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureInsufficientPermissions`
		Questo errore potrebbe essere dovuto ai seguenti fattori:
		• La policy IAM assunta dall'applicazione non autorizza l'azione. `rds-db:connect`
		• Stai assumendo il ruolo/la politica errati per la connessione *`db_user`* al database.
		• Stai assumendo la politica corretta per*`db_user`*, ma non ti stai connettendo al database corretto.
			Verifica il ruolo e/o la policy IAM che stai assumendo nella tua applicazione. Assicurati di assumere la stessa politica per generare il token utilizzata per la connessione al DB.

Errore	Metrica/e	Causa	Soluzione

`[ERROR] Failed to authorize the connection request for user db_user due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Stai effettuando troppe richieste di connessione al tuo DB in un breve lasso di tempo. Il limite di limitazione dell'autenticazione IAM DB è di 200 connessioni al secondo.	Riduci la velocità di creazione di nuove connessioni con l'autenticazione IAM. Prendi in considerazione l'implementazione del pool di connessioni utilizzando RDS Proxy per riutilizzare le connessioni stabilite nell'applicazione.

`[ERROR] Failed to authorize the connection request for user db_user due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)`	`IamDbAuthConnectionFailure`
	
	`IamDbAuthConnectionFailureThrottling`
		Si è verificato un errore interno durante l'autorizzazione della connessione DB con l'autenticazione IAM DB.	Contattateci per https://aws.amazon.com/premiumsupport/ indagare sul problema.

Risoluzione dei problemi di identità e accesso in Amazon Aurora

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Aurora e IAM.

Argomenti

- Non sono autorizzato a eseguire un'operazione in Aurora
- Non sono autorizzato a eseguire: iam:PassRole
- Voglio consentire a persone esterne al mio account AWS di accedere alle mie risorse Aurora

Non sono autorizzato a eseguire un'operazione in Aurora

Se la AWS Management Console indica che non sei autorizzato a eseguire un'operazione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso.

L'errore di esempio seguente si verifica quando l'utente `mateojackson` cerca di utilizzare la console per visualizzare i dettagli relativi a un *`widget`*, ma non dispone di autorizzazioni `rds:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le sue policy per poter accedere alla risorsa *`my-example-widget`* utilizzando l'operazione `rds:`*`GetWidget`*.

Non sono autorizzato a eseguire: iam:PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, devi contattare il tuo amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso. Richiedi a tale persona di aggiornare le tue policy per poter passare un ruolo a Aurora.

Alcuni servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo del servizio o ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per passare il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Aurora. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone di autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, Mary chiede all'amministratore di aggiornare la sue policy per poter eseguire l'operazione `iam:PassRole`.

Voglio consentire a persone esterne al mio account AWS di accedere alle mie risorse Aurora

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per capire se Aurora supporta queste funzionalità, consulta Come funziona Aurora con IAM.
- Per informazioni su come garantire l'accesso alle risorse negli account AWS che possiedi, consulta Fornire l'accesso a un utente IAM in un altro account AWS che si possiede nella Guida per l'utente di IAM.
- Per capire come fornire l'accesso alle risorse ad account AWS di terze parti, consulta Concessione dell'accesso agli account AWS di proprietà di terze parti nella Guida per l'utente di IAM.
- Per capire come fornire l'accesso tramite la federazione delle identità, consulta Fornire accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Registrazione e monitoraggio in Amazon Aurora

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon Aurora e AWS delle tue soluzioni. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. AWS offre diversi strumenti per monitorare le risorse Amazon Aurora e rispondere a potenziali incidenti:

CloudWatch Allarmi Amazon

> Utilizzando Amazon CloudWatch alarms, controlli una singola metrica per un periodo di tempo specificato. Se la metrica supera una determinata soglia, viene inviata una notifica a un argomento o una policy di Amazon SNS. AWS Auto Scaling CloudWatch gli allarmi non richiamano azioni perché si trovano in uno stato particolare. È necessario invece cambiare lo stato e mantenerlo per un numero di periodi specificato.

AWS CloudTrail Registri

CloudTrail fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in . CloudTrail acquisisce tutte le chiamate API per Amazon Aurora come eventi, incluse le chiamate dalla console e le chiamate di codice alle operazioni dell'API Amazon RDS. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta effettuata ad Aurora, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi. Per ulteriori informazioni, consulta Monitoraggio delle chiamate API Aurora in AWS CloudTrail.

Monitoraggio avanzato

Amazon Aurora fornisce parametri in tempo reale per il sistema operativo (OS) su cui viene eseguito il cluster di istanze . Puoi visualizzare i parametri per il tuo cluster di DB utilizzando la console o utilizzare l'output JSON di Enhanced Monitoring di Amazon CloudWatch Logs in un sistema di monitoraggio a tua scelta. Per ulteriori informazioni, consulta Monitoraggio dei parametri del sistema operativo con il monitoraggio avanzato.

Performance Insights di Amazon RDS

Performance Insights amplia le funzionalità di monitoraggio esistenti di Amazon Aurora per illustrare le prestazioni del database e aiutarti ad analizzare eventuali problemi che lo riguardano. Con il pannello di controllo di Performance Insights, è possibile visualizzare il carico del database e filtrare il carico in base alle attese, alle istruzioni SQL, agli host o agli utenti. Per ulteriori informazioni, consulta Monitoraggio del carico DB con Performance Insights su Amazon Aurora.

Log di database

Puoi visualizzare, scaricare e guardare i log del database utilizzando l'API AWS Management Console, AWS CLI o RDS. Per ulteriori informazioni, consulta Monitoraggio dei file di log di Aurora.

Raccomandazioni per Amazon Aurora

Amazon Aurora fornisce consigli automatici per le risorse di database. Questi consigli forniscono indicazioni sulle best practice analizzando la configurazione, l'utilizzo e i dati sulle prestazioni del cluster di DB. Per ulteriori informazioni, consulta .

Notifica di Amazon Aurora

Amazon Aurora utilizza Amazon Simple Notification Service (Amazon SNS) per fornire notifiche quando si verifica un evento Amazon Aurora. Queste notifiche possono essere in qualsiasi modulo di notifica supportato da Amazon SNS per una AWS regione, ad esempio un'e-mail,

un messaggio di testo o una chiamata a un endpoint HTTP. Per ulteriori informazioni, consulta Utilizzo delle notifiche di RDS eventi di Amazon.

AWS Trusted Advisor

Trusted Advisor si basa sulle migliori pratiche apprese servendo centinaia di migliaia di AWS clienti. Trusted Advisor ispeziona l' AWS ambiente e quindi formula raccomandazioni quando esistono opportunità per risparmiare denaro, migliorare la disponibilità e le prestazioni del sistema o contribuire a colmare le lacune di sicurezza. Tutti i AWS clienti hanno accesso a cinque Trusted Advisor controlli. I clienti con un piano di supporto Business o Enterprise possono visualizzare tutti i Trusted Advisor controlli.

Trusted Advisor dispone dei seguenti controlli Amazon Aurora:

- Amazon Aurora inattive
- Rischio di accesso al gruppo di sicurezza Amazon Aurora
- Amazon Aurora
- Amazon Aurora Multi-AZ
- Accessibilità dell'istanza database Aurora

Per ulteriori informazioni su questi controlli, consulta best practice Trusted Advisor (Controlli).

Flussi di attività di database

I flussi di attività di database proteggono i database da minacce interne controllando l'accesso DBA ai flussi di attività di database. Pertanto, la raccolta, la trasmissione, l'archiviazione e la successiva elaborazione del flusso di attività del database non sono accessibili a chi gestisce DBAs il database. Gli stream dell'attività di database consentono di fornire protezioni per il database e soddisfare requisiti normativi e di conformità. Per ulteriori informazioni, consulta Monitoraggio di Amazon Aurora tramite i flussi di attività del database.

Per ulteriori informazioni sul monitoraggio di Aurora, vedere. Monitoraggio dei parametri in un cluster di database Amazon Aurora

Convalida della conformità per

Revisori di terze parti valutano la sicurezza e la conformità di Amazon Aurora nell'ambito di AWS diversi programmi di conformità. Questi includono SOCPCI, Fed RAMP e HIPAA altri.

Per un elenco dei AWS servizi che rientrano nell'ambito di specifici programmi di conformità, consulta la sezione AWS Servizi rientranti nell'ambito del programma di conformità. Per informazioni generali, consultare Programmi per la conformità di AWS.

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact.

La tua responsabilità di conformità quando usi Aurora è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità dell'organizzazione e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide introduttive su sicurezza e conformità: queste guide all'implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità. AWS
- Architettura per la HIPAA sicurezza e la conformità su Amazon Web Services: questo white paper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni conformi. HIPAA
- AWS risorse per la conformità: questa raccolta di cartelle di lavoro e guide che potrebbero riguardare il tuo settore e la tua località.
- AWS Config— Questo AWS servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub— Ciò Servizio AWS fornisce una visione completa dello stato di sicurezza dell'utente all'interno. AWS La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.

Resilienza in Amazon Aurora

L'infrastruttura globale di AWS è basata su regioni AWS e zone di disponibilità. AWS Le Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e velocità effettiva elevata. Con le zone di disponibilità, è possibile progettare e gestire le applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo.

Per ulteriori informazioni sulle regioni e le zone di disponibilità AWS, consulta Infrastruttura globale di AWS.

Oltre all'infrastruttura globale AWS, Aurora offre numerose funzionalità per supportare la resilienza dei dati e le esigenze di backup.

Backup e ripristino

Aurora esegue automaticamente il backup del volume del cluster e conserva i dati per la durata del tempo di conservazione del backup. I backup Aurora sono continui e incrementali, in modo da poter eseguire rapidamente un ripristino da qualsiasi punto del tempo di conservazione del backup. Durante la scrittura dei dati di backup, non si verifica alcun impatto sulle prestazioni o interruzione del funzionamento del servizio del database. Puoi specificare un tempo di conservazione del backup, da 1 a 35 giorni, durante la creazione o la modifica di un cluster di database.

Se desideri conservare un backup oltre il tempo di conservazione del backup, puoi anche eseguire uno snapshot dei dati del volume del cluster. Aurora mantiene i dati di ripristino incrementali per l'intero tempo di conservazione del backup. Quindi, è sufficiente creare uno snapshot per i dati che desideri conservare oltre il tempo di conservazione del backup. Puoi creare un nuovo cluster di database dalla snapshot.

Puoi recuperare i dati creando un nuovo cluster di database Aurora dai dati di backup conservati da Aurora o da uno snapshot del cluster di database salvata. Puoi ripristinare rapidamente una nuova copia di un cluster di database creato dai dati di backup a un momento qualsiasi del tempo di conservazione del backup. Data la natura continua e incrementale dei backup di Aurora a durante il tempo di conservazione del backup, non dovrai effettuare snapshot frequenti dei dati per migliorare i tempi di ripristino.

Per ulteriori informazioni, consulta Backup e ripristino di un cluster DB Amazon Aurora.

Replica

Le repliche Aurora sono endpoint indipendenti in un cluster di database Aurora, utilizzati al meglio per operazioni di dimensionamento della lettura e maggiore disponibilità. È possibile distribuire fino a 15 repliche di Aurora nelle zone di disponibilità sulle quali si estende un cluster di database in una regione AWS. Il volume del cluster di database si compone di più copie dei dati per il cluster di database. Tuttavia, i dati nel volume del cluster sono rappresentati come singolo volume logico all'istanza database primaria e alle repliche di Aurora nel cluster di database. Se l'istanza database primaria non va a buon fine, una replica di Aurora viene promossa a istanza primaria.

Aurora supporta inoltre opzioni di replica specifiche per Aurora MySQL e Aurora PostgreSQL.

Per ulteriori informazioni, consulta Replica con Amazon Aurora.

Failover

Aurora memorizza copie di dati in un cluster di database in più zone di disponibilità in una singola regione AWS. Lo storage avviene indipendentemente dal fatto che le istanze database nel cluster di database comprendano più zone di disponibilità. Quando si creano repliche di Aurora tra le zone di disponibilità, Aurora effettua automaticamente il provisioning e le mantiene in modo sincrono. L'istanza database primaria viene replicata in modo sincrono nelle zone di disponibilità sulle repliche di Aurora per fornire ridondanza dati, eliminare blocchi I/O e ridurre al minimo i picchi di latenza durante i backup di sistema. Eseguendo un cluster di database con disponibilità elevata, è possibile migliorare la disponibilità durante la manutenzione pianificata del sistema e consentire di proteggere i database da errori e interruzioni relative alle zone di disponibilità.

Per ulteriori informazioni, consultare Elevata disponibilità di Amazon Aurora.

Sicurezza dell'infrastruttura in Aurora

In quanto servizio gestito, Amazon Relational Database Service è protetto AWS dalla sicurezza di rete globale. Per informazioni sui servizi di AWS sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi API le chiamate AWS pubblicate per accedere ad Amazon RDS tramite la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). Richiediamo TLS 1.2 e consigliamo TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS) come (Ephemeral Diffie-Hellman) o DHE (Elliptic Curve Ephemeral Diffie-Hellman). ECDHE La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale. IAM O puoi utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per sottoscrivere le richieste.

Inoltre, Aurora offre funzionalità che aiutano a supportare la sicurezza dell'infrastruttura.

Gruppi di sicurezza

I gruppi di sicurezza controllano l'accesso del traffico in entrata e in uscita di un cluster database. Per impostazione predefinita, l'accesso alla rete è disattivato per un cluster database. Puoi specificare delle norme in un gruppo di sicurezza che consente l'accesso da un intervallo di indirizzi IP, dalla porta o da un gruppo di sicurezza. Una volta configurate le regole in ingresso, si applicano le stesse regole a tutti i cluster database con associazione a tale gruppo di sicurezza.

Per ulteriori informazioni, consulta Controllo dell'accesso con i gruppi di sicurezza.

Public accessibility (Accesso pubblico)

Quando avvii un'istanza DB all'interno di un cloud privato virtuale (VPC) basato sul VPC servizio Amazon, puoi attivare o disattivare l'accessibilità pubblica per quell'istanza DB. Per indicare se l'istanza DB che crei ha un DNS nome che si risolve in un indirizzo IP pubblico, utilizzi il parametro Public accessibility. Questo parametro consente di stabilire se esiste un accesso pubblico all'istanza

database. È possibile modificare un'istanza database per attivare o disattivare l'accessibilità pubblica modificando il parametro Public accessibility (Accessibilità pubblica).

Per ulteriori informazioni, consulta Nascondere cluster database in un VPC da Internet.

> **Note**
>
> Se l'istanza DB si trova in un account VPC ma non è accessibile pubblicamente, puoi anche utilizzare una connessione AWS da sito a sito o una VPN connessione per accedervi da una AWS Direct Connect rete privata. Per ulteriori informazioni, consulta Riservatezza del traffico Internet.

API Amazon RDS ed endpoint VPC dell'interfaccia (AWS PrivateLink)

È possibile stabilire una connessione privata tra gli endpoint VPC e l'API Amazon RDS creando un endpoint VPC di interfaccia. Endpoint di interfaccia con tecnologia AWS PrivateLink.

AWS PrivateLink consente di accedere in modo privato alle operazioni delle API di Amazon RDS senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. AWS Direct Connect Le istanze database nel VPC non necessitano di indirizzi IP pubblici per comunicare con gli endpoint dell'API Amazon RDS per avviare, modificare o terminare le istanze database e i cluster DB. Inoltre, le istanze database non richiedono indirizzi IP pubblici per utilizzare le operazioni dell'API RDS disponibili. Il traffico di rete tra il VPC e Amazon RDS non esce dalla rete Amazon.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle sottoreti. Per maggiori informazioni sulle interfacce di rete elastiche, consulta le interfacce di rete elastiche nella Guida dell'utente di Amazon EC2.

Per ulteriori informazioni sugli endpoint VPC, consulta Interface VPC endpoints () nella Amazon VPC User AWS PrivateLink Guide. Per informazioni sulle operazioni API RDS, consulta Documentazione di riferimento delle API di Amazon RDS.

Non è necessaria un'interfaccia endpoint VPC per connettersi a un cluster database. Per ulteriori informazioni, consulta Scenari per l'accesso a un cluster di DB in un VPC.

Considerazioni sugli endpoint VPC

Prima di impostare un endpoint VPC di interfaccia per endpoint Amazon RDS API, verificare di esaminare le proprietà e le limitazioni degli endpoint di interfaccia in Guida per l'utente di Amazon VPC.

Tutte le operazioni API RDS rilevanti per la gestione delle risorse Amazon Aurora sono disponibili dal VPC utilizzando AWS PrivateLink.

Le policy degli endpoint VPC sono supportate per gli endpoint dell'API RDS. Per impostazione predefinita, l'accesso completo alle operazioni API RDS è consentito attraverso l'endpoint. Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Disponibilità

L'API Amazon RDS attualmente supporta gli endpoint VPC nelle seguenti regioni: AWS

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Canada occidentale (Calgary)
- Cina (Pechino)
- China (Ningxia)
- Europa (Francoforte)
- Europa (Zurigo)
- Europa (Irlanda)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- Europa (Milano)
- Israele (Tel Aviv)
- Medio Oriente (Bahrein)
- Sud America (San Paolo)
- AWS GovCloud (Stati Uniti orientali)

- AWS GovCloud (Stati Uniti occidentali)

Creazione di un endpoint VPC di interfaccia per l'API Amazon RDS

Puoi creare un endpoint VPC per l'API Amazon RDS utilizzando la console Amazon VPC o il (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Creazione di un endpoint dell'interfaccia nella Guida per l'utente di Amazon VPC.

Crea un endpoint VPC per l'API Amazon RDS utilizzando il nome del servizio `com.amazonaws.region.rds`.

Escludendo AWS le regioni in Cina, se abiliti il DNS privato per l'endpoint, puoi effettuare richieste API ad Amazon RDS con l'endpoint VPC utilizzando ad esempio il nome DNS predefinito per la regione. AWS `rds.us-east-1.amazonaws.com` Per le AWS regioni Cina (Pechino) e Cina (Ningxia), puoi effettuare richieste API con l'endpoint VPC utilizzando e, rispettivamente. `rds-api.cn-north-1.amazonaws.com.cn` `rds-api.cn-northwest-1.amazonaws.com.cn`

Per ulteriori informazioni, consulta Accesso a un servizio tramite un endpoint dell'interfaccia in Guida per l'utente di Amazon VPC.

Creazione di una policy di endpoint VPC per l'API Amazon RDS

Puoi allegare una policy di endpoint all'endpoint VPC che controlla l'accesso all'API Amazon RDS. Questo codice specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC in Guida per l'utente di Amazon VPC.

Ad esempio, policy di endpoint VPC per le operazioni API Amazon RDS

Di seguito è riportato un esempio di una policy endpoint per l'API Amazon RDS. Se collegato a un endpoint, questa policy concede l'accesso alle operazioni API Amazon RDS riportate per tutte le entità su tutte le risorse.

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

Esempio: policy degli endpoint VPC che nega tutti gli accessi da un account specificato AWS

La seguente politica degli endpoint VPC nega all' AWS account `123456789012` tutti gli accessi alle risorse che utilizzano l'endpoint. La policy consente tutte le operazioni da altri account.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Best practice relative alla sicurezza di Amazon Aurora

Usa gli account AWS Identity and Access Management (IAM) per controllare l'accesso alle operazioni dell'API Amazon RDS, in particolare le operazioni che creano, modificano o eliminano risorse Amazon Aurora. Tali risorse includono DB, cluster, gruppi di sicurezza e gruppi di parametri.

- Crea un utente individuale per ogni persona che gestisce le risorse Amazon Aurora, incluso te stesso. Non utilizzare credenziali AWS root per gestire le risorse Amazon Aurora.
- Assegna a ciascun utente un set minimo di autorizzazioni richieste per eseguire le proprie mansioni.
- Utilizza gruppi IAM per gestire in modo efficace le autorizzazioni per più utenti.
- Ruota periodicamente le credenziali IAM.
- Configura AWS Secrets Manager per ruotare automaticamente i segreti per Aurora. Per ulteriori informazioni, consulta Rotazione dei segreti AWS Secrets Manager nella Guida per l'utente di AWS Secrets Manager . Puoi anche recuperare le credenziali da programmaticamente. AWS Secrets Manager Per ulteriori informazioni, consulta Recupero del valore segreto nella Guida per l'utente di AWS Secrets Manager .

Per ulteriori informazioni sulla sicurezza di Amazon Aurora, consulta. Sicurezza in Amazon Aurora Per ulteriori informazioni su IAM, consulta AWS Identity and Access Management. Per informazioni sulle best practice di IAM, consulta Best practice di IAM.

AWS Security Hub utilizza controlli di sicurezza per valutare le configurazioni delle risorse e gli standard di sicurezza per aiutarti a rispettare vari framework di conformità. Per ulteriori informazioni sull'utilizzo di Security Hub per la valutazione delle risorse RDS, consulta Controlli di Amazon Relational Database Service (Amazon RDS) nella Guida per l'utente di AWS Security Hub .

Puoi monitorare l'uso di RDS in relazione alle best practice sulla sicurezza utilizzando Centrale di sicurezza. Per ulteriori informazioni, consulta Cos'è? AWS Security Hub .

Usa l' AWS Management Console AWS CLI, the o l'API RDS per modificare la password per il tuo utente principale. Se usi uno strumento diverso, ad esempio un client SQL, per modificare la password dell'utente master, i privilegi per l'utente potrebbero venire revocati involontariamente.

Amazon GuardDuty è un servizio di monitoraggio continuo della sicurezza che analizza ed elabora varie fonti di dati, inclusa l'attività di accesso ad Amazon RDS. Utilizza feed di intelligence sulle minacce e apprendimento automatico per identificare comportamenti di accesso imprevisti, potenzialmente non autorizzati e sospetti e attività dannose all'interno del tuo ambiente. AWS

Quando Amazon GuardDuty RDS Protection rileva un tentativo di accesso potenzialmente sospetto o anomalo che indica una minaccia per il database, GuardDuty genera una nuova scoperta con dettagli sul database potenzialmente compromesso. Per ulteriori informazioni, consulta Monitoraggio delle minacce con Amazon GuardDuty RDS Protection per Amazon Aurora.

Controllo dell'accesso con i gruppi di sicurezza

I gruppi di sicurezza VPC controllano l'accesso del traffico in entrata e in uscita da un cluster di DB. Per impostazione predefinita, l'accesso alla rete è disattivato per un cluster di DB. Puoi specificare delle norme in un gruppo di sicurezza che consente l'accesso da un intervallo di indirizzi IP, dalla porta o da un gruppo di sicurezza. Una volta configurate le regole di ingresso, le stesse regole si applicano a tutti i cluster di DB associati a quel gruppo di sicurezza. Puoi specificare fino a 20 norme in un gruppo di sicurezza.

Panoramica dei gruppi di sicurezza VPC

Ogni regola del gruppo di sicurezza VPC consente a una fonte specifica di accedere a un cluster di DB in un VPC associato a quel gruppo di sicurezza VPC. L'origine può essere una serie di indirizzi (ad esempio, 203.0.113.0/24) oppure un altro gruppo di sicurezza VPC. Specificando un gruppo di sicurezza VPC come origine, consenti il traffico in entrata da tutte le istanze (in genere i server dell'applicazione) che usano il gruppo di sicurezza VPC. I gruppi di sicurezza VPC possono avere regole che gestiscono sia il traffico in entrata che in uscita. Le regole del traffico in uscita si applicano solo se il cluster di DB funge da client. È necessario utilizzare l' EC2API Amazon o l'opzione Security Group sulla console VPC per creare gruppi di sicurezza VPC.

Quando crei regole per il tuo gruppo di sicurezza VPC che consentono l'accesso ai cluster di nel tuo VPC, devi specificare una porta per ogni intervallo di indirizzi a cui la regola consente l'accesso. Ad esempio, se si desidera abilitare l'accesso SSH (Secure Shell) alle istanze nel VPC, devi creare una regola che consenta l'accesso alla porta TCP 22 per l'intervallo di indirizzi specificato.

È possibile configurare più gruppi di sicurezza VPC che consentono l'accesso a porte diverse per istanze diverse nel VPC. Ad esempio, è possibile creare un gruppo di sicurezza VPC che consenta l'accesso alla porta TCP 80 per i server web nel VPC. Puoi quindi creare un altro gruppo di sicurezza VPC che consenta l'accesso alla porta TCP 3306 per RDS per le istanze DB nel tuo VPC.

> **Note**
>
> In un cluster database Aurora, il gruppo di sicurezza VPC associato al cluster database è anche associato a tutte le istanze database nel cluster database. Se modifichi il gruppo di sicurezza VPC per il cluster database o per un'istanza database, la modifica viene applicata automaticamente a tutte le istanze database nel cluster database.

Per ulteriori informazioni sui gruppi di sicurezza VPC, consulta la pagina Gruppi di sicurezza nella Guida per l'utente di Amazon Virtual Private Cloud.

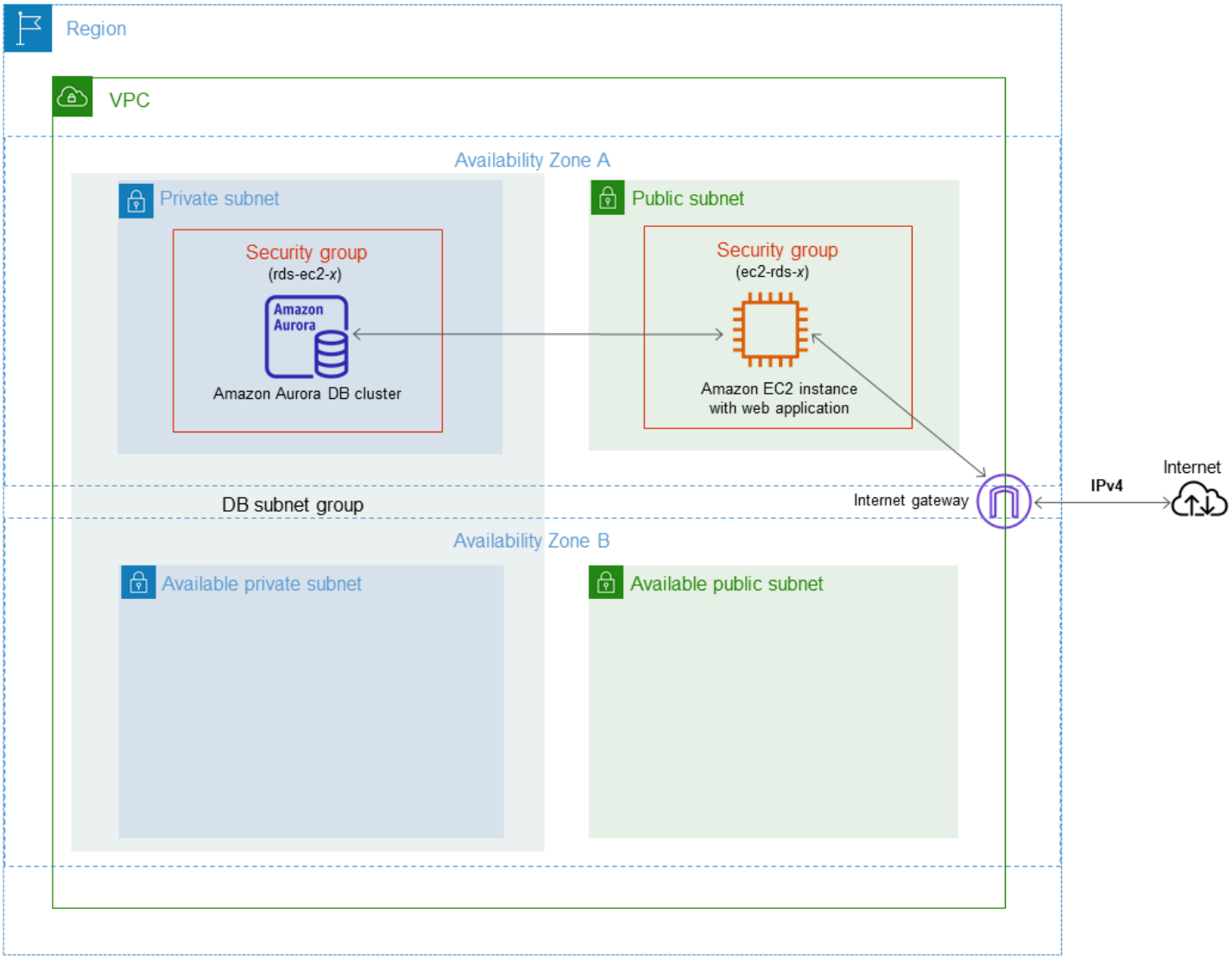
> **Note**
>
> Se il cluster di DB si trova in un VPC ma non è accessibile pubblicamente, puoi anche utilizzare una connessione AWS Site-to-Site VPN o una AWS Direct Connect connessione per accedervi da una rete privata. Per ulteriori informazioni, consulta Riservatezza del traffico Internet.

Scenario del gruppo di sicurezza

Un uso comune di un cluster di DB in un VPC consiste nel condividere i dati con un server di applicazioni in esecuzione in un' EC2 istanza Amazon nello stesso VPC, a cui accede un'applicazione client esterna al VPC. In questo scenario, si utilizzano le pagine RDS e VPC su o AWS Management Console le operazioni RDS EC2 e API per creare le istanze e i gruppi di sicurezza necessari:

1. Creare un gruppo di sicurezza VPC (ad esempio, `sg-0123ec2example`) e definire le regole in entrata che utilizzano l'indirizzo IP dell'applicazione client usato nell'indirizzo IP dell'applicazione del client come origine. Questo gruppo di sicurezza consente all'applicazione client di connettersi alle EC2 istanze in un VPC che utilizza questo gruppo di sicurezza.
2. Crea un' EC2 istanza per l'applicazione e aggiungi l' EC2 istanza al gruppo di sicurezza VPC (`sg-0123ec2example`) creato nel passaggio precedente.
3. Creare un secondo gruppo di sicurezza VPC (ad esempio, `sg-6789rdsexample`) e creare una nuova regola specificando il gruppo di sicurezza VPC creato nel passaggio 1 (`sg-0123ec2example`) come l'origine.
4. Crea un nuovo cluster di DB e aggiungi il cluster di DB al gruppo di sicurezza VPC (`sg-6789rdsexample`) creato nel passaggio precedente. Quando crei il cluster di DB, utilizza lo stesso numero di porta specificato per la regola del gruppo di sicurezza VPC (`sg-6789rdsexample`) creata nel passaggio 3.

Il seguente diagramma mostra questo scenario.



Per istruzioni dettagliate sulla configurazione di un VPC per questo scenario, vedere. Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4 Per ulteriori informazioni sull'utilizzo di un VPC, vedere. Amazon VPC e Aurora

Creazione di un gruppo di sicurezza VPC

Puoi creare un gruppo di sicurezza VPC per un'istanza database tramite la console VPC. Per informazioni sulla creazione di un gruppo di sicurezza, consulta le pagine Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza e Gruppi di sicurezza nella Guida per l'utente di Amazon Virtual Private Cloud.

Associazione di un gruppo di sicurezza a un cluster database

Puoi associare un gruppo di sicurezza a un cluster DB utilizzando Modify cluster sulla console RDS, l'API `ModifyDBCluster` Amazon RDS o il `modify-db-cluster` AWS CLI comando.

Il seguente esempio CLI associa un gruppo VPC specifico e rimuove i gruppi di sicurezza DB dal cluster DB.

```
aws rds modify-db-cluster --db-cluster-identifier dbName --vpc-security-group-ids sg-ID
```

Per informazioni sulla modifica di un cluster DB, vedere. Modifica di un cluster database Amazon Aurora

Privilegi dell'account utente master

Quando crei un nuovo cluster di DB, l'utente master predefinito che utilizzi ottiene determinati privilegi per quel cluster di DB. Non è possibile modificare il nome utente principale dopo la creazione del cluster di DB.

> **Important**
>
> Si consiglia di non utilizzare l'utente master direttamente nelle applicazioni. Rispetta piuttosto la best practice di utilizzare un utente del database creato con i privilegi minimi richiesti per l'applicazione.

> **Note**
>
> Se elimini accidentalmente le autorizzazioni per l'utente master, puoi ripristinarle modificando il cluster di DB e impostando una nuova password per l'utente principale. Per ulteriori informazioni sulla modifica di un cluster di DB, consulta.

La tabella seguente mostra i privilegi e i ruoli di database ottenuti dall'utente master per ciascuno dei motori di database.

Motore del database	Privilegio del sistema	Ruolo di database
Aurora MySQL	Versione 2: `ALTER,ALTER ROUTINE, CREATE,CREATE ROUTINE,CREATE TEMPORARY TABLES,CREATE USER,CREATE VIEW,DELETE,DROP, EVENT,EXECUTE,GRANT OPTION,INDEX,INSERT, LOAD FROM S3,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATI ON CLIENT , REPLICATION SLAVE ,SELECT, SELECT INTO S3,SHOW DATABASES , SHOW VIEW,TRIGGER, UPDATE`	—
	Versione 3: `ALTER,APPLICATION_PASSWORD_ADMIN , ALTER ROUTINE,CONNECTION_ADMIN , CREATE,CREATE ROLE,CREATE ROUTINE,CREATE TEMPORARY TABLES, CREATE USER,CREATE VIEW, DELETE,DROP,DROP ROLE, EVENT,EXECUTE,INDEX, INSERT,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATION CLIENT , REPLICATI ON SLAVE ,ROLE_ADMIN , SET_USER_ ID ,SELECT,SHOW DATABASES ,SHOW VIEW,TRIGGER, UPDATE,XA_RECOVER_ADMIN`  A partire dalla versione 3.04.0 di Aurora MySQL, anche l'utente master ottiene il privilegio. `SHOW_ROUT INE`  A partire dalla versione 3.09.0 di Aurora MySQL, l'utente master ottiene anche i privilegi,, e. `FLUSH_OPTIMIZER_COSTS FLUSH_STATUS FLUSH_TABLES FLUSH_USER_RESOURCES`	`rds_superuser_role` Per ulteriori informazioni su rds_superuser_role, vedere. Privilegio basato sui ruoli

Motore del database	Privilegio del sistema	Ruolo di database
Aurora PostgreSQL	`LOGIN,NOSUPERUSER , INHERIT,CREATEDB, CREATEROLE ,NOREPLICATION ,VALID UNTIL 'infinity'`	`RDS_SUPERUSER` Per ulteriori informazioni su RDS_SUPERUSER, vedere. Informazioni su ruoli e autorizzazioni di PostgreSQL

Utilizzo di ruoli collegati ai servizi per Amazon Aurora

Amazon Aurora AWS Identity and Access Management utilizza ruoli collegati ai servizi (IAM). Un ruolo collegato ai servizi è un tipo univoco di ruolo IAM collegato direttamente a Amazon Aurora. I ruoli collegati ai servizi sono predefiniti da Aurora e includono tutte le autorizzazioni richieste dal servizio per chiamare altri servizi per tuo conto. AWS

Un ruolo collegato ai servizi semplifica l'uso di Amazon Aurora perché non sarà più necessario aggiungere manualmente le autorizzazioni. Amazon Aurora definisce le autorizzazioni dei ruoli collegati ai servizi e, salvo diversamente definito, solo Amazon Aurora può assumere il ruolo. Le autorizzazioni definite includono la policy di trust e la policy delle autorizzazioni. Una policy delle autorizzazioni specifica non può essere collegata a un'altra entità IAM.

È possibile eliminare i ruoli solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon Aurora perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consulta i servizi AWS che funzionano con IAM e cerca i servizi che riportano Yes (Sì) nella colonna Service-Linked Role (Ruolo associato ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon Aurora

Il ruolo collegato al servizio AWSService RoleFor RDS prevede che i seguenti servizi assumano il ruolo:

- `rds.amazonaws.com`

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonRDSServiceRolePolicy` che concede le autorizzazioni per operare nell'account.

Per ulteriori informazioni su questa politica, incluso il documento sulla politica JSON, consulta Amazon RDSService RolePolicy nella AWS Managed Policy Reference Guide.

> **Note**
>
> Per consentire a un'entità IAM (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi, devi configurare le autorizzazioni. Se viene visualizzato il messaggio di errore seguente:
> Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
> Accertati che le seguenti autorizzazioni siano abilitate:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon Aurora

Non devi creare manualmente un ruolo collegato ai servizi. Quando si crea un cluster di database, Amazon Aurora crea nuovamente il ruolo collegato al servizio per conto dell'utente.

> **Important**
>
> Se utilizzavi il servizio Amazon Aurora prima del 1° dicembre 2017, quando ha iniziato a supportare ruoli collegati al servizio, Amazon nel tuo account. AWSService RoleFor Per ulteriori informazioni, vedi Un nuovo ruolo è apparso nel mio account. AWS

Se elimini questo ruolo collegato ai servizi e quindi devi ricrearlo di nuovo, puoi utilizzare lo stesso processo per ricreare il ruolo nel tuo account. Quando si crea un cluster di database, Amazon Aurora crea nuovamente il ruolo collegato al servizio per tuo conto.

Modifica di un ruolo collegato ai servizi per Amazon Aurora

Amazon Aurora non consente di modificare AWSService RoleFor il ruolo collegato al servizio RDS. Dopo aver creato un ruolo collegato al servizio, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon Aurora

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare quel ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, prima di poter eliminare il ruolo collegato al servizio, devi eliminare tutti i cluster database.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM per eliminare un ruolo collegato ai servizi, devi innanzitutto verificare che il ruolo non abbia sessioni attive ed eliminare tutte le risorse utilizzate dal ruolo.

Per verificare se il ruolo collegato ai servizi dispone di una sessione attiva nella console IAM

1. Accedi AWS Management Console e apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione della console IAM seleziona Ruoli. Quindi scegli il nome (non la casella di controllo) del ruolo AWSService RoleFor RDS.
3. Nella pagina di riepilogo per il ruolo scelto, scegli la scheda Ultimo accesso.
4. Nella scheda Ultimo accesso, esamina le attività recenti relative al ruolo collegato al servizio.

> **Note**
>
> Se non sei sicuro che Amazon Aurora stia utilizzando AWSService RoleFor il ruolo RDS, puoi provare a eliminarlo. Se il servizio sta utilizzando il ruolo, l'eliminazione non andrà a buon fine e potrai visualizzare le regioni AWS in cui il ruolo viene utilizzato. Se il ruolo è in uso, prima di poterlo eliminare dovrai attendere il termine della sessione. Non puoi revocare la sessione per un ruolo collegato al servizio.

Eliminazione di tutti i cluster

Utilizza una delle procedure seguenti per eliminare un singolo cluster. Ripeti la procedura per ogni cluster.

Per eliminare un cluster (console)

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nell'elenco Databases (Database), scegliere il cluster da eliminare.
3. Per Cluster Actions (Operazioni cluster), scegliere Delete (Elimina).
4. Scegliere Delete (Elimina).

Per eliminare un cluster (CLI)

Consulta `delete-db-cluster` in Riferimento ai comandi AWS CLI .

Per eliminare un cluster (API)

Consulta `DeleteDBCluster` nella Amazon RDS API Reference.

Puoi utilizzare la console IAM, l'IAM CLI o l'API IAM per eliminare il ruolo collegato al servizio AWSService RoleFor RDS. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato al servizio nella Guida per l'utente di IAM.

Autorizzazioni di ruolo collegate ai servizi per Amazon RDS Beta

Amazon Aurora utilizza il ruolo collegato al servizio denominato `AWSServiceRoleForRDSBeta` per consentire ad Amazon Aurora di AWS chiamare i servizi per conto delle tue risorse DB RDS.

Il ruolo AWSService RoleFor RDSBeta collegato al servizio prevede che i seguenti servizi assumano il ruolo:

- `rds.amazonaws.com`

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonRDSBetaServiceRolePolicy` che concede le autorizzazioni per operare nell'account. Per ulteriori informazioni, consulta AWS politica gestita: A mazonRDSBeta ServiceRolePolicy.

> **Note**
>
> Per consentire a un'entità IAM (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi, devi configurare le autorizzazioni. Se viene visualizzato il messaggio di errore seguente:
> Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
> Accertati che le seguenti autorizzazioni siano abilitate:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSBetaServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Ruolo collegato ai servizi per Amazon RDS Preview

Amazon Aurora utilizza il ruolo collegato al servizio denominato `AWSServiceRoleForRDSPreview` per consentire ad Amazon Aurora di AWS chiamare i servizi per conto delle tue risorse DB RDS.

Il ruolo AWSService RoleFor RDSPreview collegato al servizio prevede che i seguenti servizi assumano il ruolo:

- `rds.amazonaws.com`

A questo ruolo collegato ai servizi è collegata un policy di autorizzazione denominata `AmazonRDSPreviewServiceRolePolicy` che concede le autorizzazioni per operare nell'account. Per ulteriori informazioni, consulta AWS politica gestita: A mazonRDSPreview ServiceRolePolicy.

> **Note**
>
> Per consentire a un'entità IAM (ad esempio un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi, devi configurare le autorizzazioni. Se viene visualizzato il messaggio di errore seguente:
> Unable to create the resource. (Impossibile creare la risorsa. Verify that you have permission to create service linked role. (Verifica di possedere le autorizzazioni necessarie per creare un ruolo collegato ai servizi.) Otherwise wait and try again later. (In caso contrario, attendi e riprova più tardi.
> Accertati che le seguenti autorizzazioni siano abilitate:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSPreviewServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Amazon VPC e Aurora

Amazon Virtual Private Cloud (AmazonVPC) ti consente di avviare AWS risorse, come i cluster Aurora DB, in un cloud privato virtuale (). VPC

Quando si utilizza unVPC, si ha il controllo sul proprio ambiente di rete virtuale. Puoi scegliere il tuo intervallo di indirizzi IP, creare sottoreti e configurare liste di routing e di controllo accessi. L'esecuzione del cluster di DB in unVPC.

Gli account hanno un valore predefinitoVPC. Tutti i nuovi cluster di DB vengono creati in modo predefinito, a VPC meno che non venga specificato diversamente.

Argomenti

- Uso di un cluster database in un VPC
- Scenari per l'accesso a un cluster di DB in un VPC
- Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4
- Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Di seguito, puoi trovare una discussione sulle VPC funzionalità relative alle (cluster Amazon Aurora DB). Per ulteriori informazioni su AmazonVPC, consulta Amazon VPC Getting Started Guide e Amazon VPC User Guide.

Uso di un cluster database in un VPC

Il cluster database deve essere all'interno di un cloud privato virtuale (VPC). Un VPC è una rete virtuale logicamente isolata dalle altre reti virtuali nel cloud. AWS Amazon VPC ti consente di avviare AWS risorse, come un di istanze Amazon o un'istanza Amazon EC2 , in un VPC. Il VPC può essere un VPC predefinito fornito con l'account o uno creato da te. Tutti VPCs sono associati al tuo account. AWS

Il VPC predefinito ha tre sottoreti che è possibile usare per isolare le risorse all'interno del VPC. Il VPC predefinito ha anche un gateway Internet che può essere usato per fornire l'accesso alle risorse all'interno del VPC dall'esterno del VPC.

Per un elenco di scenari relativi ai cluster database Amazon Aurora in un VPC , consulta Scenari per l'accesso a un cluster di DB in un VPC.

Argomenti

- Uso di un cluster database in un VPC
- Utilizzo di gruppi di sottoreti database
- Sottoreti condivise
- Assegnazione di indirizzi IP in Amazon Aurora
- Nascondere cluster database in un VPC da Internet
- Creazione di un cluster database in un VPC

Nei seguenti tutorial puoi imparare a creare un VPC da usare per uno scenario Amazon Aurora comune:

- Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4
- Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Uso di un cluster database in un VPC

Ecco alcuni consigli per l'uso di un cluster database in un VPC:

- Il VPC deve avere almeno due sottoreti. Una sottorete è un segmento dell'intervallo di indirizzi IP di un VPC che è possibile specificare e utilizzare per raggruppare i cluster database in base alle esigenze operative e di sicurezza.
- Se desideri che il cluster database nel VPC sia accessibile a livello pubblico, verifica di aver attivato gli attributi VPC DNS hostnames (Nomi host DNS) e DNS resolution (Risoluzione DNS).
- Il VPC deve includere un gruppo di sottoreti database creato da te. Puoi creare un gruppo di sottoreti di database, specificando le sottoreti create. Amazon Aurora sceglie una sottorete e un indirizzo IP all'interno di quella stessa sottorete da associare all'istanza database primaria nel cluster DB. L'istanza database primaria utilizza la zona di disponibilità che contiene la sottorete.
- Il VPC deve avere un gruppo di sicurezza VPC che consente l'accesso al cluster database.

  Per ulteriori informazioni, consulta Scenari per l'accesso a un cluster di DB in un VPC.
- I blocchi CIDR in ciascuna delle sottoreti devono essere sufficientemente grandi da contenere gli indirizzi IP di riserva per Amazon Aurora da utilizzare durante le attività di manutenzione, inclusi il failover e il dimensionamento delle risorse di calcolo. Ad esempio, un intervallo come 10.0.0.0/24 e 10.0.1.0/24 è in genere sufficientemente ampio.

- Un VPC può avere un attributo di tenancy di istanza o predefinito o dedicato. Tutte le impostazioni predefinite VPCs hanno l'attributo di tenancy dell'istanza impostato su default e un VPC predefinito può supportare qualsiasi classe di istanza DB.

  Se scegli di avere il tuo cluster di DB in un VPC dedicato in cui l'attributo di tenancy dell'istanza è impostato su dedicato, la classe di istanza DB del tuo cluster di DB deve essere uno dei tipi di istanza EC2 dedicata Amazon approvati. Ad esempio, l'istanza EC2 dedicata r5.large corrisponde alla classe di istanza DB db.r5.large. Per informazioni sulla tenancy di istanza in un VPC, consulta Istanze dedicate nella Guida per l'utente Amazon Elastic Compute Cloud.

  Per ulteriori informazioni sui tipi di istanze che possono essere presenti in un'istanza dedicata, consulta le istanze EC2 dedicate di Amazon nella pagina EC2 dei prezzi di Amazon.

  > **Note**
  >
  > Quando si imposta l'attributo di locazione dell'istanza su dedicato per un cluster database, non è garantita l'esecuzione del cluster database su un host dedicato.

Utilizzo di gruppi di sottoreti database

Le sottoreti sono segmenti di un intervallo di indirizzi IP di un VPC che si designa per raggruppare le risorse in base alle esigenze operative e di sicurezza. Un gruppo di sottoreti DB è una raccolta di sottoreti (generalmente private) creata in un VPC e che è possibile indicare per i cluster database. Utilizzando un gruppo di sottoreti DB, è possibile specificare un particolare VPC durante la creazione di cluster di istanze utilizzando AWS CLI l'API o RDS. Se utilizzi la console, puoi scegliere il VPC e i gruppi di sottorete che desideri usare.

Ogni gruppo di sottoreti database deve avere almeno due zone di disponibilità in una determinata Regione AWS. Quando si crea un cluster database in un VPC, è necessario selezionare anche un gruppo di sottoreti DB. Dal gruppo di sottoreti DB, Amazon Aurora sceglie una sottorete e un indirizzo IP all'interno di essa da associare all'istanza database primaria nel cluster database. Il database utilizza la zona di disponibilità che contiene la sottorete. Aurora Amazon assegna sempre un indirizzo IP da una sottorete con spazio libero per gli indirizzi IP.

Le sottoreti di un gruppo di sottoreti DB sono pubbliche o private. Le sottoreti sono pubbliche o private, a seconda della configurazione impostata per le relative liste di controllo degli accessi alla rete (rete) e le tabelle di routing. ACLs Affinché un cluster database possa essere accessibile a livello pubblico, tutte le sottoreti nel gruppo di sottoreti database devono essere pubbliche. Se una sottorete

associata a un cluster database accessibile pubblicamente cambia da pubblica a privata, ciò può avere ripercussioni sulla disponibilità del cluster database.

Per creare un gruppo di sottoreti DB che supporti la modalità dual-stack, assicuratevi che a ogni sottorete che aggiungete al gruppo di sottorete DB sia associato un blocco CIDR del protocollo Internet versione 6 (). IPv6 Per ulteriori informazioni, consulta Assegnazione di indirizzi IP in Amazon Aurora la sezione «Migrazione verso» IPv6 nella Amazon VPC User Guide.

Quando Amazon Aurora crea un cluster database in un VPC, assegna un'interfaccia di rete al cluster database usando un indirizzo IP dal gruppo di sottoreti del database. Tuttavia, consigliamo vivamente di usare il nome del sistema dei nomi di dominio (DNS) per collegare il cluster database, perché l'indirizzo IP sottostante varia durante un failover.

> **Note**
>
> Per ogni cluster database in esecuzione in un VPC, assicurati di riservare almeno un indirizzo in ogni sottorete nel gruppo di sottoreti DB per l'utilizzo da parte di Amazon Aurora per le operazioni di ripristino.

Sottoreti condivise

Puoi creare un cluster database in un VPC condiviso.

Alcune considerazioni da tenere a mente durante l'utilizzo di shared: VPCs

- È possibile spostare un cluster database da una sottorete VPC condivisa a una sottorete VPC non condivisa e viceversa.
- I membri di un VPC condiviso devono creare un gruppo di sicurezza nel VPC per consentire loro di creare un cluster database.
- I proprietari e i membri di un VPC condiviso possono accedere al database utilizzando le query SQL. Tuttavia, solo il creatore di una risorsa può effettuare chiamate API sulla risorsa.

Assegnazione di indirizzi IP in Amazon Aurora

Gli indirizzi IP permettono alle risorse nel VPC di comunicare tra loro e con le risorse su Internet. Amazon Aurora supporta IPv4 entrambi i protocolli di IPv6 indirizzamento. Per impostazione

predefinita, Amazon Aurora e Amazon VPC utilizzano il protocollo di indirizzamento. IPv4 Non puoi disattivare questo comportamento. Quando crei un VPC, assicurati di specificare un blocco IPv4 CIDR (un intervallo di indirizzi privati IPv4).

Il supporto per il IPv6 protocollo amplia il numero di indirizzi IP supportati. Utilizzando il IPv6 protocollo, ti assicuri di disporre di indirizzi disponibili sufficienti per la crescita futura di Internet. Le risorse RDS nuove ed esistenti possono utilizzare IPv4 e IPv6 indirizzare all'interno del tuo VPC. La configurazione, la protezione e la traduzione del traffico di rete tra i due protocolli utilizzati in diverse parti di un'applicazione può causare sovraccarico operativo. Puoi standardizzare il IPv6 protocollo per le risorse Amazon RDS per semplificare la configurazione di rete.

Argomenti

- IPv4 indirizzi
- IPv6 indirizzi
- Modalità dual-stack

IPv4 indirizzi

Quando si crea un VPC, è necessario specificare un intervallo di IPv4 indirizzi per il VPC sotto forma di blocco CIDR, ad esempio. `10.0.0.0/16` Un gruppo di sottoreti DB definisce l'intervallo di indirizzi IP in questo blocco CIDR che può essere usato da cluster database. Questi indirizzi IP possono essere privati o pubblici.

Un IPv4 indirizzo privato è un indirizzo IP non raggiungibile su Internet. Puoi utilizzare IPv4 indirizzi privati per la comunicazione tra il tuo cluster di DB e altre risorse, come EC2 le istanze Amazon, nello stesso VPC. Ogni cluster database dispone di un indirizzo IP privato per la comunicazione nel VPC.

Un indirizzo IP pubblico è un IPv4 indirizzo raggiungibile da Internet. Puoi utilizzare gli indirizzi pubblici per la comunicazione tra cluster database e risorse su Internet, ad esempio un client SQL. Puoi controllare se cluster database ricevono un indirizzo IP pubblico.

Per un tutorial che mostra come creare un VPC con solo IPv4 indirizzi privati da utilizzare per uno scenario Amazon comune, consulta. Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4

IPv6 indirizzi

Facoltativamente, puoi associare un blocco IPv6 CIDR al tuo VPC e alle sottoreti e assegnare IPv6 gli indirizzi di quel blocco alle risorse del tuo VPC. Ogni indirizzo è unico a livello globale. IPv6

Il blocco IPv6 CIDR per il tuo VPC viene assegnato automaticamente dal pool IPv6 di indirizzi di Amazon. Non è possibile scegliere l'intervallo in modo autonomo.

Quando ti connetti a un IPv6 indirizzo, assicurati che siano soddisfatte le seguenti condizioni:

- Il client è configurato in modo da consentire il trasferimento del traffico tra IPv6 client e database.
- I gruppi di sicurezza RDS utilizzati dall'istanza DB sono configurati correttamente in modo da consentire il trasferimento del traffico tra client e database. IPv6
- Lo stack del sistema operativo client consente il traffico sull' IPv6 indirizzo e i driver e le librerie del sistema operativo sono configurati per scegliere l'endpoint di istanza DB predefinito corretto (uno o due IPv4). IPv6

Per ulteriori informazioni IPv6, consulta la sezione Indirizzamento IP nella Amazon VPC User Guide.

Modalità dual-stack

Quando un cluster di DB è in grado di comunicare sia tramite il protocollo di IPv4 IPv6 indirizzamento che tramite quello di indirizzamento, viene eseguito in modalità dual-stack. Pertanto, le risorse possono comunicare con il cluster di DB su o IPv4 entrambi IPv6. RDS disabilita l'accesso a Internet Gateway per gli IPv6 endpoint delle istanze DB private in modalità dual-stack. RDS esegue questa operazione per garantire che gli IPv6 endpoint siano privati e accessibili solo dall'interno del VPC.

Argomenti

- Modalità dual-stack e gruppi di sottoreti database
- Utilizzo di istanze database in modalità dual-stack
- Modifica dei cluster di IPv4 sole DB per utilizzare la modalità dual-stack
- Disponibilità di cluster database di rete dual-stack
- Limitazioni per cluster database di rete dual-stack

Per un tutorial che mostra come creare un VPC con entrambi IPv4 IPv6 gli indirizzi da utilizzare per uno scenario Amazon comune, consulta. Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Modalità dual-stack e gruppi di sottoreti database

Per utilizzare la modalità dual-stack, assicurati che a ogni sottorete del gruppo di sottoreti DB associato al cluster di DB sia associato un blocco CIDR. IPv6 Per soddisfare questo requisito, puoi

creare un nuovo gruppo di sottoreti database o modificare un gruppo di sottoreti database esistente. Quando cluster database sono in modalità dual-stack, i client possono connettersi normalmente. Assicurati che i firewall di sicurezza dei client e i gruppi di sicurezza delle istanze RDS DB siano configurati accuratamente per consentire il trasferimento del traffico. IPv6 Per connettersi, i client utilizzano l'endpoint dell'istanza principale del cluster database. Le applicazioni client possono specificare il protocollo preferito per la connessione a un database. In modalità dual-stack, il cluster di DB rileva il protocollo di rete preferito del client IPv6, IPv4 o lo utilizza per la connessione.

Se un gruppo di sottoreti database smette di supportare la modalità dual-stack a causa dell'eliminazione della sottorete o dell'annullamento dell'associazione con il blocco CIDR, si può verificare una situazione di stato di rete incompatibile per le istanze database associate al gruppo di sottoreti database. Inoltre, non puoi utilizzare il gruppo di sottoreti database quando crei un nuovo cluster database in modalità dual-stack.

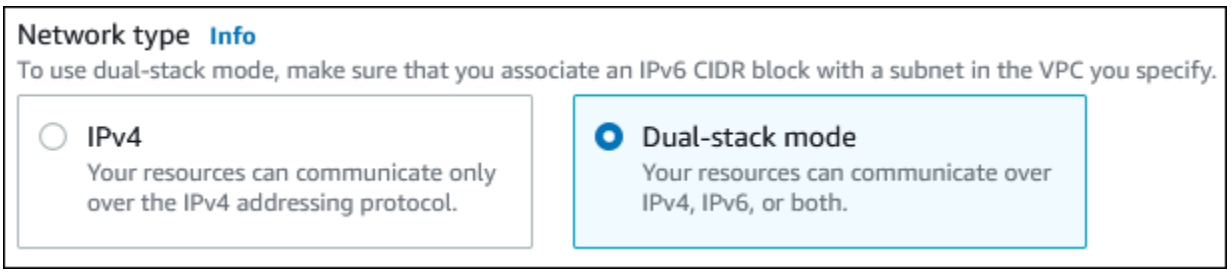
Per determinare se un gruppo di sottoreti DB supporta la modalità dual-stack utilizzando il AWS Management Console, visualizza il Tipo di rete nella pagina dei dettagli del gruppo di sottoreti DB. Per determinare se un gruppo di sottoreti DB supporta la modalità dual-stack utilizzando il AWS CLI, esegui il comando e visualizza nell'output. describe-db-subnet-groups`SupportedNetworkTypes`

Le repliche di lettura vengono trattate come istanze database indipendenti e possono avere un tipo di rete diverso dall'istanza database principale. Se si modifica il tipo di rete dell'istanza database principale di una replica di lettura, tale modifica non interessa la replica di lettura. Quando si ripristina un'istanza database, è possibile ripristinarla su qualsiasi tipo di rete supportato.

Utilizzo di istanze database in modalità dual-stack

Quando si crea o si modifica un cluster di DB, è possibile specificare la modalità dual-stack per consentire alle risorse di comunicare con il cluster di o entrambi. IPv4 IPv6

Quando si utilizza AWS Management Console per creare o modificare un'istanza DB, è possibile specificare la modalità dual-stack nella sezione Tipo di rete. L'immagine seguente mostra la sezione Network type (Tipo di rete) nella console.



Quando utilizzi la AWS CLI per creare o modificare un cluster di DB, imposta l'`--network-type`opzione per utilizzare la modalità `DUAL` dual-stack. Quando usi l'API RDS per creare o modificare un cluster database, per utilizzare la modalità dual-stack imposta il parametro `NetworkType` su `DUAL`. Quando modifichi il tipo di rete di un'istanza database, è possibile che si verifichi un periodo di inattività. Se la modalità dual-stack non è supportata dalla versione del motore del database o dal gruppo di sottoreti database in uso, viene restituito l'errore `NetworkTypeNotSupported`.

Ulteriori informazioni sulla creazione di un cluster di database sono disponibili in Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

Per determinare se un cluster database è in modalità dual-stack utilizzando la console, visualizza l'opzione Network type (Tipo di rete) nella scheda Connectivity & security (Connettività e sicurezza) per il cluster database in questione.

Modifica dei cluster di IPv4 sole DB per utilizzare la modalità dual-stack

A tale scopo, devi modificare il tipo di rete del cluster database. La modifica potrebbe comportare tempi di inattività.

Ti consigliamo di modificare il tipo di rete dei cluster Amazon Aurora durante una finestra di manutenzione. L'impostazione predefinita del tipo di rete delle nuove istanze sulla modalità dual stack non è al momento supportata. È possibile impostare il tipo di rete manualmente utilizzando il comando `modify-db-cluster`.

Prima di modificare un cluster database per utilizzare la modalità dual-stack, assicurati che il gruppo di sottoreti DB supporti la modalità dual-stack. Se il gruppo di sottoreti DB associato al cluster database non supporta la modalità dual-stack, specifica un gruppo di sottoreti database diverso che supporta tale modalità quando modifichi il cluster database. La modifica del gruppo di sottoreti di database di un cluster di database può causare tempi di inattività.

Se si modifica il gruppo di sottoreti di database di un cluster di database prima di modificare il cluster di database per l'utilizzo della modalità Dual stack, assicurati che il gruppo di sottoreti di database sia valido per il cluster di database prima e dopo la modifica.

Ti consigliamo di eseguire l'modify-db-clusterAPI solo con il `--network-type` parametro con valore `DUAL` per modificare la rete di un cluster Amazon Aurora in modalità dual-stack. L'aggiunta di altri parametri al parametro `--network-type` nella stessa chiamata API potrebbe causare tempi di inattività.

Se non riesci a connetterti al cluster di DB dopo la modifica, assicurati che i firewall di sicurezza del client e del database e le tabelle di routing siano configurati accuratamente per consentire il traffico verso il database sulla rete selezionata (IPv4 o IPv6). Potrebbe inoltre essere necessario modificare i parametri del sistema operativo, le librerie o i driver per connetterti utilizzando un IPv6 indirizzo.

Per modificare un cluster IPv4 di sole DB per utilizzare la modalità dual-stack

1. Modificare un gruppo di sottoreti database per supportare la modalità dual-stack o creare un gruppo di sottoreti database che supporti la modalità dual-stack:

   a. Associa un blocco IPv6 CIDR al tuo VPC.

      Per istruzioni, consulta Aggiungere un blocco IPv6 CIDR al tuo VPC nella Amazon VPC User Guide.

   b. Collega il blocco IPv6 CIDR a tutte le sottoreti del gruppo di sottoreti DB.

      Per istruzioni, consulta Aggiungere un blocco IPv6 CIDR alla sottorete nella Amazon VPC User Guide.

   c. Verificare che il gruppo di sottoreti database supporti la modalità dual-stack.

      Se utilizzi il AWS Management Console, seleziona il gruppo di sottoreti DB e assicurati che il valore Supported network types sia Dual,. IPv4

      Se utilizzi il AWS CLI, esegui il describe-db-subnet-groupscomando e assicurati che il `SupportedNetworkType` valore per l'istanza DB sia`Dual, IPv4`.

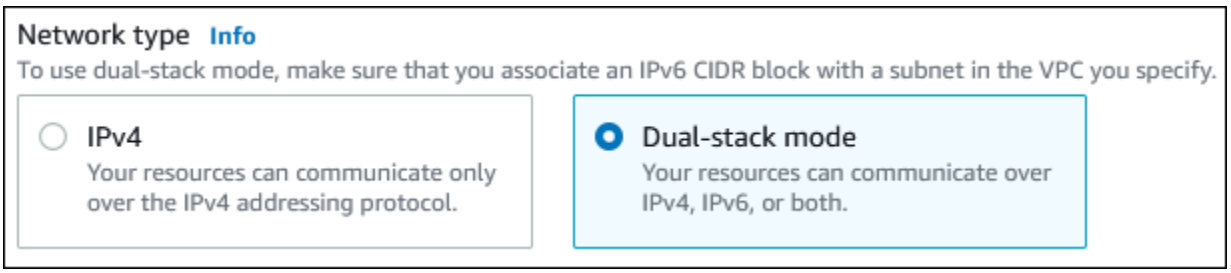
2. Modifica il gruppo di sicurezza associato al cluster di DB per consentire IPv6 le connessioni al database o crea un nuovo gruppo di sicurezza che consenta IPv6 le connessioni.

   Per le istruzioni, vedere Regole del gruppo di sicurezza nella Guida per l'utente di Amazon VPC.

3. Modifica il cluster database in modo che supporti la modalità dual-stack. A questo scopo, imposta l'opzione Network type (Tipo di rete) su Dual-stack mode (Modalità dual-stack).

   In caso di utilizzo della console, accertati che le impostazioni seguenti siano corrette:

   - Tipo di rete–Dual-stack mode (Modalità dual-stack)



- DB subnet group (Gruppo di sottoreti DB): gruppo di sottoreti database configurato in un passaggio precedente
- Security group (Gruppo di sicurezza): gruppo di sicurezza di default configurato in un passaggio precedente

Se utilizzi la AWS CLI, assicurati che le seguenti impostazioni siano corrette:

- `--network-type` – `dual`
- `--db-subnet-group-name`: gruppo di sottoreti database configurato in un passaggio precedente
- `--vpc-security-group-ids`: gruppo di sicurezza VPC configurato in un passaggio precedente

Per esempio:

```
aws rds modify-db-cluster --db-cluster-identifier my-cluster --network-type "DUAL"
```

4. Verifica che il cluster database supporti la modalità dual-stack.

   Se utilizzi la console, scegli la scheda Configuration (Configurazione) per il cluster database. In quella scheda, assicurati che il valore dell'opzione Network type (Tipo di rete) sia Dual-stack mode (Modalità dual-stack).

   Se utilizzi il AWS CLI, esegui il describe-db-clusterscomando e assicurati che il `NetworkType` valore per il cluster DB sia`dual`.

   Esegui il `dig` comando sull'endpoint dell'istanza Writer DB per identificare l' IPv6indirizzo ad esso associato.

   ```
   dig db-instance-endpoint AAAA
   ```

Utilizza l'endpoint dell'istanza Writer DB, non l' IPv6 indirizzo, per connetterti al cluster di DB.

Disponibilità di cluster database di rete dual-stack

I cluster DB di rete dual-stack sono disponibili in tutti Regioni AWS gli ambienti ad eccezione dei seguenti:

- Asia Pacific (Hyderabad)
- Asia Pacifico (Malesia)
- Asia Pacifico (Melbourne)
- Canada occidentale (Calgary)
- Europa (Spagna)
- Europa (Zurigo)
- Israele (Tel Aviv)
- Medio Oriente (Emirati Arabi Uniti)

Le seguenti versioni del motore DB supportano i cluster DB di rete dual-stack:

- Versioni Aurora MySQL:
  - 3.02 o versioni successive alla 3
  - 2.09.1 o versioni successive alla 2

  Per ulteriori informazioni sulle versioni di Aurora MySQL, consulta la sezione Note di rilascio di Aurora MySQL.
- Versioni di Aurora PostgreSQL:
  - 15.2 e tutte le versioni successive
  - 14.3 o versioni successive alla 14
  - 13.7 o versioni successive alla 13

  Per ulteriori informazioni sulle versioni di Aurora PostgreSQL, consulta Note di rilascio di Aurora PostgreSQL.

Limitazioni per cluster database di rete dual-stack

Le seguenti limitazioni si applicano ai cluster database di rete dual-stack:

- I cluster di DB non possono utilizzare esclusivamente il IPv6 protocollo. Possono utilizzare IPv4 esclusivamente oppure possono utilizzare il IPv6 protocollo IPv4 and (modalità dual-stack).
- Amazon RDS non supporta IPv6 sottoreti native.
- I cluster database che utilizzano la modalità dual-stack devono essere di tipo privato. Non possono essere accessibili pubblicamente.
- Non è possibile utilizzare il proxy RDS con cluster database in modalità dual-stack.

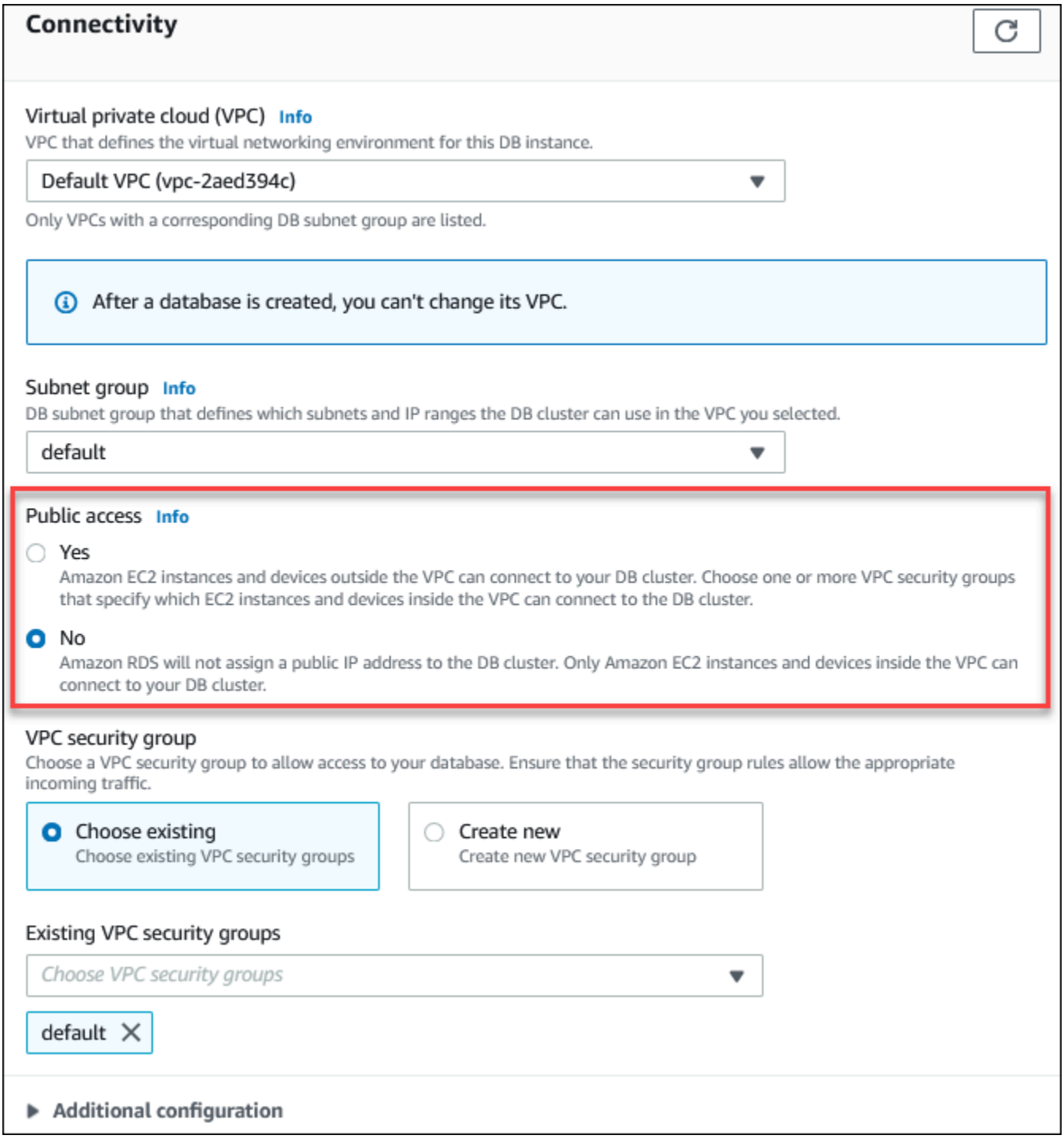
Nascondere cluster database in un VPC da Internet

Uno scenario comune di Amazon Aurora consiste nell'avere un VPC in cui è presente un'istanza EC2 Amazon con un'applicazione Web rivolta al pubblico e un cluster di istanze non accessibile al pubblico. Ad esempio, puoi creare un VPC con una sottorete pubblica e una sottorete privata. EC2 le istanze che funzionano come server Web possono essere distribuite nella sottorete pubblica. L'implementazione di cluster database viene invece eseguita nella sottorete privata. In tale implementazione, solo i server Web hanno accesso ai cluster database. Per un'illustrazione di questo scenario, consulta Un cluster di DB in un VPC ambiente accessibile da un'EC2istanza Amazon nello stesso VPC.

Quando si avvia un cluster database all'interno di un VPC, il cluster database dispone di un indirizzo IP privato per il traffico all'interno del VPC. Questo indirizzo IP privato non è accessibile pubblicamente. Puoi utilizzare l'opzione Public access (Accesso pubblico) per indicare se il cluster database dispone anche di un indirizzo IP pubblico oltre all'indirizzo IP privato. Se il cluster database è definito come accessibile pubblicamente, il relativo endpoint DNS utilizza l'indirizzo IP privato dall'interno del VPC. Utilizza invece l'indirizzo IP pubblico dall'esterno del VPC. L'accesso al cluster database è in ultima analisi controllato dal gruppo di sicurezza in uso. Questo accesso pubblico non è consentito se il gruppo di sicurezza assegnato al cluster database non include regole in entrata che lo consentono. Per un cluster database che deve essere accessibile pubblicamente, le sottoreti nel relativo gruppo di sottoreti DB devono disporre di un gateway Internet. Per ulteriori informazioni, consulta Impossibile connettersi all'istanza database di Amazon RDS

Puoi modificare un cluster database per attivare o disattivare l'accessibilità pubblica modificando l'opzione Public access (Accesso pubblico). Nella figura seguente viene illustrata l'opzione Public access (Accesso pubblico) nella sezione Additional connectivity configuration (Configurazioni di

connettività aggiuntiva) . Per impostare l'opzione, apri la sezione Additional connectivity configuration (Configurazioni di connettività aggiuntiva) nella sezione Connectivity (Connettività) .



Per informazioni sulla modifica di un'istanza database per impostare l'opzione Public access (Accesso pubblico), consulta Modifica di un'istanza database in un cluster database.

Creazione di un cluster database in un VPC

Le procedure seguenti aiutano a creare un cluster database in un VPC. Per utilizzare il VPC predefinito, puoi iniziare con il passaggio 2 e utilizzare il VPC e il gruppo di sottoreti DB creati automaticamente. Se desideri creare un VPC aggiuntivo, puoi creare un nuovo VPC.

> **Note**
>
> Se desideri che il cluster database nel VPC sia pubblicamente accessibile, devi aggiornare le informazioni DNS per il VPC attivando gli attributi VPC DNS hostnames (Nomi host DNS) e DNS resolution (Risoluzione DNS). Per informazioni sull'aggiornamento delle informazioni DNS per un'istanza VPC, consulta Aggiornamento del supporto DNS per il VPC.

Segui questa procedura per creare un'istanza database in un VPC:

- Fase 1. Creazione di un VPC
- Fase 2: creazione di un gruppo di sottoreti database
- Fase 3: creazione di un gruppo di sicurezza VPC
- Passaggio 4: creazione di un'istanza database nel VPC

Fase 1. Creazione di un VPC

Crea un VPC con sottoreti in almeno due zone di disponibilità. Usi queste sottoreti quando crei un gruppo di sottoreti database. Se disponi di un VPC di default, viene creata automaticamente una sottorete in ciascuna zona di disponibilità nella Regione AWS.

Per ulteriori informazioni, consulta Creazione di un VPC con sottoreti pubbliche e private oppure Creazione di un VPC nella Guida per l'utente di Amazon VPC..

Fase 2: creazione di un gruppo di sottoreti database

Un gruppo di sottoreti DB è una raccolta di sottoreti (generalmente private) creata per un VPC e che è possibile definire per i cluster database. Un gruppo di sottoreti DB consente di specificare un particolare VPC quando si creano cluster di istanze utilizzando AWS CLI l'API o RDS. Se utilizzi la console, puoi scegliere il VPC e le sottoreti che desideri usare. Ogni gruppo di sottoreti database deve avere almeno una sottorete in almeno due zone di disponibilità nella Regione AWS. Come best

practice, ogni gruppo di sottoreti database deve disporre di almeno una sottorete per ogni zona di disponibilità nella Regione AWS.

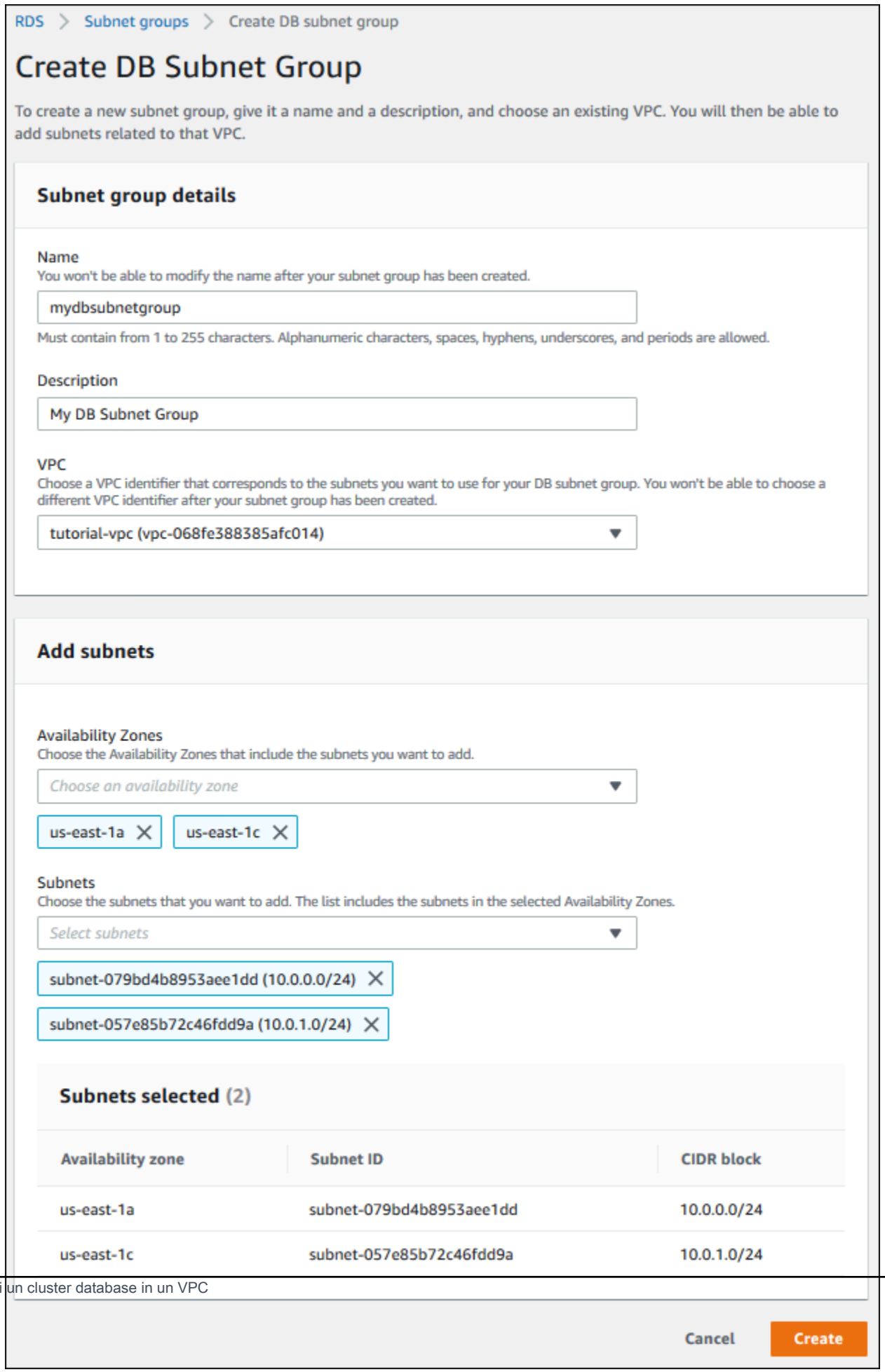
Per un cluster database che deve essere accessibile pubblicamente, le sottoreti nel gruppo di sottoreti database devono disporre di un gateway Internet. Per ulteriori informazioni sui gateway Internet, consulta Eseguire la connessione a Internet utilizzando un gateway Internet nella Guida per l'utente di Amazon VPC.

Quando crei un cluster database in un VPC, puoi selezionare un gruppo di sottoreti DB. Amazon Aurora sceglie una sottorete e un indirizzo IP al suo interno da associare al cluster database. Se non esistono gruppi di sottoreti DB, Amazon Aurora crea un gruppo di sottoreti predefinito quando crei un cluster database. Amazon Aurora crea e associa un'interfaccia di rete elastica al cluster database con tale indirizzo IP. Il cluster database utilizza la zona di disponibilità contenente la sottorete.

In questo passaggio, si crea un gruppo di sottoreti database e si aggiungono le sottoreti create per il VPC.

Creare un gruppo di sottoreti database

1. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
3. Scegli Create DB Subnet Group (Crea gruppo di sottoreti del database).
4. Per Name (Nome), digita il nome del gruppo di sottoreti database.
5. Per Description Descrizione), digita una descrizione per il gruppo di sottoreti database.
6. In VPC, scegli il VPC predefinito o il VPC creato in precedenza.
7. Nella sezione Aggiungi sottoreti, scegliere le zone di disponibilità che includono le sottoreti da Zone di disponibilità, quindi scegliere le sottoreti da Sottoreti.


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

8. Scegliere Create (Crea).

   Il nuovo gruppo di sottoreti database viene visualizzato nell'elenco dei gruppi di sottoreti database sulla console RDS. Puoi scegliere il gruppo di sottoreti database per visualizzare i dettagli, comprese tutte le sottoreti associate al gruppo, nel riquadro dei dettagli nella parte inferiore della finestra.

Fase 3: creazione di un gruppo di sicurezza VPC

Prima di creare un cluster database, devi creare un gruppo di sicurezza VPC da associare tale cluster database. Se non crei un gruppo di sicurezza VPC, puoi utilizzare il gruppo di sicurezza predefinito quando crei un cluster database. Per istruzioni su come creare un gruppo di sicurezza per il cluster database, consulta Creazione di un gruppo di sicurezza VPC per un cluster database privato oppure Controlla il traffico verso le risorse utilizzando gruppi di sicurezza nella Guida per l'utente di Amazon VPC.

Passaggio 4: creazione di un'istanza database nel VPC

In questo passaggio, si crea un cluster database e si utilizza il nome VPC, il gruppo di sottoreti DB e il gruppo di sicurezza VPC creato nel passaggio precedente.

> **Note**
>
> Se desideri che il cluster database nel VPC sia pubblicamente accessibile, devi abilitare gli attributi VPC DNS hostnames (Nomi host DNS) e DNS resolution (Risoluzione DNS). Per ulteriori informazioni, consulta Attributi DNS per il VPC nella Guida per l'utente di Amazon VPC.

Per informazioni dettagliate su come creare un cluster di database, consulta Creazione di un cluster database Amazon Aurora.

Quando richiesto nella sezione Connectivity (Connettività), inserisci il nome VPC, il gruppo di sottoreti DB e il gruppo di sicurezza VPC.

> **Note**
>
> L'aggiornamento VPCs non è attualmente supportato per i cluster Aurora DB.

Scenari per l'accesso a un cluster di DB in un VPC

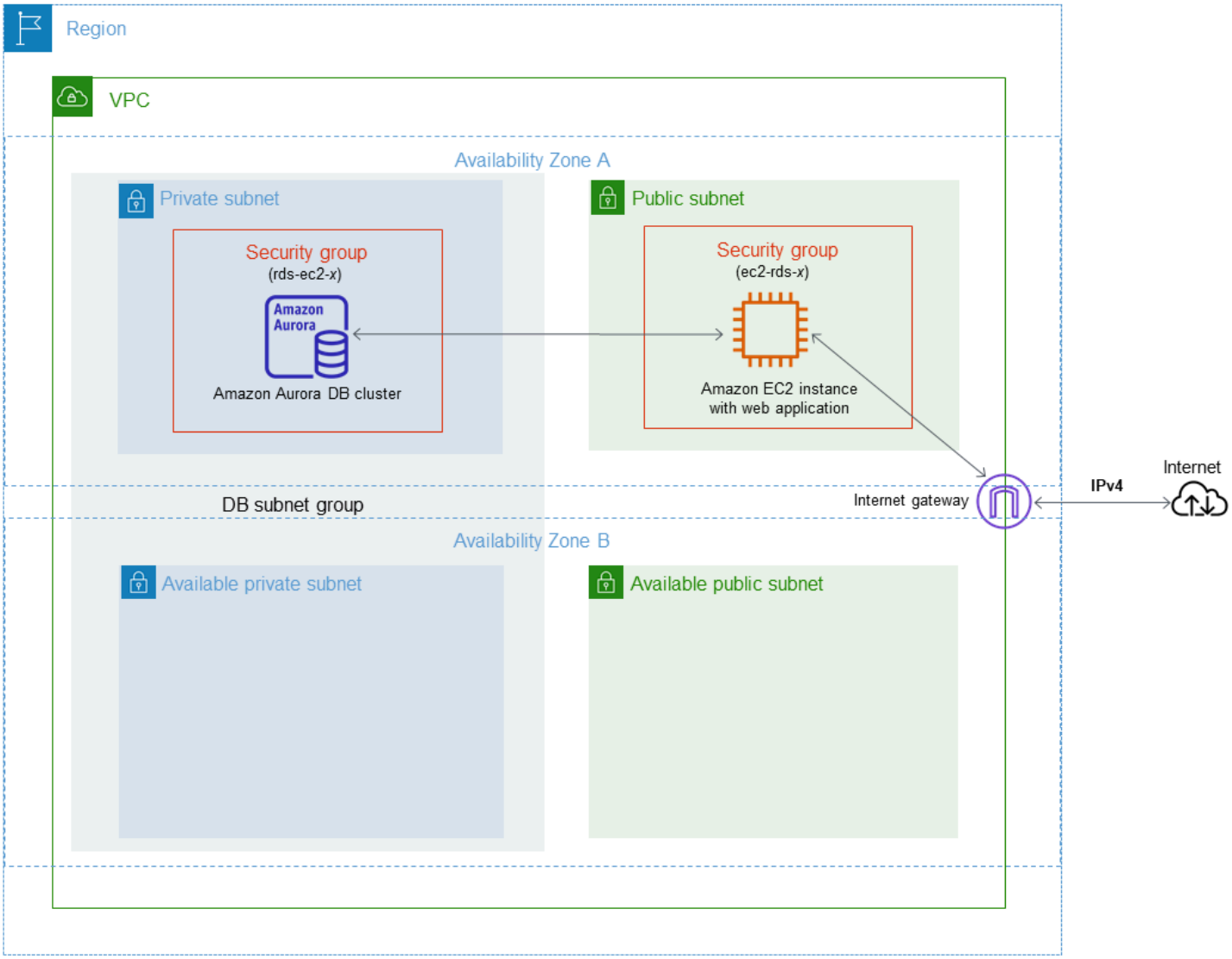
Amazon Aurora supporta i seguenti scenari per l'accesso a un cluster di DB in un: VPC

- Un'EC2istanza Amazon nella stessa VPC
- Un'EC2istanza in un altro VPC
- Un'applicazione client tramite internet
- Una rete privata

Un cluster di DB in un VPC ambiente accessibile da un'EC2istanza Amazon nello stesso VPC

Un uso comune di un cluster di DB in a VPC consiste nel condividere i dati con un server di applicazioni in esecuzione in un'EC2istanza Amazon nella stessa istanzaVPC.

Il seguente diagramma mostra questo scenario.



Il modo più semplice per gestire l'accesso tra EC2 istanze e cluster di DB all'interno degli stessi cluster VPC consiste nel fare quanto segue:

- Crea un gruppo VPC di sicurezza per i cluster DB. Questo gruppo di sicurezza può essere usato per limitare l'accesso ai cluster database. Ad esempio, puoi creare una regola personalizzata per questo gruppo di sicurezza. Ciò potrebbe consentire TCP l'accesso utilizzando la porta assegnata al cluster di DB al momento della creazione e un indirizzo IP utilizzato per accedere al cluster di DB per lo sviluppo o per altri scopi.
- Crea un gruppo VPC di sicurezza per le tue EC2 istanze (server Web e client). Questo gruppo di sicurezza può, se necessario, consentire l'accesso all'EC2istanza da Internet utilizzando la tabella VPC di routing. Ad esempio, è possibile impostare regole su questo gruppo di sicurezza per consentire TCP l'accesso all'EC2istanza tramite la porta 22.

- Crea regole personalizzate nel gruppo di sicurezza per i cluster DB che consentono le connessioni dal gruppo di sicurezza che hai creato per le tue EC2 istanze. Queste regole potrebbero consentire a qualsiasi membro del gruppo di sicurezza di accedere ai cluster database.

È disponibile una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Un gruppo di sottoreti RDS DB richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica il passaggio a un'implementazione Multi-AZ di un'istanza DB in futuro.

Per un tutorial che mostra come creare una VPC con sottoreti pubbliche e private per questo scenario, vedi. Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4

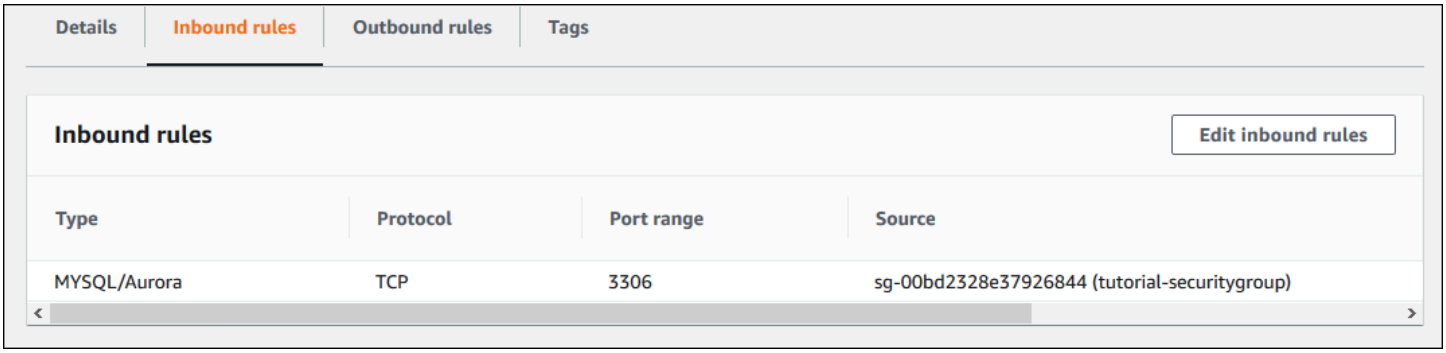
> **Tip**
>
> Puoi configurare automaticamente la connettività di rete tra un'EC2istanza Amazon e un cluster di DB quando crei il cluster di DB. Per ulteriori informazioni, consulta Configura la connettività di rete automatica con un'istanza EC2 .

Per creare una regola in un gruppo VPC di sicurezza che consenta le connessioni da un altro gruppo di sicurezza, procedi come segue:

1. Accedi a AWS Management Console e apri la VPC console Amazon all'indirizzo https://console.aws.amazon.com/vpc.
2. Fai clic su Security Groups (Gruppi di sicurezza) nel pannello di navigazione.
3. Scegli o crea un gruppo di sicurezza per il quale desideri concedere l'accesso ai membri di un altro gruppo di sicurezza. Nello scenario precedente, questo è il gruppo di sicurezza utilizzato per i cluster database. Seleziona la scheda Regole in entrata, quindi scegli Modifica regola.
4. Nella scheda Modifica regole in entrata, seleziona Aggiungi regola.
5. Per Tipo, scegli la voce che corrisponde alla porta utilizzata quando hai creato il cluster di / Aurora.
6. Nella casella Origine iniziare a digitare l'ID del gruppo di sicurezza, che elenca i gruppi di sicurezza corrispondenti. Scegli il gruppo di sicurezza con i membri che desideri abbiano accesso alle risorse protette da questo gruppo di sicurezza. Nello scenario precedente, questo è il gruppo di sicurezza utilizzato per l'istanza. EC2

7. Se necessario, ripeti i passaggi del TCP protocollo creando una regola con Tutto TCP come Tipo e il gruppo di sicurezza nella casella Origine. Se intendi utilizzare il UDP protocollo, crea una regola con All UDP come Type e il tuo gruppo di sicurezza in Source.
8. Scegliere Salva regole.

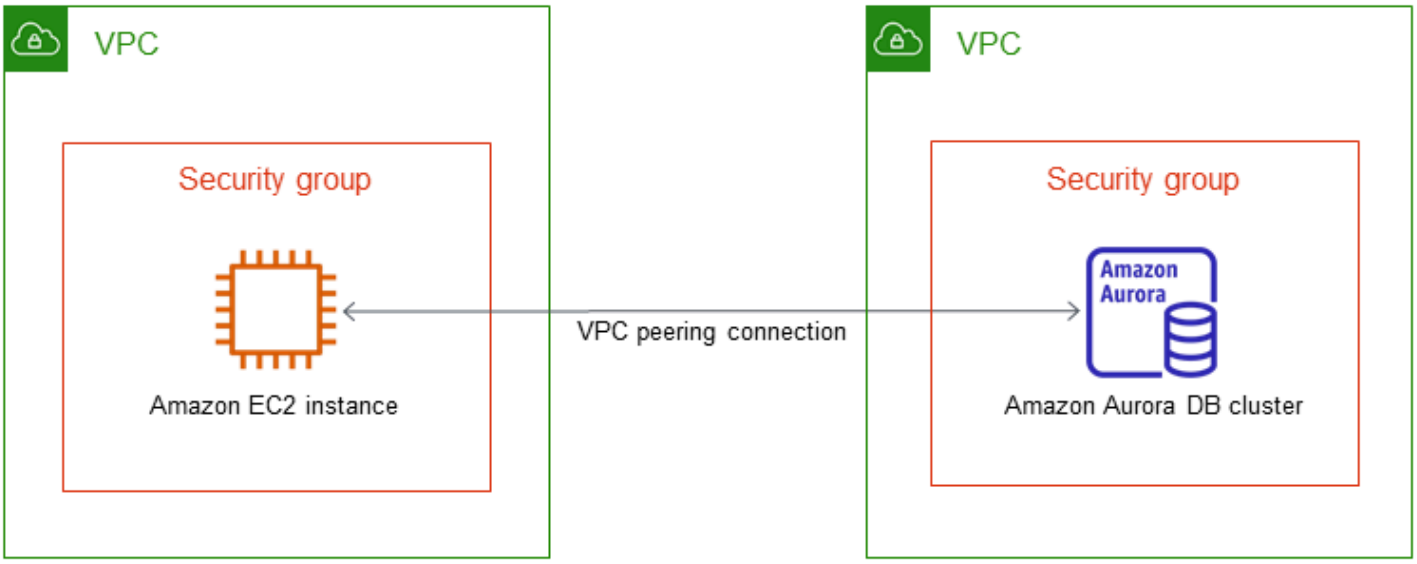
Nella schermata seguente viene illustrata una regola in entrata con un gruppo di sicurezza per la relativa origine.



Per ulteriori informazioni sulla connessione all' del cluster DB dalla propria EC2 istanza, consulta .

Un cluster di DB in un'istanza a VPC cui accede un'EC2istanza in un'altra VPC

Il seguente diagramma mostra questo scenario.



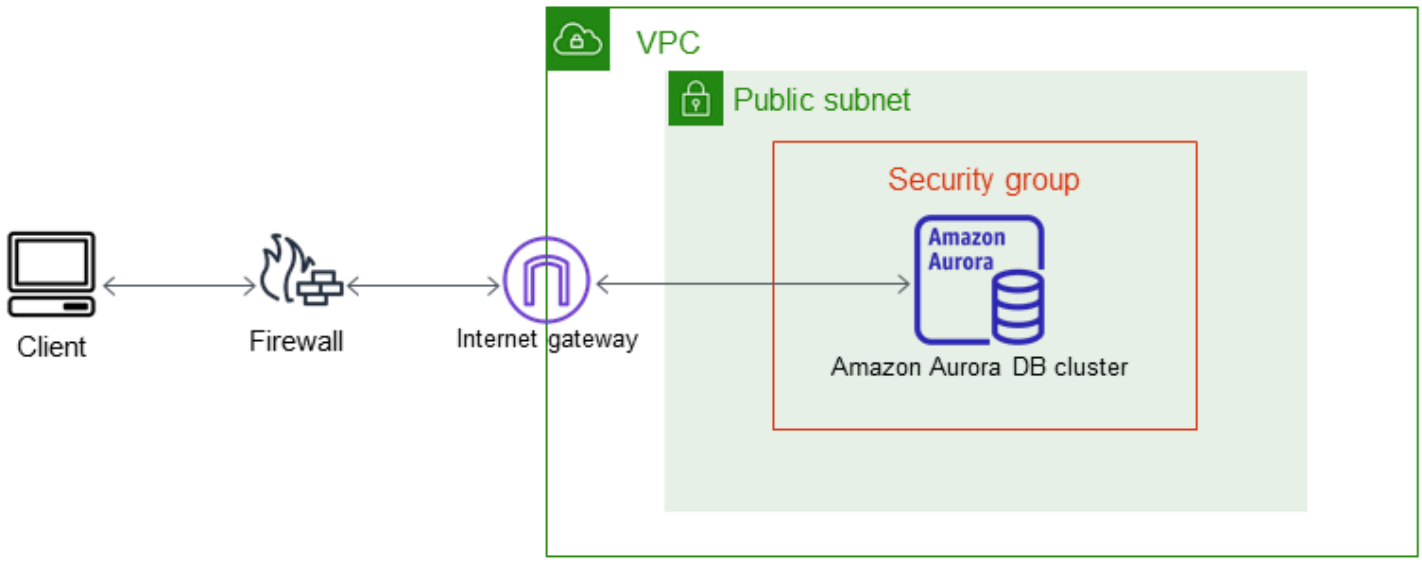
Una connessione VPC peering è una connessione di rete tra due VPCs che consente di instradare il traffico tra di esse utilizzando indirizzi IP privati. Le risorse di entrambe VPC possono comunicare tra loro come se si trovassero all'interno della stessa rete. È possibile creare una connessione VPC

peering tra il proprio accountVPCs, con un VPC altro AWS account o con un VPC altro Regione AWS. Per ulteriori informazioni sul VPC peering, consulta la sezione VPCpeering nella Amazon Virtual Private Cloud User Guide.

Un cluster di DB in un VPC ambiente accessibile da un'applicazione client tramite Internet

Per accedere ai cluster di DB in un VPC da un'applicazione client tramite Internet, è necessario configurare un'VPCunica sottorete pubblica e un gateway Internet per consentire la comunicazione su Internet.

Il seguente diagramma mostra questo scenario.



È consigliabile utilizzare la seguente configurazione:

- A VPC di dimensione /16 (ad esempioCIDR: 10.0.0.0/16). Questa dimensione fornisce indirizzi 65.536 indirizzi IP privati.
- Una sottorete di dimensione /24 (ad esempio: 10.0.0.0/24). CIDR Questa dimensione fornisce 256 indirizzi IP privati.
- Un cluster di Aurora DB associato alla sottorete VPC e alla sottorete. Amazon RDS assegna un indirizzo IP all'interno della sottorete al cluster di DB.
- Un gateway Internet che collega Internet e VPC ad altri AWS prodotti.
- Un gruppo di sicurezza associato al cluster database. Le regole in entrata del gruppo di sicurezza consentono all'applicazione client di accedere al cluster database.

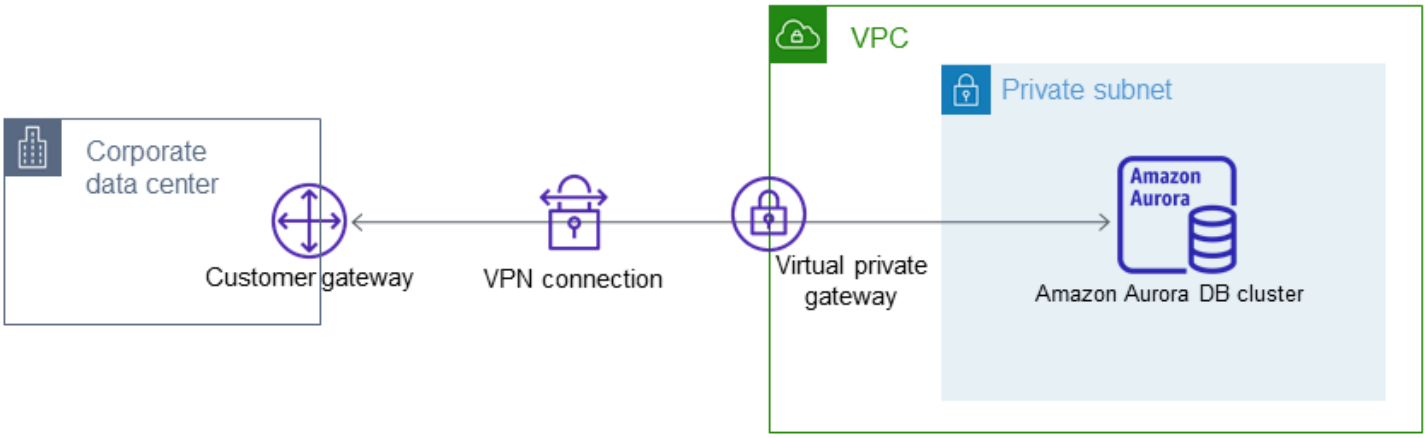
Per informazioni sulla creazione di cluster di DB in aVPC, vedere. Creazione di un cluster database in un VPC

Un cluster di DB in una rete a VPC cui si accede da una rete privata

Se il cluster database non è accessibile pubblicamente, sono disponibili le seguenti opzioni per consentire l'accesso da una rete privata:

- Una AWS Site-to-Site VPN connessione. Per ulteriori informazioni, consulta What is AWS Site-to-Site VPN?
- Una AWS Direct Connect connessione. Per ulteriori informazioni, vedi Cos'è AWS Direct Connect?
- Una AWS Client VPN connessione. Per ulteriori informazioni, consulta What is AWS Client VPN?

Il diagramma seguente mostra uno scenario con una AWS Site-to-Site VPN connessione.

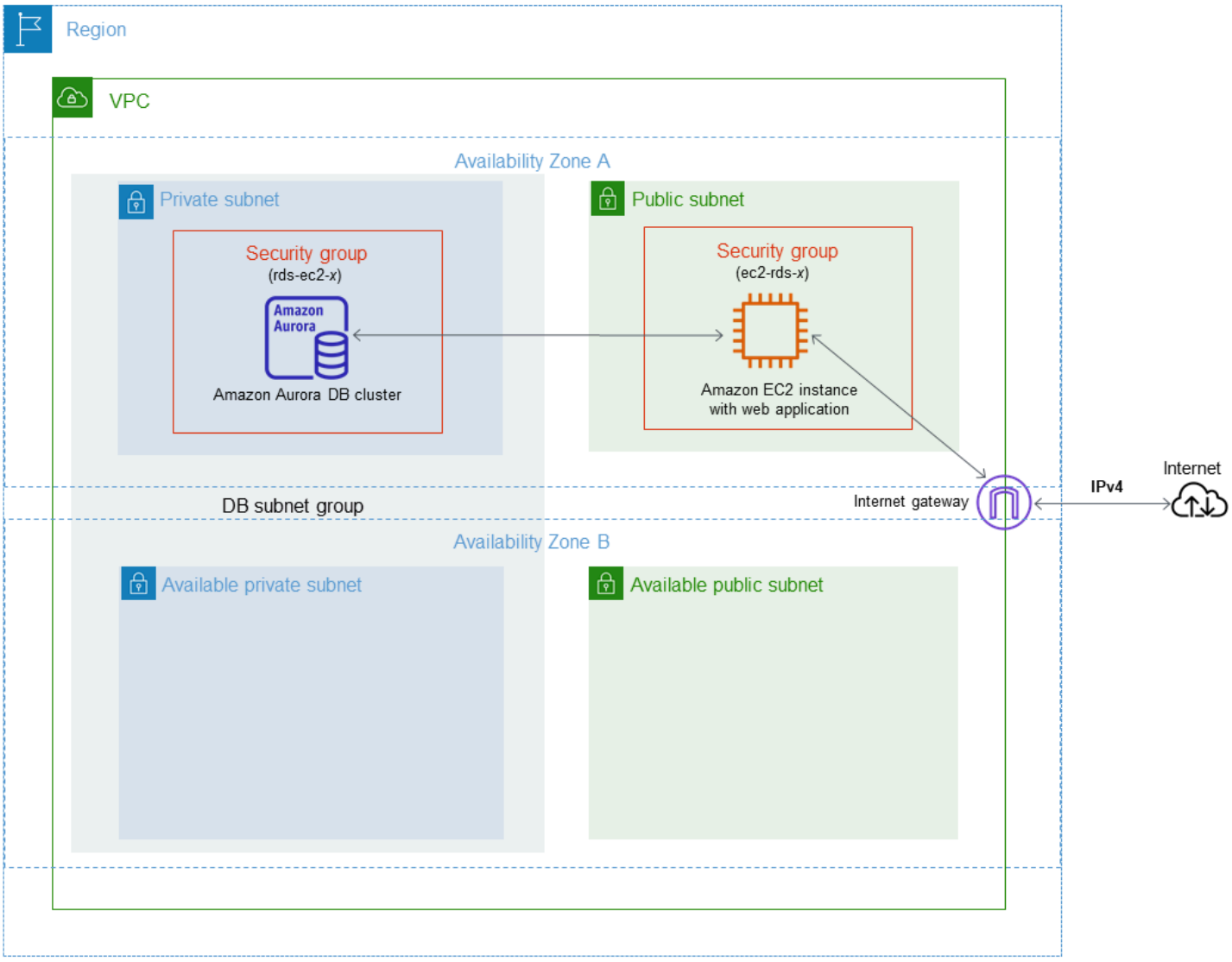


Per ulteriori informazioni, consulta Riservatezza del traffico Internet.

Tutorial: crea un VPC da utilizzare con un cluster di DB (solo) IPv4

Uno scenario comune include un cluster database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Questo VPC condivide i dati con un server Web in esecuzione nello stesso VPC. In questo tutorial eseguirai la creazione del VPC in questo scenario.

Il seguente diagramma mostra questo scenario. Per informazioni su altri scenari, consulta Scenari per l'accesso a un cluster di DB in un VPC.



Il cluster database deve essere disponibile solo per il server Web e non per la rete Internet pubblica. Pertanto, occorre creare un VPC con sottoreti pubbliche e private. Il server Web è ospitato nella sottorete pubblica, in modo da poter raggiungere l'Internet pubblico. Il cluster database è ospitato in una sottorete privata. Il server Web può connettersi al cluster database perché è ospitato nello stesso

VPC. Tuttavia, il cluster database non è disponibile per la rete Internet pubblica e ciò garantisce una maggiore sicurezza.

Questo tutorial configura una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Queste sottoreti non vengono utilizzate dal tutorial. Un gruppo di sottoreti DB RDS richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica la configurazione di più istanze database Aurora.

In questa esercitazione viene descritta la configurazione di un VPC per cluster di database Amazon Aurora. Per un'esercitazione che illustra come creare un server Web per questo scenario VPC, consulta Tutorial: creazione di un server Web e un cluster database Amazon Aurora. Per ulteriori informazioni su Amazon VPC, consulta Guida alle operazioni di base di Amazon VPC e Guida per l'utente di Amazon VPC.

> **Tip**
>
> Puoi configurare automaticamente la connettività di rete tra un' EC2 istanza Amazon e un cluster di DB quando crei il cluster di DB. La configurazione di rete è simile a quella descritta in questo tutorial. Per ulteriori informazioni, consulta Configura la connettività di rete automatica con un'istanza EC2 .

Creazione di un VPC con sottoreti pubbliche e private

Utilizza la procedura seguente per creare un VPC con sottoreti pubbliche e private.

Per creare un VPC e sottoreti

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli la regione in cui creare il tuo VPC. In questo esempio viene utilizzata la regione Stati Uniti occidentali (Oregon).
3. Nell'angolo in alto a sinistra, scegli VPC Dashboard (Pannello di controllo VPC). Per iniziare a creare un VPC, scegli Create VPC (Crea VPC).
4. In Resources to create (Risorse da creare) nell'area VPC settings (Impostazioni VPC), scegli VPC and more (VPC e altro).
5. In VPC settings (Impostazioni VPC) restanti, imposta i seguenti valori:
   - Name tag auto-generation (Generazione automatica del tag del nome): **`tutorial`**

- IPv4 Blocco CIDR: **`10.0.0.0/16`**
- IPv6 Blocco CIDR: nessun IPv6 blocco CIDR
- Tenancy (Locazione): Default
- Numero di zone di disponibilità (AZs) — 2
- Personalizza AZs: mantiene i valori predefiniti.
- Number of public subnet (Numero di sottoreti pubbliche): 2
- Number of private subnets (Numero di sottoreti private): 2
- Customize subnets CIDR blocks (Personalizza blocchi CIDR delle sottoreti): mantieni i valori predefiniti.
- NAT gateways ($) (Gateway NAT): nessuna
- VPC endpoints (Endpoint VPC): nessuno
- DNS options (Opzioni DNS): mantieni i valori predefiniti.

6. Seleziona Create VPC (Crea VPC).

Creazione di un gruppo di sicurezza VPC per un server Web pubblico

È possibile a questo punto aggiungere un gruppo di sicurezza per l'accesso pubblico. Per connetterti alle EC2 istanze pubbliche nel tuo VPC, aggiungi regole in entrata al tuo gruppo di sicurezza VPC. Queste consentono al traffico di connettersi da Internet.

Per creare un gruppo di sicurezza VPC

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-securitygroup`**
   - Descrizione: **`Tutorial Security Group`**
   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc- (tutorial-vpc) *`identifier`*
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Determina l'indirizzo IP da utilizzare per connetterti alle EC2 istanze nel tuo VPC utilizzando Secure Shell (SSH). Per determinare il tuo indirizzo IP pubblico, in un'altra finestra o scheda

del browser, puoi utilizzare il servizio all'indirizzo. https://checkip.amazonaws.com Un esempio di indirizzo IP è `203.0.113.25/32`.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, trova l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per l'accesso SSH, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, autorizza solo un determinato indirizzo IP o un intervallo di indirizzi per accedere alle istanze utilizzando SSH.

b. Nella sezione Regole in entrata, scegliere Aggiungi regola.

c. Imposta i seguenti valori per la tua nuova regola in entrata per consentire l'accesso SSH alla tua istanza Amazon EC2 . In tal caso, puoi connetterti alla tua EC2 istanza Amazon per installare il server Web e altre utilità. Inoltre, ti connetti alla tua EC2 istanza per caricare contenuti per il tuo server web.

- Tipo: **`SSH`**
- Origine: indirizzo IP o intervallo ottenuto nella fase 1, ad esempio **`203.0.113.25/32`**.

d. Scegli Aggiungi regola.

e. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso HTTP al server Web:

- Tipo: **`HTTP`**
- Origine: **`0.0.0.0/0`**

5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

   Prendi nota dell'ID del gruppo di sicurezza perché sarà necessario in seguito in questo tutorial.

Creazione di un gruppo di sicurezza VPC per un cluster database privato

Per mantenere il cluster database privato, crea un secondo gruppo di sicurezza per l'accesso privato. Per connetterti ai cluster di DB private nel tuo VPC, aggiungi regole in entrata al tuo gruppo di sicurezza VPC che consentono il traffico solo dal tuo server web.

Per creare un gruppo di sicurezza VPC

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-db-securitygroup`**
   - Descrizione: **`Tutorial DB Instance Security Group`**
   - VPC: scegli il VPC che hai creato in precedenza, ad esempio: vpc- (tutorial-vpc) *`identifier`*
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   b. Imposta i seguenti valori per la tua nuova regola in entrata per consentire il traffico MySQL sulla porta 3306 dalla tua istanza Amazon. EC2 In tal caso, puoi connetterti dal server Web al cluster database ed eseguire l'archiviazione e il recupero dei dati dall'applicazione Web nel database.
      - Tipo: **`MySQL/Aurora`**
      - Source (Origine): l'identificatore del gruppo di sicurezza tutorial-securitygroup creato in precedenza in questo tutorial, ad esempio sg-9edd5cfb.
5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

Per creare un gruppo di sottoreti del database

Un gruppo di sottoreti DB è una raccolta di sottoreti creata in un VPC e che è possibile indicare per i cluster database. Un gruppo di sottoreti DB consente di specificare un determinato VPC quando si creano cluster database.

Creare un gruppo di sottoreti database

1. Identifica le sottoreti private per il database nel VPC.

   a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

   b. Scegli VPC Dashboard (Pannello di controllo VPC), quindi seleziona Subnets (Sottoreti).

   c. Nota la sottorete delle sottoreti denominata IDs 1-us-west-2a e 2-us-west-2b. tutorial-subnet-private tutorial-subnet-private

      La sottorete è necessaria quando create il gruppo di sottoreti DB. IDs

2. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   Assicurarsi di connettersi alla console Amazon RDS e non alla console Amazon VPC.

3. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).

4. Scegli Create DB Subnet Group (Crea gruppo di sottoreti del database).

5. Nella pagina Create DB subnet group (Crea gruppo di sottoreti del database) impostare questi valori in Subnet group details (Dettagli gruppi di sottoreti):

   - Nome: **`tutorial-db-subnet-group`**
   - Descrizione: **`Tutorial DB Subnet Group`**
   - VPC: tutorial-vpc (vpc-) *`identifier`*

6. Nella sezione Aggiungi sottoreti, scegliere Zone di disponibilità e Sottoreti.

   Per questo tutorial, scegli us-west-2a e us-west-2b per l'opzione Availability Zones (Zone di disponibilità). In Subnets (Sottoreti), scegli le sottoreti private identificate nella fase precedente.

7. Scegli Create (Crea).

   Il nuovo gruppo di sottoreti database viene visualizzato nell'elenco dei gruppi di sottoreti database sulla console RDS. Puoi scegliere il gruppo di sottoreti DB per visualizzare i dettagli nel riquadro dei dettagli nella parte inferiore della finestra. Questi dettagli includono tutte le sottoreti associate al gruppo.

> **Note**
>
> Se questo VPC è stato creato per il completamento di Tutorial: creazione di un server Web e un cluster database Amazon Aurora, creare il cluster di database seguendo le istruzioni riportate in Creazione di un cluster di database Amazon Aurora.

Eliminazione del VPC

Dopo aver creato il VPC e altre risorse per questo tutorial, è possibile eliminarle se non sono più necessarie.

> **Note**
>
> Se hai aggiunto risorse nel VPC creato per questo tutorial, potrebbe essere necessario eliminarle prima di poter eliminare il VPC. Per ulteriori informazioni, consulta Eliminazione del VPC nella Guida per l'utente di Amazon VPC.

Per eliminare un VPC e le risorse correlate

1. Eliminare il gruppo di sottoreti di database.
    a. Aprire la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
    b. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
    c. Seleziona il gruppo di sottoreti DB che desideri eliminare, ad esempio. tutorial-db-subnet-group
    d. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra di conferma.
2. Nota l'ID VPC.
    a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. Scegli VPC Dashboard, quindi scegli. VPCs
    c. Nell'elenco, identifica il VPC creato, ad esempio tutorial-vpc.
    d. Prendi nota del valore ID VPC del VPC creato. L'ID VPC è richiesto nei passaggi successivi.
3. Eliminare i gruppi di sicurezza.
    a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).

c. Seleziona il gruppo di sicurezza per l'istanza database di Amazon RDS, ad esempio. tutorial-db-securitygroup

d. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

e. Nella pagina Security Groups, seleziona il gruppo di sicurezza per l' EC2 istanza Amazon, ad esempio tutorial-securitygroup.

f. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

4. Elimina il VPC.

a. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

b. Scegli VPC Dashboard, quindi scegli. VPCs

c. Selezionare il VPC che si desidera eliminare, ad esempio tutorial-vpc.

d. In Operazioni, scegli Elimina VPC.

   Nella pagina di conferma vengono visualizzate altre risorse associate al VPC che verranno eliminate, incluse le subnet associate.

e. Nella pagina di conferma, immetti **`delete`** e scegliere Delete (Elimina).

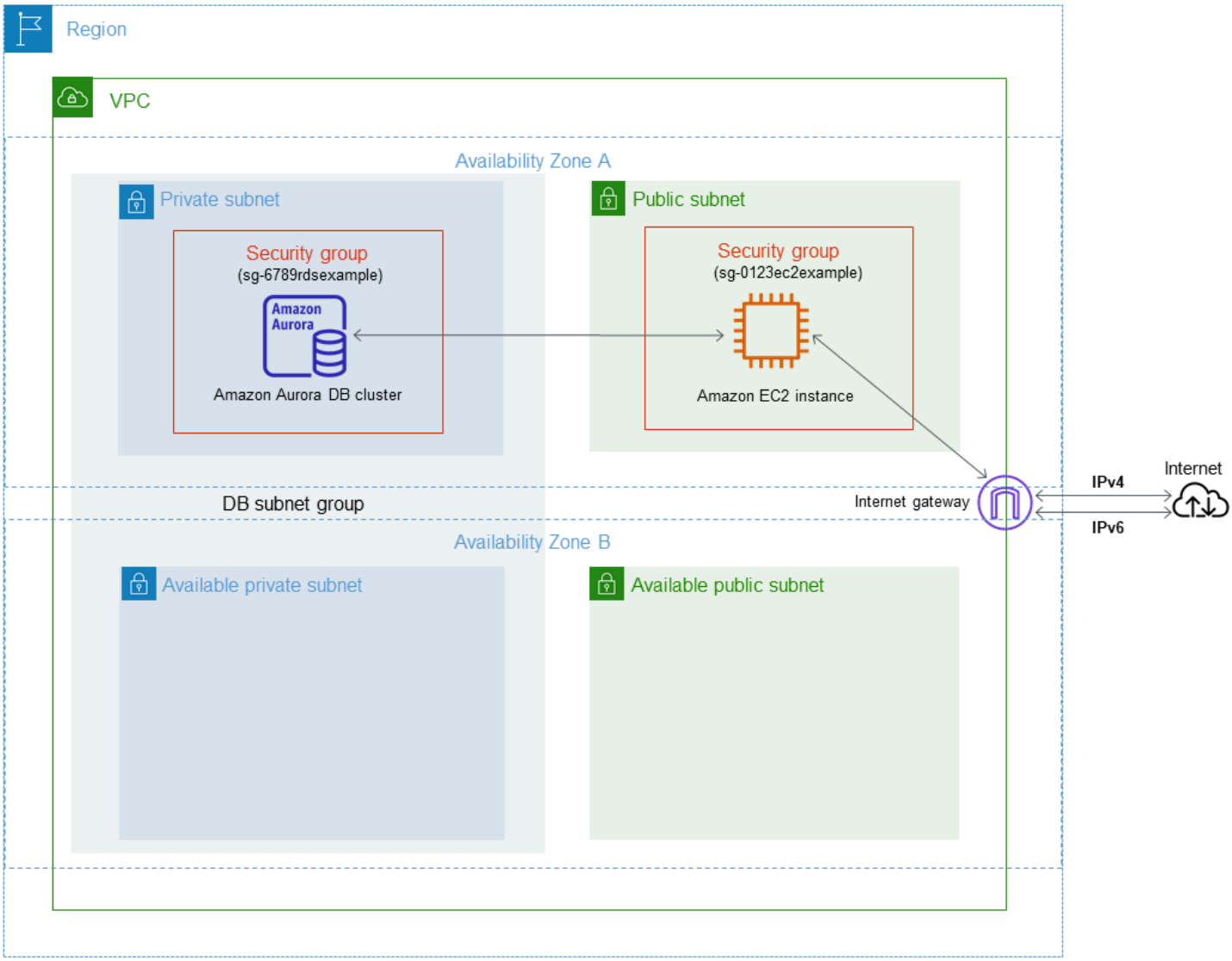
Tutorial: Creazione di un VPC per l'utilizzo con un cluster database (modalità dual-stack)

Uno scenario comune include un cluster database in un cloud privato virtuale (VPC) basato sul servizio Amazon VPC. Questo VPC condivide i dati con un'istanza Amazon EC2 pubblica in esecuzione nello stesso VPC.

In questo tutorial, si crea il VPC per questo scenario che funziona con un database in esecuzione in modalità dual-stack. Modalità dual-stack per abilitare la connessione tramite il protocollo di indirizzamento IPv6. Per ulteriori informazioni sull'indirizzamento IP, consulta Assegnazione di indirizzi IP in Amazon Aurora.

I cluster di rete dual-stack sono supportati nella maggior parte delle regioni. Per ulteriori informazioni, consulta Disponibilità di cluster database di rete dual-stack. Per esaminare le limitazioni della modalità dual-stack, consulta Limitazioni per cluster database di rete dual-stack.

Il seguente diagramma mostra questo scenario.



Per informazioni su altri scenari, consulta Scenari per l'accesso a un cluster di DB in un VPC.

Il cluster database deve essere disponibile solo per l'istanza Amazon EC2 e non per la rete Internet pubblica. Pertanto, occorre creare un VPC con sottoreti pubbliche e private. Il server Web è ospitato nella sottorete pubblica, in modo da poter raggiungere l'Internet pubblico. Il cluster database è ospitato in una sottorete privata. L'istanza Amazon EC2 può connettersi al cluster database perché è ospitata nello stesso VPC. Tuttavia, il cluster database non è disponibile per la rete Internet pubblica, fornendo una maggiore sicurezza.

Questo tutorial configura una sottorete pubblica e privata aggiuntiva in una zona di disponibilità separata. Queste sottoreti non vengono utilizzate dal tutorial. Un gruppo di sottoreti DB RDS richiede una sottorete in almeno due zone di disponibilità. La sottorete aggiuntiva semplifica la configurazione di più istanze database Aurora.

Per creare un cluster database che utilizza la modalità dual-stack, specifica Dual-stack mode (Modalità dual-stack) per l'opzione Network type (Tipo di rete). Puoi, inoltre, modificare un cluster database usando la stessa impostazione. Ulteriori informazioni sulla creazione di un cluster di database sono disponibili in Creazione di un cluster database Amazon Aurora. Per ulteriori informazioni sulla modifica di un cluster di database, consultare Modifica di un cluster database Amazon Aurora.

In questa esercitazione viene descritta la configurazione di un VPC per cluster di database Amazon Aurora. Per ulteriori informazioni su Amazon VPC, consulta la Guida per l'utente di Amazon VPC.

Creazione di un VPC con sottoreti pubbliche e private

Utilizza la procedura seguente per creare un VPC con sottoreti pubbliche e private.

Per creare un VPC e sottoreti

1. Apri alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nell'angolo in alto a destra di AWS Management Console, scegli la regione in cui creare il tuo VPC. Questo esempio utilizza la regione Stati Uniti orientali (Ohio).
3. Nell'angolo in alto a sinistra, scegli VPC Dashboard (Pannello di controllo VPC). Per iniziare a creare un VPC, scegli Create VPC (Crea VPC).
4. In Resources to create (Risorse da creare) nell'area VPC settings (Impostazioni VPC), scegli VPC and more (VPC e altro).
5. Nei campi VPC settings (Impostazioni VPC) restanti, imposta i seguenti valori:

   - Name tag auto-generation (Generazione automatica del tag del nome): **`tutorial-dual-stack`**
   - IPv4 CIDR block (Blocco CIDR IPv4): **`10.0.0.0/16`**
   - IPv6 CIDR block (Blocco CIDR IPv6): blocco CIDR IPv6 fornito da Amazon
   - Tenancy (Locazione): Default
   - Number of Availability Zones (AZs) (Numero di zone di disponibilità): 2
   - Customize AZs (Personalizza le zone di disponibilità): mantieni i valori predefiniti.
   - Number of public subnet (Numero di sottoreti pubbliche): 2
   - Number of private subnets (Numero di sottoreti private): 2
   - Customize subnets CIDR blocks (Personalizza blocchi CIDR delle sottoreti): mantieni i valori predefiniti.

- NAT gateways ($) (Gateway NAT): nessuna
- Egress only internet gateway (Gateway Internet solo in uscita): No
- VPC endpoints (Endpoint VPC): nessuno
- DNS options (Opzioni DNS): mantieni i valori predefiniti.

> **Note**
>
> Amazon RDS richiede almeno due sottoreti in due zone di disponibilità diverse per supportare le implementazioni Multi-AZ di un'istanza DB. In questo tutorial viene creata crea un'implementazione Single-AZ, ma il requisito semplifica la conversione in un'implementazione Multi-AZ di un'istanza DB in futuro.

6. Seleziona Crea VPC.

Per creare un gruppo di sicurezza VPC per un'istanza Amazon EC2 pubblica

È possibile a questo punto aggiungere un gruppo di sicurezza per l'accesso pubblico. Per connettersi alle istanze EC2 nel VPC, aggiungi regole in entrata al gruppo di sicurezza VPC per consentire il traffico da Internet.

Per creare un gruppo di sicurezza VPC

1. Aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-dual-stack-securitygroup`**
   - Descrizione: **`Tutorial Dual-Stack Security Group`**
   - VPC: scegli il VPC creato in precedenza, ad esempio vpc-***`identifier`*** (tutorial-dual-stack-vpc)
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Determina l'indirizzo IP da utilizzare per connettersi alle istanze EC2 nel VPC utilizzando Secure Shell (SSH).

Un esempio di indirizzo Internet Protocol versione 4 (IPv4) è `203.0.113.25/32`.
Un esempio di intervalli di indirizzi Internet Protocol versione 6 (IPv6) è `2001:db8:1234:1a00::/64`.

In molti casi, è possibile eseguire la connessione tramite un fornitore di servizi Internet (ISP) o con la protezione di un firewall senza un indirizzo IP statico. In tal caso, trova l'intervallo di indirizzi IP utilizzati dai computer client.

> **Warning**
>
> Se utilizzi `0.0.0.0/0` per IPv4 o `::0` per IPv6, consenti a tutti gli indirizzi IP di accedere alle istanze pubbliche utilizzando SSH. Questo approccio è accettabile per un breve periodo di tempo in un ambiente di test, ma non è sicuro per gli ambienti di produzione. In produzione, è preferibile autorizzare l'accesso alle istanze solo a indirizzo IP o a un intervallo di indirizzi specifico.

b. Nella sezione Regole in entrata, scegliere Aggiungi regola.

c. Imposta i seguenti valori per la nuova regola in entrata per consentire l'accesso Secure Shell (SSH) all'istanza Amazon EC2. In questo caso, è possibile connettersi all'istanza EC2 per installare client SQL e altre applicazioni. Specifica un indirizzo IP per poter accedere all'istanza EC2:

- Tipo: **`SSH`**
- Origine: indirizzo IP o intervallo ottenuto nel passaggio a. Un esempio di indirizzo IP IPv4 è **`203.0.113.25/32`**. Un esempio di indirizzo IP IPv6 è **`2001:DB8::/32`**.

5. Per creare il gruppo di sicurezza, scegli Create security group (Crea gruppo di sicurezza).

   Prendi nota dell'ID del gruppo di sicurezza perché sarà necessario in seguito in questo tutorial.

Creazione di un gruppo di sicurezza VPC per un cluster database privato

Per mantenere il cluster database privato, crea un secondo gruppo di sicurezza per l'accesso privato. Per connetterti ai cluster database privati nel VPC, aggiungi le regole in entrata al gruppo di sicurezza VPC. Queste consentono il traffico solo dall'istanza Amazon EC2.

Per creare un gruppo di sicurezza VPC

1. Aprire la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Scegliere VPC Dashboard (Pannello di controllo VPC), Security Groups (Gruppi di sicurezza) e quindi Create security group (Crea gruppo di sicurezza).
3. Nella pagina Create security group (Crea gruppo di sicurezza) impostare questi valori:
   - Security group name: (Nome del gruppo di sicurezza: **`tutorial-dual-stack-db-securitygroup`**
   - Descrizione: **`Tutorial Dual-Stack DB Instance Security Group`**
   - VPC: scegli il VPC creato in precedenza, ad esempio vpc-***`identifier`*** (tutorial-dual-stack-vpc)
4. Aggiungere regole in entrata al gruppo di sicurezza
   a. Nella sezione Regole in entrata, scegliere Aggiungi regola.
   b. Impostare i valori seguenti per la nuova regola in entrata per consentire il traffico MySQL sulla porta 3306 dall'istanza Amazon EC2. In tal modo, puoi eseguire la connessione dall'istanza EC2 al cluster database ed Questa operazione consente di inviare dati dall'istanza EC2 al database.
      - Tipo: MySQL/Aurora
      - Source (Origine): l'identificatore del gruppo di sicurezza tutorial-dual-stack-securitygroup creato in precedenza in questo tutorial, ad esempio sg-9edd5cfb.
5. Per creare il gruppo di sicurezza, scegliere Crea gruppo di sicurezza.

Per creare un gruppo di sottoreti del database

Un gruppo di sottoreti DB è una raccolta di sottoreti creata in un VPC e che è possibile indicare per i cluster database. L'utilizzo di un gruppo di sottoreti DB consente di specificare un determinato VPC durante la creazione di cluster database. Per creare un gruppo di sottoreti database compatibile con `DUAL`, tutte le sottoreti devono essere compatibili con `DUAL`. Per essere compatibile con `DUAL`, a una sottorete deve essere associato un CIDR IPv6.

Creare un gruppo di sottoreti database

1. Identifica le sottoreti private per il database nel VPC.

a. Apri alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
b. Scegli VPC Dashboard (Pannello di controllo VPC), quindi seleziona Subnets (Sottoreti).
c. Prendi nota degli ID sottorete delle sottoreti denominati tutorial-dual-stack-subnet-private1-us-west-2a e tutorial-dual-stack-subnet-private2-us-west-2b.

   Gli ID sottorete saranno necessari quando si crea il gruppo di sottoreti DB.

2. Apri la console di Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.

   Assicurarsi di connettersi alla console Amazon RDS e non alla console Amazon VPC.
3. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
4. Scegli Create DB Subnet Group (Crea gruppo di sottoreti del database).
5. Nella pagina Create DB subnet group (Crea gruppo di sottoreti del database) impostare questi valori in Subnet group details (Dettagli gruppi di sottoreti):

   - Nome: **`tutorial-dual-stack-db-subnet-group`**
   - Descrizione: **`Tutorial Dual-Stack DB Subnet Group`**
   - VPC: tutorial-dual-stack-vpc (identificatore vpc)
6. Nella sezione Add subnets (Aggiungi sottoreti), scegli i valori desiderati per le opzioni Availability Zones (Zone di disponibilità) e Subnets (Sottoreti).

   Per questo tutorial, scegli us-east-2b e us-east-2c per l'opzione Availability Zones (Zone di disponibilità). In Subnets (Sottoreti), scegli le sottoreti private identificate nella fase precedente.
7. Scegliere Create (Crea).

Il nuovo gruppo di sottoreti database viene visualizzato nell'elenco dei gruppi di sottoreti database sulla console RDS. È possibile scegliere il gruppo di sottoreti database per visualizzarne i dettagli. I dettagli includono i protocolli di indirizzamento supportati e tutte le sottoreti associate al gruppo e il tipo di rete supportato dal gruppo di sottoreti database.

Creare un'istanza Amazon EC2 in modalità dual-stack

Per creare un'istanza Amazon EC2, segui le istruzioni in Launch an instance using the new launch instance wizard nella Amazon EC2 User Guide.

Nella pagina Configure Instance Details (Configura i dettagli dell'istanza), imposta questi valori e lascia gli altri valori come predefiniti:

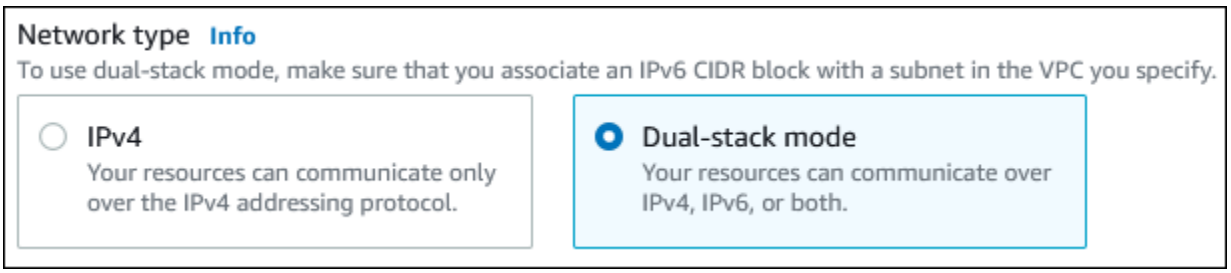
- Rete: scegli un VPC esistente con sottoreti sia pubbliche che private, ad esempio tutorial-dual-stack-vpc (vpc-*identifier*) creato in Creazione di un VPC con sottoreti pubbliche e private.
- Subnet: scegli una sottorete pubblica esistente, ad esempio subnet- ***identifier*** | -public1-us-east-2a | us-east-2a creata in. tutorial-dual-stack-subnet Per creare un gruppo di sicurezza VPC per un'istanza Amazon EC2 pubblica
- Auto-assign Public IP (Assegna automaticamente IP pubblico): scegli Enable (Abilita).
- Auto-assign IPv6 IP (Assegna automaticamente IP IPv6): scegli Enable (Abilita).
- Firewall (security groups) (Firewall (gruppi di sicurezza)): scegli Select an existing security group (Seleziona un gruppo di sicurezza esistente).
- Gruppi di sicurezza comuni: scegli un gruppo di sicurezza esistente, ad esempio il `tutorial-securitygroup` creato in Per creare un gruppo di sicurezza VPC per un'istanza Amazon EC2 pubblica. Assicurarsi che il gruppo di sicurezza scelto includa regole in ingresso per Secure Shell (SSH) e l'accesso HTTP.

Creazione di un cluster database in modalità dual-stack

In questo passaggio, viene creata un cluster database che viene eseguito in modalità dual-stack.

Per creare un'istanza database.

1. Accedi AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Questo esempio utilizza la regione Stati Uniti orientali (Ohio).
3. Nel riquadro di navigazione, scegliere Databases (Database).
4. Scegliere Create database (Crea database).
5. Nella pagina Create database (Crea database), assicurati che l'opzione Standard create (Creazione standard) sia selezionata, quindi scegli il tipo di motore DB Aurora MySQL.
6. Nella sezione Connettività, imposta i seguenti valori:
    - Network type (Tipo di rete): scegli Dual-stack mode (Modalità dual-stack).



- Virtual private cloud (VPC) Cloud privato virtuale (VPC): scegli un VPC esistente con sottoreti sia pubbliche che private, ad esempio tutorial-dual-stack-vpc (vpc-*identifier*) creato in Creazione di un VPC con sottoreti pubbliche e private.

  Il VPC deve avere sottoreti in diverse zone di disponibilità.
- DB subnet group (Gruppo di sottoreti DB): scegli un gruppo di sottoreti DB per il VPC, ad esempio tutorial-dual-stack-db-subnet-group creato in Per creare un gruppo di sottoreti del database.
- Public access (Accesso pubblico): scegli No.
- VPC security group (firewall) (Gruppo di sicurezza VPC (firewall)): seleziona Choose existing (Sceglie esistente).
- Gruppi di sicurezza VPC esistenti: scegli un gruppo di sicurezza VPC esistente che sia configurato per accesso privato, ad esempio tutorial-dual-stack-db-securitygroup creato in Creazione di un gruppo di sicurezza VPC per un cluster database privato.

  Rimuovere gli altri gruppi di sicurezza, come quello predefinito, selezionando la X associata a esso.
- Availability Zone (Zona di disponibilità): scegli us-west-2a.

  Per evitare il traffico tra AZ, assicurati che l'istanza database e l'istanza EC2 si trovino nella stessa zona di disponibilità.

7. Per le restanti sezioni, specifica le impostazioni del cluster di database. Per informazioni su ciascuna impostazione, consulta Impostazioni per cluster di database Aurora.

Connessione all'istanza Amazon EC2 e al cluster database

Dopo aver creato l'istanza Amazon EC2 e il cluster database in modalità dual-stack, puoi eseguire la connessione utilizzando il protocollo IPv6. Per connetterti a un'istanza Amazon EC2 utilizzando il protocollo IPv6, segui le istruzioni in Connetti alla tua istanza Linux nella Amazon EC2 User Guide.

Per connetterti al cluster database Aurora MySQL dall'istanza Amazon EC2, segui le istruzioni in Connessione a un cluster di database Aurora MySQL.

Eliminazione del VPC

Dopo aver creato il VPC e altre risorse per questo tutorial, è possibile eliminarle se non sono più necessarie.

Se hai aggiunto risorse nel VPC creato per questo tutorial, potrebbe essere necessario eliminarle prima di poter eliminare il VPC. Esempi di risorse sono le istanze Amazon EC2 o i cluster database. Per ulteriori informazioni, consulta Eliminazione del VPC nella Guida per l'utente di Amazon VPC.

Per eliminare un VPC e le risorse correlate

1. Eliminare il gruppo di sottoreti database:
    a. Apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
    b. Nel pannello di navigazione selezionare Subnet groups (Gruppi di sottoreti).
    c. Seleziona il gruppo di sottoreti database da eliminare, ad esempio tutorial-db-subnet-group.
    d. Scegliere Delete (Elimina) e quindi scegliere Delete (Elimina) nella finestra di conferma.
2. Annotare l'ID VPC:
    a. Apri alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. ScegliereVPC Dashboard (Pannello di controllo VPC) e quindi scegliere VPC.
    c. Nell'elenco, individua il VPC creato, ad esempio tutorial-dual-stack-vpc.
    d. Annotare il valore ID VPC del VPC creato. Sarà necessario usare questo ID VPC nei passaggi successivi.
3. Eliminare i gruppi di sicurezza:
    a. Apri alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
    b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere Security Groups (Gruppi di sicurezza).
    c. Selezionare il gruppo di sicurezza per l'istanza database di Amazon RDS, ad esempio tutorial-dual-stack-db-securitygroup.
    d. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

e. Nella pagina Security Groups (Gruppi di sicurezza), seleziona il gruppo di sicurezza per l'istanza Amazon EC2, ad esempio tutorial-dual-stack-securitygroup..

f. In Actions (Operazioni), scegli Delete security groups (Elimina gruppi di sicurezza) e quindi scegli Delete (Elimina) nella pagina di conferma.

4. Elliminare il gateway NAT:

a. Apri alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

b. Scegliere VPC Dashboard (Pannello di controllo VPC) e quindi scegliere NAT Gateways (Gateway NAT).

c. Seleziona il gateway NAT del VPC creato. Utilizzare l'ID VPC per identificare il gateway NAT corretto.

d. In Actions (Operazioni), scegli Delete NAT gateway (Elimina gateway NAT).

e. Nella pagina di conferma, immetti **`delete`** e scegliere Delete (Elimina).

5. Eliminare il VPC

a. Apri alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.

b. ScegliereVPC Dashboard (Pannello di controllo VPC) e quindi scegliere VPC.

c. Seleziona il VPC da eliminare, ad esempio tutorial-dual-stack-vpc.

d. In Operazioni, scegli Elimina VPC.

Nella pagina di conferma vengono visualizzate altre risorse associate al VPC che verranno eliminate, incluse le subnet associate.

e. Nella pagina di conferma, immetti **`delete`** e scegliere Delete (Elimina).

6. Rilasciare gli indirizzi IP elastici:

a. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.

b. Scegliere EC2 Dashboard (Pannello di controllo EC2) e quindi scegliere Elastic IPs (IP elastici).

c. Seleziona l'indirizzo IP elastico da rilasciare.

d. In Actions (Operazioni), scegli Release Elastic IP addresses (Rilascia indirizzi IP elastici).

e. Nella pagina di conferma scegli Release (Rilasciare).

Quote e vincoli per Amazon Aurora

Di seguito, è possibile trovare una descrizione delle quote di risorse e dei vincoli di denominazione per Amazon Aurora.

Argomenti

- Quote in Amazon Aurora
- Vincoli per la denominazione in Amazon Aurora
- Limiti di dimensione Amazon Aurora

Quote in Amazon Aurora

Ogni AWS account ha quote, per ogni AWS regione, sul numero di risorse Aurora che è possibile creare. Una volta raggiunta la quota della risorsa, le ulteriori richieste di creazione di tale risorsa restituiranno un errore con un'eccezione.

La tabella seguente elenca le risorse e le relative quote per regione. AWS

Nome	Predefinita	Adatta e	Descrizione
Autorizzazioni per gruppo di sicurezza DB	Ogni regione supportata: 20	No	Numero di autorizzazioni per gruppo di sicurezza DB
Endpoint personalizzati per cluster DB	Ogni regione supportata: 5	Sì	Il numero massimo di endpoint personalizzati che è possibile creare per cluster Aurora DB in questo account nella regione corrente. Questo valore riflette il maggior numero di endpoint personalizzati in un cluster DB dell'account. Altri cluster DB nell'acco

Nome	Predefinita	Adatta e	Descrizione
			unt potrebbero avere un numero inferiore di endpoint personalizzati.
Versioni personalizzate del motore	Ogni regione supportata: 40	Sì	Il numero massimo di versioni personalizzate del motore consentite in questo account nella regione corrente
Gruppi di parametri di cluster database	Ogni Regione supportata: 50	Sì	Il numero massimo di gruppi di parametri del cluster di database
Cluster database	Ogni regione supportata: 40	Sì	Il numero massimo di cluster Aurora consentit i in questo account nella regione corrente
Istanze DB	ap-south-1:20 Ciascuna delle altre regioni supportate: 40	Sì	Il numero massimo di istanze database consentite in questo account nella regione corrente
Gruppi di shard DB	Ogni regione supportata: 5	Sì	Il numero massimo di gruppi di shard DB per Aurora Limitless Database in questo account nella regione corrente
Gruppi di sottoreti database	ap-south-1:20 Ogni altra regione supportata: 50	Sì	Il numero massimo di gruppi di sottoreti di database

Nome	Predefinita	Adatta e	Descrizione
Dimensione del corpo della richiesta HTTP dell'API dati	Ogni regione supportata: 4 MB	No	La dimensione massima consentita per il corpo della richiesta HTTP.
Numero massimo di coppie segrete del cluster simultanee dell'API dati	Ogni regione supportata: 30	No	Il numero massimo di coppie uniche di cluster e segreti DB Aurora Serverless v1 nelle richieste simultanee di Data API per questo account nella regione corrente. AWS
Numero massimo di richieste simultane e dell'API dati	Ogni regione supportata: 500	No	Il numero massimo di richieste Data API a un cluster DB Aurora Serverless v1 che utilizzano lo stesso segreto e possono essere elaborate contempor aneamente. Le richieste aggiuntive vengono messe in coda ed elaborate al termine delle richieste in corso.
Dimensione massima del set di risultati dell'API dati	Ogni regione supportata: 1 MB	No	La dimensione massima del set di risultati del database che può essere restituito dall'API dati.

Nome	Predefinita	Adatta e	Descrizione
Dimensione massima dell'API dati della stringa di risposta JSON	Ogni regione supportata: 10 megabyte	No	La dimensione massima della stringa di risposta JSON semplificata restituita dall'API dati RDS.
Richieste API dati al secondo	Ogni regione supportata: 1.000 al secondo	No	Il numero massimo di richieste all'API Data al secondo consentito per questo account nella regione corrente. AWS Questa quota si applica solo ai cluster Amazon Aurora Serverless v1.
Abbonamenti a eventi	Ogni regione supportata: 20	Sì	Il numero massimo di sottoscrizioni di eventi
Ruoli IAM per cluster di database	Ogni regione supportata: 5	Sì	Il numero massimo di ruoli IAM associati a un cluster di database
Ruoli IAM per istanza database	Ogni regione supportata: 5	Sì	Il numero massimo di ruoli IAM associati a un'istanza database
Integrazioni	Ogni regione supportata: 100	No	Il numero massimo di integrazioni consentite in questo account nella regione corrente AWS
Snapshot del cluster di database manuale	Ogni regione supportata: 100	Sì	Il numero massimo di snapshot manuali del cluster di database

Nome	Predefinita	Adattabile	Descrizione
Snapshot manuali dell'istanza database	Ogni regione supportata: 100	Sì	Il numero massimo di snapshot manuali di istanza database
Gruppi di opzioni	Ogni regione supportata: 20	Sì	Il numero massimo di gruppi di opzioni
Gruppi di parametri	ap-south-1:20 Ogni altra regione supportata: 50	Sì	Il numero massimo di gruppi di parametri
Proxy	Ogni regione supportata: 20	Sì	Il numero massimo di proxy consentito in questo account nella regione corrente AWS
Lettura delle repliche per primario	Ogni regione supportata: 15	Sì	Il numero massimo di repliche di lettura per istanza database primario. Questa quota non può essere modificata per Amazon Aurora.
Istanze database riservate	Ogni regione supportata: 40	Sì	Il numero massimo di istanze DB riservate consentite in questo account nella regione corrente AWS
Gruppi di sicurezza	ap-south-1:20 Ogni altra regione supportata: 25	Sì	Il numero massimo di gruppi di sicurezza DB

Nome	Predefinita	Adatta e	Descrizione
Sottoreti per gruppo di sottoreti del database	Ogni regione supportata: 20	No	Il numero massimo di sottoreti per gruppo di sottoreti di database
Totale storage per tutte le istanze database	Ogni regione supportata: 100.000 GB	Sì	Lo spazio di archiviazione massimo totale (in GB) per tutte le istanze database di Amazon RDS aggiunte insieme. Questa quota non si applica ad Amazon Aurora, che ha un volume cluster massimo di 128 TiB per ogni cluster DB.

Note

Per impostazione predefinita, puoi avere fino a 40 istanze database. Le istanze database RDS, le istanze database Aurora, le istanze Amazon Neptune e le istanze Amazon DocumentDB si applicano a questa quota.
Se l'applicazione richiede più istanze database, puoi richiedere altre istanze database aprendo la console Service Quotas. Nel pannello di navigazione, scegliere servizi AWS . Scegli Amazon Relational Database Service (Amazon RDS), scegli una quota e segui le istruzioni per richiedere un aumento della quota. Per ulteriori informazioni, consulta Richiesta di un aumento di quota nella Guida per l'utente per Service Quotas.
Per RDS for Oracle, è possibile creare fino a 15 repliche di lettura per istanza DB di origine in ogni regione, ma si consiglia di limitare le repliche a 5 per ridurre al minimo il ritardo di replica.
I backup gestiti da AWS Backup sono considerati istantanee manuali del cluster DB, ma non vengono conteggiati ai fini della quota di snapshot di cluster manuali. Per informazioni in merito AWS Backup, consulta la Guida per gli sviluppatori AWS Backup .

Se si utilizza una delle operazioni dell'API RDS e si supera la quota predefinita del numero di chiamate al secondo, l'API Amazon RDS emette un errore simile al seguente.

ClientError: Si è verificato un errore (ThrottlingException) durante la chiamata dell'*API_name*operazione: Frequenza superata.

Qui, riduci il numero di chiamate al secondo. La quota è destinata a coprire la maggior parte dei casi d'uso. Se sono necessarie quote più elevate, puoi richiedere un aumento della quota utilizzando una delle seguenti opzioni:

- Dalla console, apri la console Service Quotas.
- Da AWS CLI, usa il request-service-quota-increase AWS CLI comando.

Per maggiori informazioni, consulta Guida per l'utente di Service Quotas.

Vincoli per la denominazione in Amazon Aurora

I vincoli di denominazione in Amazon RDS Aurora sono i seguenti:

-
  - Devono contenere da 1 a 63 caratteri alfanumerici o trattini.
  - Il primo carattere deve essere una lettera.
  - Non può terminare con un trattino o contenere due trattini consecutivi.
  - Deve essere univoco per tutte le istanze DB per AWS account, per regione. AWS
- Nome iniziale del database: i vincoli relativi al nome del database differiscono tra Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consulta le impostazioni disponibili durante la creazione di ogni cluster DB.
- Nome utente principale: i vincoli del nome utente principale differiscono per ogni motore di database. Per ulteriori informazioni, consulta le impostazioni disponibili durante la creazione del cluster di DB.
- Password principale:
  - La password dell'utente master del database può includere qualsiasi carattere ASCII stampabile, tranne `/`, `'`, `"`, `@` o uno spazio.
  - La password può contenere il seguente numero di caratteri ASCII stampabili a seconda del motore DB:
    - Aurora MySQL: 8 - 41

    - Aurora PostgreSQL: 8 - 99
- Gruppo di parametri DB:
    - Devono contenere da 1 a 255 caratteri alfanumerici.
    - Il primo carattere deve essere una lettera.
    - I trattini sono consentiti, ma il nome non può terminare con un trattino o contenere due trattini consecutivi.
- Gruppo di sottoreti DB:
    - Devono contenere da 1 a 255 caratteri.
    - Sono consentiti caratteri alfanumerici, spazi, trattini, trattini bassi e punti.

Limiti di dimensione Amazon Aurora

Limiti delle dimensioni dello storage

Un volume di cluster Aurora può raggiungere una dimensione massima di 128 tebibytes (TiB) per le seguenti versioni del motore:

- Tutte le versioni di Aurora MySQL versione 3 disponibili, Aurora MySQL versione 2, la versione 2.09 e successive
- Tutte le versioni di Aurora PostgreSQL disponibili

Per le versioni precedenti del motore, la dimensione massima di un volume del cluster Aurora è 64 TiB. Per ulteriori informazioni, consulta Ridimensionamento automatico dello storage Aurora.

Per monitorare lo spazio di archiviazione rimanente, è possibile utilizzare il parametro `AuroraVolumeBytesLeftTotal`. Per ulteriori informazioni, consulta Parametri a livello di cluster per Amazon Aurora.

Limiti delle dimensioni delle tabelle SQL

Per un cluster di database Aurora MySQL, la dimensione massima della tabella è 64 tebibyte (TiB). Per un cluster database Aurora PostgreSQL, la dimensione massima della tabella è 32 tebibyte (TiB). Consigliamo di seguire le best practice per la progettazione delle tabelle, ad esempio partizionando le tabelle di grandi dimensioni.

Limiti di ID spazio delle tabelle

L'ID spazio delle tabelle massimo per Aurora MySQL è 2147483647. Se crei ed elimini spesso tabelle, assicurati di prestare attenzione allo spazio a disposizione tra le tabelle IDs e pianifica

di utilizzare i dump logici. Per ulteriori informazioni, consulta Migrazione logica da My SQL ad Amazon Aurora My SQL utilizzando mysqldump.

Risoluzione dei problemi per Amazon Aurora

Utilizza le sezioni seguenti per risolvere i problemi che riscontri con le istanze database in Amazon RDS e Amazon Aurora.

Argomenti

- Impossibile connettersi all'istanza database di Amazon RDS
- Problemi relativi alla sicurezza di Amazon RDS
- Reimpostazione della password del ruolo di proprietario dell'istanza di database
- Errore o riavvio di un'istanza di database Amazon RDS
- Modifiche ai parametri di database Amazon RDS che non hanno effetto
- Problemi di memoria liberabile in Amazon Aurora
- Problemi di replica relativi a Amazon Aurora MySQL

Per informazioni sui problemi di debug con l'API Amazon RDS, consulta Risoluzione dei problemi delle applicazioni in Aurora.

Impossibile connettersi all'istanza database di Amazon RDS

Quando non è possibile connettersi a un'istanza database, le cause comuni sono le seguenti:

- Regole in entrata: le regole di accesso applicate dal firewall locale e gli indirizzi IP autorizzati per accedere all'istanza database potrebbero non corrispondere. Il problema è probabilmente correlato alle regole in entrata del gruppo di sicurezza.

  Per impostazione predefinita, le istanze database non consentono l'accesso. L'accesso viene concesso tramite un gruppo di sicurezza associato al VPC che consente il traffico in entrata e in uscita dall'istanza database. Se necessario, aggiungi regole in entrata e in uscita per la situazione particolare al gruppo di sicurezza. Puoi specificare un indirizzo IP, un intervallo di indirizzi IP o un altro gruppo di sicurezza VPC.

> **Note**
>
> Quando si aggiunge una nuova regola in entrata, puoi scegliere My IP (Il mio IP) per Source (Origine) per consentire l'accesso all'istanza database dall'indirizzo IP rilevato nel browser.

Per ulteriori informazioni sulla configurazione di un gruppo di sicurezza, consulta Fornitura dell'accesso al cluster di database nel VPC creando un gruppo di sicurezza.

> **Note**
>
> Le connessioni client da indirizzi IP all'interno dell'intervallo 169.254.0.0/16 non sono consentite. Si tratta di un intervallo APIPA (Automatic Private IP Addressing), utilizzato per gli indirizzi con collegamenti locali.

- Public accessibility (Accesso pubblico): per connettersi all'istanza database dall'esterno del VPC, ad esempio utilizzando un'applicazione client, occorre assegnare all'istanza un indirizzo IP pubblico.

  Per rendere l'istanza accessibile pubblicamente, modificarla e scegliere Yes (Sì) in Public accessibility (Accesso pubblico). Per ulteriori informazioni, consulta Nascondere cluster database in un VPC da Internet.
- Porta: la porta specificata al momento della creazione dell'istanza DB non può essere utilizzata per inviare o ricevere comunicazioni a causa delle restrizioni del firewall locale. Per determinare se la rete consente l'utilizzo della porta specificata per le comunicazioni in entrata e in uscita, consulta il tuo amministratore di rete.
- Disponibilità: per una nuova istanza database creata, lo stato dell'istanza database è `creating` finché non è pronta per essere utilizzata. Quando lo stato cambia in `available`, puoi connetterti all'istanza database. A seconda delle dimensioni dell'istanza di database, potresti dover attendere circa 20 minuti prima che questa diventi disponibile.
- Gateway Internet – Per un'istanza database che deve essere accessibile pubblicamente, le sottoreti nel gruppo di sottoreti del database devono disporre di un gateway Internet.

Per configurare un gateway Internet per una sottorete

1. Accedi a AWS Management Console e apri la console Amazon RDS all'indirizzo https://console.aws.amazon.com/rds/.
2. Nel riquadro di navigazione, scegliere Databases (Database) e selezionare il nome dell'istanza database.
3. Nella scheda Connectivity & security (Connettività e sicurezza) annotare i valori dell'ID VPC in VPC e l'ID della sottorete in Subnets (Sottoreti).
4. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
5. Nel riquadro di navigazione, scegliere Internet Gateways. Verificare che al VPC sia associato un Internet gateway. In caso contrario, scegliere Create Internet Gateway (Crea Internet gateway) per crearne uno. Selezionare l'Internet gateway, quindi Attach to VPC (Associa a VPC) e seguire le istruzioni di associazione del gateway al VPC.
6. Nel riquadro di navigazione scegliere Subnets (Sottoreti) e selezionare la sottorete desiderata.
7. Nella scheda Route Table (Tabella di routing) verificare che sia presente un instradamento con `0.0.0.0/0` come destinazione e l'Internet gateway del VPC come target.

   Se ti connetti alla tua istanza utilizzando il relativo IPv6 indirizzo, verifica che esista un percorso per tutto IPv6 il traffico (`::/0`) che punti al gateway Internet. In caso contrario, eseguire le seguenti operazioni:

   a. Selezionare l'ID per la tabella di routing (rtb-xxxxxxxx) per navigare alla tabella di routing.
   b. Nella scheda Routes (Route), scegliere Edit routes (Modifica route). Selezionare Add route (Aggiungi route), utilizzare `0.0.0.0/0` come destinazione e il gateway internet come target.

      Ad esempio IPv6, scegli Aggiungi percorso, utilizza `::/0` come destinazione e il gateway Internet come destinazione.
   c. Seleziona Salva route.

   Inoltre, se stai tentando di connetterti all' IPv6 endpoint, assicurati che l'intervallo di IPv6 indirizzi del client sia autorizzato a connettersi all'istanza DB.

Per ulteriori informazioni, consulta Uso di un cluster database in un VPC.

Test della connessione a un'istanza database

Puoi verificare la connessione a un'istanza database utilizzando strumenti comuni di Linux o Windows.

Da un terminale Linux o Unix, puoi eseguire il test della connessione immettendo quanto segue. Sostituisci *DB-instance-endpoint* con l'endpoint e *port* con la porta dell'istanza database.

```
nc -zv DB-instance-endpoint port
```

Di seguito è riportato un comando di esempio e il valore restituito.

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Gli utenti Windows possono utilizzare Telnet per eseguire il test della connessione a un'istanza di database. Le operazioni di Telnet non sono supportate, ad eccezione del test della connessione. Se una connessione ha esito positivo, l'operazione non restituisce alcun messaggio. Se una connessione non va a buon fine, riceverai un messaggio di errore del tipo seguente.

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Se le operazioni di Telnet hanno esito positivo, il tuo gruppo di sicurezza è configurato correttamente.

> **Note**
>
> Amazon RDS non accetta il traffico del protocollo ICMP (Internet Control Message Protocol), incluso il ping.

Risoluzione di problemi di autenticazione della connessione

In alcuni casi, è possibile connettersi all'istanza database, ma si ricevono errori di autenticazione. In questi casi, potrebbe essere necessario ripristinare la password utente master per l'istanza database. A tale scopo, è necessario modificare l'istanza RDS.

Problemi relativi alla sicurezza di Amazon RDS

Per evitare problemi di sicurezza, non utilizzare mai l'indirizzo e-mail e la password dell'utente Account AWS root per un account utente. È consigliabile utilizzare l'utente root per creare utenti e assegnarli agli account utente DB. È inoltre possibile utilizzare l'utente root per creare altri account utente, se necessario.

Per informazioni sulla creazione di utenti, consulta Creazione di un utente IAM nell' Account AWS. Per informazioni sulla creazione di utenti in AWS IAM Identity Center, consulta Gestire le identità in IAM Identity Center.

Messaggio di errore "Impossibile recuperare gli attributi dell'account, alcune funzioni della console potrebbero non essere attive."

Puoi ricevere questo errore per diversi motivi. È possibile che l'account non disponga delle autorizzazioni o che l'account non sia stato configurato correttamente. Se si tratta di un nuovo account, potrebbe essere necessario attendere che l'account sia pronto. Se si tratta di un account esistente, nelle policy di accesso potrebbero non essere disponibili alcune autorizzazioni per eseguire determinate operazioni, come creare un'istanza database. Per risolvere il problema, l'amministratore deve fornire i ruoli necessari per il tuo account. Per ulteriori informazioni, consulta la documentazione di IAM.

Reimpostazione della password del ruolo di proprietario dell'istanza di database

Se si viene bloccati dal cluster di DB, è possibile accedere come utente master. È quindi possibile reimpostare le credenziali per altri utenti o ruoli amministrativi. Se non riesci ad accedere come utente principale, il proprietario dell' AWS account può reimpostare la password dell'utente principale. Per informazioni dettagliate sugli account o sui ruoli amministrativi che potrebbe essere necessario reimpostare, vedere Privilegi dell'account utente master.

Puoi modificare la password dell'istanza DB utilizzando la console Amazon RDS, il AWS CLI comando modify-db-instanceo utilizzando l'operazione Modify DBInstance API. Per ulteriori informazioni sulla modifica di un'istanza database in un cluster di database, consulta Modifica di un'istanza database in un cluster database.

Errore o riavvio di un'istanza di database Amazon RDS

Un errore dell'istanza database può verificarsi quando viene riavviata. Può anche verificarsi quando l'istanza database viene inserita in uno stato che impedisce l'accesso e quando il database viene riavviato. Un riavvio può verificarsi quando si riavvia manualmente l'istanza database. Un riavvio può anche verificarsi quando si modifica un'impostazione dell'istanza database che richiede un riavvio prima che la modifica diventi effettiva.

Il riavvio di un'istanza database può verificarsi quando si modifica un'impostazione che richiede un riavvio o quando il riavvio viene innescato manualmente. Un riavvio può verificarsi immediatamente se si modifica un'impostazione e si richiede che la modifica abbia effetto immediato. Oppure può verificarsi durante la finestra di manutenzione dell'istanza database.

Un riavvio dell'istanza del database si verifica immediatamente quando si verifica una delle seguenti condizioni:

- Il periodo di retention dei backup per un'istanza database viene modificato da 0 a un valore diverso da zero o da un valore diverso da zero a 0. Quindi si imposta Apply Immediately (Applica immediatamente) su `true`.
- Si modifica la classe di istanza database e si imposta Apply Immediately (Applica immediatamente) su `true`.

Un riavvio dell'istanza di database avviene durante la finestra di manutenzione quando si verifica una delle seguenti condizioni:

- Si modifica il periodo di retention dei backup per un'istanza database da 0 a un valore diverso da zero o da un valore diverso da zero a 0 e Apply Immediately (Applica immediatamente) è impostato su `false`.
- Si modifica la classe di istanza database e si imposta Apply Immediately (Applica immediatamente) su `false`.

Quando si modifica un parametro statico in un gruppo di parametri database, la modifica non diventa effettiva fino al riavvio dell'istanza database associata al gruppo di parametri. La modifica richiede un riavvio manuale. L'istanza database non viene riavviata automaticamente durante la finestra di manutenzione.

Modifiche ai parametri di database Amazon RDS che non hanno effetto

In alcuni casi, si potrebbe modificare un parametro in un gruppo di parametri database senza che le modifiche diventino effettive. In tal caso, è probabile che sia necessario riavviare l'istanza database associata al gruppo di parametri database. Quando si modifica un parametro dinamico, la modifica diventa immediatamente effettiva. Quando si modifica un parametro statico, la modifica non diventa effettiva finché l'istanza database associata al gruppo di parametri non viene riavviata.

Puoi riavviare un'istanza database utilizzando la console RDS. In alternativa puoi chiamare esplicitamente l'operazione API `RebootDBInstance`. È possibile riavviare senza failover se l'istanza database è in un'implementazione Multi-AZ. La necessità di riavviare l'istanza database associata dopo la modifica di un parametro statico consente di ridurre il rischio di errore di configurazione di un parametro che influenza una chiamata API. Un esempio è la chiamata di `ModifyDBInstance` per modificare la classe di istanza database. Per ulteriori informazioni, consulta Modifica dei parametri in un gruppo di parametri DB in .

Problemi di memoria liberabile in Amazon Aurora

La memoria liberabile è la memoria RAM (Random Access Memory) su un'istanza database che può essere resa disponibile al motore di database. È la somma della memoria libera del sistema operativo (OS) e della memoria disponibile del buffer e della cache delle pagine. Il motore di database utilizza la maggior parte della memoria sull'host, ma anche i processi del sistema operativo utilizzano RAM. La memoria attualmente allocata al motore di database o utilizzata dai processi del sistema operativo non è inclusa nella memoria liberabile. Quando il motore di database sta per esaurire la memoria, l'istanza database può utilizzare lo spazio temporaneo normalmente utilizzato per il buffering e la memorizzazione nella cache. Come accennato in precedenza, questo spazio temporaneo è incluso nella memoria liberabile.

Utilizzi la `FreeableMemory` metrica di Amazon CloudWatch per monitorare la memoria disponibile. Per ulteriori informazioni, consulta Strumenti di monitoraggio per Aurora.

Se la memoria liberabile dell'istanza database diventa insufficiente o viene utilizzato uno spazio di scambio, valutare l'aumento a una classe di istanza database più grande. Per ulteriori informazioni, consulta Classi di istanze DB Amazon Aurora.

Inoltre, puoi modificare le impostazioni della memoria. Ad esempio, in Aurora MySQL , puoi regolare le dimensioni del parametro `innodb_buffer_pool_size`. Questo parametro è impostato per default sul 75% della memoria fisica. Per ulteriori suggerimenti per la risoluzione di problemi di MySQL, consulta Come è possibile risolvere i problemi di memoria liberabile insufficiente in un database Amazon RDS per MySQL?

In Aurora Serverless v2, `FreeableMemory` rappresenta la quantità di memoria inutilizzata disponibile quando Aurora Serverless v2 L'istanza DB viene scalata fino alla capacità massima. È possibile che l'istanza sia stata ridotta a una capacità relativamente bassa, ma continui a segnalare un valore elevato per `FreeableMemory`, perché l'istanza può aumentare. Questa memoria non è disponibile al momento, ma puoi recuperarla se ti serve.

Per ogni unità di capacità Aurora (ACU) la cui capacità corrente è inferiore alla capacità massima, `FreeableMemory` aumenta di circa 2 GiB. Pertanto, questo parametro non si avvicina a zero finché non viene eseguito il dimensionamento verso l'alto dell'istanza database fino al valore più alto possibile.

Se questo parametro si avvicina al valore `0`, è stato raggiunto il massimo di aumento dell'istanza database. Ovvero al valore più prossimo al limite di memoria disponibile. Valuta la possibilità di aumentare l'impostazione ACU massima per il cluster. Se il valore di questo parametro si avvicina a `0` per un'istanza database di lettura, valuta la possibilità di aggiungere altre istanze database di lettura al cluster. In questo modo, puoi distribuire la parte di sola lettura del carico di lavoro su più istanze database e ridurre conseguentemente l'utilizzo della memoria su ogni istanza database di lettura. Per ulteriori informazioni, consulta CloudWatch Parametri Amazon importanti per Aurora Serverless v2.

In Aurora Serverless v1, è possibile modificare l'intervallo di capacità per utilizzarne di più ACUs. Per ulteriori informazioni, consulta Modificare un Aurora Serverless v1 cluster di database.

Problemi di replica relativi a Amazon Aurora MySQL

Alcuni problemi di replica MySQL si applicano anche a Aurora MySQL. Puoi diagnosticare e risolvere questi problemi.

Argomenti

- Diagnosi e risoluzione del ritardo tra repliche di lettura

- Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL
- Errore di replica interrotta
- La replica in lettura non riesce a inizializzare la struttura dei metadati

Diagnosi e risoluzione del ritardo tra repliche di lettura

Quando una replica di lettura MySQL viene creata ed è disponibile, Amazon RDS esegue per prima cosa la replica delle modifiche apportate all'istanza database di origine dal momento in cui l'operazione di creazione della replica di lettura è stata avviata. Durante questa fase, il ritardo di replica per la replica di lettura è maggiore di 0. Puoi monitorare questo ritardo in Amazon CloudWatch visualizzando la metrica Amazon RDS.

Il parametro `AuroraBinlogReplicaLag` indica il valore del campo `Seconds_Behind_Master` del comando `SHOW REPLICA STATUS` di MySQL. Per ulteriori informazioni, consulta Istruzione SHOW REPLICA STATUS nella documentazione di MySQL.

Quando il parametro `AuroraBinlogReplicaLag` è 0, la replica ha raggiunto l'istanza del database di origine. Se il parametro `AuroraBinlogReplicaLag` restituisce -1, la replica potrebbe non essere attiva. Per la risoluzione di problemi relativi a un errore di replica, consulta Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL. Un valore di `AuroraBinlogReplicaLag` pari a -1 può anche indicare che il valore di `Seconds_Behind_Master` non può essere determinato oppure è `NULL`.

> **Note**
>
> Versioni precedenti di Aurora MySQL utilizzavano `SHOW SLAVE STATUS` al posto di `SHOW REPLICA STATUS`. Se si utilizza una versione 1 o 2 Aurora MySQL, utilizzare `SHOW SLAVE STATUS`. Utilizzare `SHOW REPLICA STATUS` per Aurora MySQL versione 3 e successive.

Il parametro `AuroraBinlogReplicaLag` restituisce -1 durante un'interruzione della rete o quando viene applicata una patch durante la finestra di manutenzione. In questo caso, attendi fino al ripristino della connettività di rete o al termine della finestra di manutenzione prima di controllare nuovamente il parametro `AuroraBinlogReplicaLag`.

La tecnologia di replica di lettura di MySQL è asincrona. Pertanto, è possibile che si verifichino incrementi occasionali del parametro `BinLogDiskUsage` nell'istanza database di origine e del

parametro `AuroraBinlogReplicaLag` nella replica di lettura. Ad esempio, considera una situazione in cui si verifica un elevato volume di operazioni di scrittura nell'istanza database di origine in parallelo. Contemporaneamente, le operazioni di scrittura nella replica di lettura vengono serializzate utilizzando un singolo thread I/O. Tale situazione può portare a un ritardo tra l'istanza di origine e la replica di lettura.

Per ulteriori informazioni sulle repliche di lettura e su MySQL, consulta la pagina dei dettagli dell'implementazione di repliche nella documentazione di MySQL.

Puoi ridurre il ritardo tra gli aggiornamenti di un'istanza database di origine e i successivi aggiornamenti della replica di lettura attenendoti alla seguente procedura:

- Imposta la classe dell'istanza database della replica di lettura in modo che le sue dimensioni di storage siano paragonabili a quelle dell'istanza del database di origine.
- Assicurati che le impostazioni dei parametri dei gruppi di parametri database utilizzati dall'istanza database di origine e la replica di lettura siano compatibili. Per ulteriori informazioni e un esempio, consulta la discussione sul parametro `max_allowed_packet` nella sezione seguente.
- Disabilita la cache delle query. Per le tabelle modificate di frequente, l'uso della cache delle query può aumentare il ritardo della replica perché la cache viene bloccata e aggiornata spesso. In questo caso, potresti visualizzare un ritardo della replica inferiore se disabiliti la cache delle query. Puoi disabilitare la cache delle query impostando il parametro `query_cache_type parameter` sul valore 0 nel gruppo di parametri di database dell'istanza di database. Per ulteriori informazioni sulla cache delle query, consulta la sezione relativa alla configurazione della cache delle query.
- Riscaldare il buffer pool sulla replica di lettura per InnoDB per MySQL. Ad esempio, supponi di disporre di un set ridotto di tabelle che vengono aggiornate di frequente e di utilizzare lo schema di tabella InnoDB o XtraDB. In questo caso, esegui il dump di tali tabelle nella replica di lettura In questo modo, il motore di database esegue la scansione delle righe delle tabelle del disco e le memorizza nella cache nel pool di buffer, il che può ridurre il ritardo della replica. Di seguito viene riportato un esempio.

  In Linux, macOS, oppure Unix:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

In Windows:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

Diagnosi e risoluzione di un errore relativo alla replica di lettura MySQL

Amazon RDS monitora lo stato delle repliche di lettura. RDS aggiorna il campo Replication State (Stato di replica) dell'istanza della replica di lettura con il valore `Error`, se la replica viene arrestata per qualsiasi motivo. Puoi rivedere i dettagli dell'errore associato generato dai motori MySQL visualizzando il campo Replication Error (Errore di replica). Vengono generati anche eventi che indicano lo stato della replica di lettura, inclusi RDS-EVENT-0045, RDS-EVENT-0046 e RDS-EVENT-0057. Per ulteriori informazioni sugli eventi e sulla sottoscrizione a essi, consulta Utilizzo delle notifiche di RDS eventi di Amazon. Se viene restituito un messaggio di errore MySQL, controlla l'errore nella documentazione dei messaggi di errore MySQL.

Situazioni comuni che possono causare errori di replica sono:

- Il valore del parametro `max_allowed_packet` di una replica di lettura è inferiore al parametro `max_allowed_packet` dell'istanza database di origine.

  Il parametro `max_allowed_packet` è un parametro personalizzato che puoi impostare in un gruppo di parametri database. Il parametro `max_allowed_packet` viene utilizzato per specificare la dimensione massima delle query DML (Data Manipulation Language) che possono essere eseguite sul database. In alcuni casi, il valore `max_allowed_packet` dell'istanza database di origine potrebbe essere più grande del valore `max_allowed_packet` per la replica di lettura. In questo caso, il processo di replica può generare un errore e interrompere la replica. L'errore più comune è `packet bigger than 'max_allowed_packet' bytes`. Puoi correggere questo errore impostando l'origine e la replica di lettura in modo che utilizzino i gruppi di parametri database con gli stessi valori del parametro `max_allowed_packet`.
- Scrittura in tabelle su una replica di lettura. Se crei indici su una replica di lettura, il parametro `read_only` deve essere impostato su 0 affinché gli indici vengano creati. Se esegui la scrittura su tabelle sulla replica di lettura, ciò può interrompere la replica.

- Uso di un motore di storage non transazionale come MyISAM. Le repliche di lettura richiedono un motore di storage transazionale. La replica è supportata solo per i seguenti motori di archiviazione: InnoDB per MySQL o MariaDB.

  Puoi convertire una tabella MyISAM in InnoDB, utilizzando il comando seguente:

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```

- Utilizzo di query non deterministiche non sicure come `SYSDATE()`. Per ulteriori informazioni, consulta la pagina relativa alla determinazione delle istruzioni sicure e non sicure nel logging binario nella documentazione MySQL.

La seguente procedura può essere di aiuto per la risoluzione dell'errore di replica:

- Se riscontri un errore logico e puoi ignorarlo in modo sicuro, segui le fasi descritte in Ignorare l'errore di replica corrente. L'istanza database Aurora MySQL deve eseguire una versione che includa la procedura `mysql_rds_skip_repl_error`. Per ulteriori informazioni, consulta mysql_rds_skip_repl_error.
- Se si verifica un problema di posizione del registro binario (binlog), puoi modificare la posizione di riproduzione della replica. Per farlo, utilizza il comando `mysql.rds_next_master_log` per Aurora MySQL versione 1 e 2. Per farlo, utilizza il comando `mysql.rds_next_source_log` per Aurora MySQL versione 3 e successive. L'istanza database Aurora MySQL deve eseguire una versione che supporta questo comando per modificare la posizione di riproduzione della replica. Per informazioni sulle versioni, consulta mysql_rds_next_master_log.
- Se si verifica un problema temporaneo di prestazioni a causa dell'elevato carico DML, è possibile impostare il `innodb_flush_log_at_trx_commit` parametro su 2 nel gruppo di parametri DB sulla replica di lettura. Ciò può agevolare il recupero delle prestazioni della replica di lettura, sebbene le proprietà ACID (atomicità, consistenza, isolamento e durata) subiscano una riduzione temporanea.
- Puoi eliminare la replica di lettura e creare un'istanza utilizzando lo stesso identificatore istanze DB. In questo modo, l'endpoint rimane identico a quello della vecchia replica di lettura.

Quando un problema relativo alla replica viene risolto, il campo Replication State (Stato di replica) cambia in replicating (replica in corso). Per ulteriori informazioni, consulta Risoluzione dei problemi relativi a una replica di lettura MySQL.

Errore di replica interrotta

Quando si chiama il comando `mysql.rds_skip_repl_error`, è possibile che venga visualizzato un messaggio di errore che indica che la replica è inattiva o disattivata.

Questo messaggio di errore viene visualizzato perché la replica è stata arrestata e non può essere riavviata.

Se devi ignorare un numero elevato di errori, il ritardo della replica potrebbe superare il periodo di retention predefinito per i file di log binari. In questo caso, è possibile che si verifichi un errore irreversibile a causa dell'eliminazione dei file di registro binari prima che vengano riprodotti sulla replica. Questa eliminazione causa l'arresto della replica e non è più possibile chiamare il comando `mysql.rds_skip_repl_error` per ignorare errori di replica.

Puoi limitare questo problema aumentando il numero di ore di retention dei file di log binari nella sorgente di replica. Una volta aumentato il tempo di retention dei file binlog, puoi riavviare la replica e chiamare il comando `mysql.rds_skip_repl_error` secondo necessità.

Per impostare il periodo di retention dei file binlog, utilizza la procedura mysql_rds_set_configuration . Specifica un parametro di configurazione di "ore di conservazione dei file binlog” insieme al numero di ore di conservazione dei file binlog nel cluster di database, fino a 2160 (90 giorni). Il valore predefinito per Aurora MySQL è 24 (1 giorno). Nell'esempio seguente il periodo di retention dei file binlog è impostato su 48 ore.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

La replica in lettura non riesce a inizializzare la struttura dei metadati

Quando hai tentato di avviare la replica, hai ricevuto il seguente messaggio di errore:

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

Questo errore si verifica quando c'è un problema con la struttura dei metadati della replica. Per correggere la struttura dei metadati, è necessario creare una nuova replica.

Per evitare che ciò accada in futuro, esegui una delle seguenti azioni:

- Se possibile, disattiva il multithreading sulle repliche. A partire da MySQL 8.0.27, il multithreading è abilitato per impostazione predefinita.

- Se è necessario utilizzare il multithreading sulle repliche, si consiglia di utilizzare la replica basata su GTID. Per ulteriori informazioni, consulta Utilizzo della replica GTID basata.

Documentazione di riferimento dell'API Amazon RDS

Oltre alla AWS Management Console e AWS Command Line Interface (AWS CLI), Amazon RDS fornisce anche un'API. È possibile usare l'API per automatizzare le attività per la gestione delle istanze database e altri oggetti in Amazon RDS.

- Per un elenco alfabetico delle operazioni API, consulta Operazioni.
- Per un elenco alfabetico dei tipi di dati, consulta la pagina Tipi di dati.
- Per un elenco di parametri di query comuni, consulta la pagina Parametri Comuni.
- Per le descrizioni dei codici di errore, consulta la pagina Errori comuni.

Per ulteriori informazioni sui AWS CLI, consulta Riferimento AWS Command Line Interface per Amazon RDS.

Argomenti

- Uso dell'API query
- Risoluzione dei problemi delle applicazioni in Aurora

Uso dell'API query

Le sezioni seguenti illustrano brevemente le autenticazioni dei parametri e delle richieste usate con l'API query.

Per informazioni generali sul funzionamento dell'API Query, consulta Richieste di query in Amazon EC2 API Reference.

Parametri di query

Le richieste basate su query HTTP sono richieste HTTP che utilizzano i verbi HTTP GET oppure POST e un parametro di query denominato `Action`.

Ogni richiesta di query deve includere alcuni parametri comuni per gestire l'autenticazione e la selezione di un'azione.

Alcune operazioni accettano elenchi di parametri. Questi elenchi sono specificati usando l'annotazione `param.`*n*. I valori di *n* sono numero a partire da 1.

Per ulteriori informazioni su endpoint e regioni Amazon RDS, consulta Amazon Relational Database Service (RDS) nella sezione Regioni ed endpoint di Riferimenti generali di Amazon Web Services.

Autenticazione delle richieste di query

È possibile inviare solo richieste di query tramite HTTPS ed è necessario includere una firma in ogni richiesta di query. È necessario usare AWS Signature Version 4 o Signature Version 2. Per ulteriori informazioni, consulta Processo di firma Signature Version 4 e Processo di firma Signature Version 2.

Risoluzione dei problemi delle applicazioni in Aurora

Amazon RDS fornisce errori specifici e descrittivi per aiutarti a risolvere i problemi mentre interagisci con l'API Amazon RDS.

Argomenti

- Errore durante il recupero
- Suggerimenti per la risoluzione dei problemi

Per informazioni sulla risoluzione dei problemi per le istanze database Amazon RDS, consulta Risoluzione dei problemi per Amazon Aurora.

Errore durante il recupero

In genere, si desidera che l'applicazione verifichi se una richiesta ha generato un errore prima di trascorrere del tempo a elaborare i risultati. Il modo più semplice per determinare se si è verificato un errore consiste nel cercare un nodo `Error` nella risposta dall'API Amazon RDS.

La sintassi XPath fornisce un modo semplice per rilevare la presenza di un nodo `Error`. Fornisce inoltre un modo relativamente semplice per recuperare il messaggio e il codice di errore. Il seguente snippet di codice usa Perl e il modulo XML::XPath per determinare se si è verificato un errore durante una richiesta. Se si è verificato un errore, il codice stampa il primo codice di errore e il messaggio nella risposta.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
    {print "There was an error processing your request:\n", " Error code: ",
```

```
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Suggerimenti per la risoluzione dei problemi

Ti consigliano i seguenti processi per diagnosticare e risolvere i problemi con l'API Amazon RDS:

- Verifica che Amazon RDS funzioni normalmente nella regione AWS di destinazione controllando http://status.aws.amazon.com.
- Verificare la struttura della richiesta.

  Ogni operazione Amazon RDS ha una pagina di riferimento nella documentazione di riferimento dell'API Amazon RDS. Controllare nuovamente che si stia usando i parametri correttamente. Per idee su cosa potrebbe essere sbagliato, guarda le richieste di esempio o gli scenari utente per vedere se tali esempi stanno eseguendo operazioni simili.
- Controlla AWS re:Post.

  Amazon RDS dispone di una community di sviluppo in cui puoi cercare soluzioni ai problemi che altri hanno riscontrato lungo il percorso. Per vedere gli argomenti, consulta AWS re:Post.

Cronologia dei documenti

Versione corrente dell'API: 2014-10-31

Nella tabella seguente sono descritte le modifiche importanti apportate alla Guida per l'utente di Amazon Aurora. Per ricevere notifiche sugli aggiornamenti di questa documentazione, puoi abbonarti a un feed RSS. Per ulteriori informazioni su Amazon Relational Database Service (Amazon RDS), consultare la Guida per l'utente di Amazon Relational Database Service.

> **Note**
>
> Prima del 31 agosto 2018, Amazon Aurora era documentato nella Guida per l'utente di Amazon Relational Database Service. Per la cronologia dei documenti di Aurora precedente, consulta Cronologia dei documenti nella Guida per l'utente di Amazon Relational Database Service.

Please change to "Puoi filtrare le nuove funzionalità Amazon Aurora alla pagina Quali sono le novità del database?. In Prodotti, seleziona Amazon Aurora. Quindi esegui la ricerca utilizzando parole chiave come **`global database`** o **`Serverless`**.

Modifica	Descrizione	Data
End-of-life informazioni per Performance Insights	AWS ha annunciato la end-of-life data di Performance Insights: 30 novembre 2025. Dopo questa data, Amazon RDS non supporterà più l'esperienza della console Performance Insights, i periodi di conservazione flessibili e i relativi prezzi. Ti consiglia mo di aggiornare i cluster DB a Database Insights prima di questa data. Per ulteriori informazioni, consulta Monitoraggio del carico del DB	30 maggio 2025

	con Performance Insights su Amazon Aurora.	
Il database Aurora PostgreSQL Limitless supporta Database Insights CloudWatch	Ora puoi usare CloudWatch Database Insights per monitorare il tuo database Amazon Aurora PostgreSQL Limitless su larga scala. Per ulteriori informazioni, consulta Monitoraggio del database Amazon Aurora PostgreSQL Limitless con Database Insights. CloudWatch	22 maggio 2025
Visualizzazione delle date di supporto per le versioni principali dei motori open source	È ora possibile visualizzare le informazioni sulle date di inizio e di fine per le versioni principali dei motori open source in RDS Standard Support e RDS Extended Support utilizzando l'API AWS CLI e RDS. Per ulteriori informazioni, consulta Visualizzazione delle date di supporto per le versioni del motore in Amazon RDS Extended Support.	20 maggio 2025
Le integrazioni Aurora PostgreSQL Zero-ETL supportano regioni aggiuntive	Le integrazioni Aurora PostgreSQL Zero-ETL sono ora disponibili in regioni aggiuntive. Per ulteriori informazioni, consulta Regioni supportate e motori Aurora DB per integrazioni zero-ETL con Amazon Redshift.	17 febbraio 2025

End-of-life informazioni per Aurora Serverless v1	AWS ha annunciato la end-of-life data delAurora Serverless v1: 31 marzo 2025. Consigliamo vivamente di aggiornare qualsiasi cluster Aurora Serverless v1 DB prima di tale data. Aurora Serverless v2 L'aggiornamento può comportare una modifica del numero di versione principale del motore di database. Pertanto, è importante pianificare, testare e implementare questo passaggio prima della end-of-life data. A partire dall'8 gennaio 2025, i clienti non saranno più in grado di creare nuovi Aurora Serverless v1 cluster o istanze né con la CLI. AWS Management Console Per informazioni sulla procedura di aggiornamento da Aurora Serverless v1 aAurora Serverless v2, consulta Aggiornamento da un cluster a. Aurora Serverless v1 Aurora Serverless v2	6 dicembre 2024

Aurora supporta CloudWatch Database Insights	Ora puoi usare CloudWatch Database Insights per monitorare i tuoi cluster Amazon Aurora DB su larga scala. Per ulteriori informazioni, consulta Monitoraggio dei database Amazon Aurora con CloudWatch Database Insights.	1 dicembre 2024
Amazon Aurora supporta le classi di istanze db.r8g	È ora possibile utilizzare le classi di istanze db.r8g per creare cluster Aurora DB. Per ulteriori informazioni, consulta Classi di istanze database Aurora.	21 novembre 2024
Aurora Serverless v2supporta la pausa e la ripresa automatiche	Aurora Serverless v2Le istanze DB possono ora scalare fino a zero unità di capacità Aurora ACUs () dopo un periodo di inattività. Per abilitare questa funzionalità, impostate il valore di capacità minima su 0. ACUs Per ulteriori informazioni, consulta Pausa e ripresa automatiche per. Aurora Serverless v2	20 novembre 2024

Amazon Aurora supporta la migrazione automatica dei database EC2	È possibile utilizzare la console RDS per migrare un EC2 database in Aurora. Aurora utilizza AWS Database Migration Service (AWS DMS) per migrare il database di origine. EC2 AWS DMS ti consente di migrare i database relazionali nel tuo. Cloud AWS Per ulteriori informazioni, consulta Migrazione automatica dei EC2 database su Amazon Aurora AWS Database Migration Service utilizzando.	20 novembre 2024
AWS Wrapper NodeJS avanzato generalmente disponibile	Amazon Web Services (AWS) Advanced NodeJS Wrapper è complementare ed estende la funzionalità di un driver NodeJS esistente. Per ulteriori informazioni, consulta Connessione ai cluster Aurora DB con i driver. AWS	19 novembre 2024
Amazon Aurora supporta le classi di istanze db.r7i	È ora possibile utilizzare le classi di istanze db.r7i per creare cluster Aurora DB. Per ulteriori informazioni, consulta Classi di istanze database Aurora.	18 novembre 2024

RDS Data API per Aurora MySQL supporta tutte le viste Performance Insights	Ora puoi utilizzare le viste Top hosts o Top applications per Performance Insights per monitorare le operazioni SQL eseguite utilizzando RDS Data API per Aurora MySQL. Per informazioni sul supporto di Performance Insights, consulta Monitoraggio delle query RDS Data API con Performance Insights.	31 ottobre 2024
Database Aurora PostgreSQL Limitless	Amazon Aurora PostgreSQL Limitless Database è una nuova funzionalità di scalabilità orizzontale (sharding) automatizzata di Amazon Aurora. Aurora PostgreSQL Limitless Database consente di superare i limiti esistenti di Aurora per la velocità effettiva di scrittura e lo storage distribuendo il carico di lavoro del database su più istanze di Aurora writer, pur mantenendo la possibilità di utilizzarlo come un singolo database. Per ulteriori informazioni, consulta Amazon Aurora PostgreSQL Limitless Database.	31 ottobre 2024

Fine del supporto standard per Amazon Aurora MySQL versione 2	Aurora MySQL versione 2 ha raggiunto la fine del supporto standard il 31 ottobre 2024. Per ulteriori informazioni, consulta Preparazione per la fine del supporto standard dell'edizione 2 compatibile con Amazon Aurora MySQL.	31 ottobre 2024
Amazon Aurora supporta gli aggiornamenti del sistema operativo a livello di cluster	È ora possibile applicare gli aggiornamenti del sistema operativo Aurora a livello di cluster DB. Gli aggiornamenti in sequenza applicano automaticamente gli aggiornamenti a poche istanze DB di Reader alla volta, preservando così la disponibilità di lettura. Per ulteriori informazioni, consulta Aggiornamenti del sistema operativo per i cluster Aurora DB.	30 ottobre 2024

I cluster globali Aurora supportano il tagging	Ora puoi aggiungere tag ai cluster globali Aurora. Un cluster globale Aurora è la risorsa principale che contiene i cluster DB primari e secondari in un database globale Aurora. Per informazioni generali sull'etichettatura, consulta Etichettare le risorse di Aurora e Amazon RDS. Per informazioni sull'etichettatura con Aurora Global Database, vedere Tagging Aurora Global Database resources.	24 ottobre 2024
Endpoint Aurora Global Database Writer	Ogni Aurora Global Database ora è dotato di un endpoint writer che viene aggiornato automaticamente da Aurora per instradare le richieste all'istanza di scrittura corrente del cluster DB primario dopo un'operazione di switchover e failover di Aurora Global Database. Per ulteriori informazioni sull'utilizzo dell'endpoint writer insieme allo switchover e al failover di Aurora Global Database, consulta Using switchover o failover in Amazon Aurora Global Database.	22 ottobre 2024

Le integrazioni Aurora PostgreSQL Zero-ETL con Amazon Redshift sono generalmente disponibili	Le integrazioni zero-ETL rendono disponibili i dati transazionali in Amazon Redshift entro pochi secondi dalla loro scrittura su un cluster DB di origine Aurora PostgreSQL. La funzionalità è ora disponibile a livello generale. Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	15 ottobre 2024
Aurora Serverless v2supporta 256 ACUs	È ora possibile creare cluster Aurora Serverless v2 DB con una capacità massima di 256 unità ACUs di capacità Aurora (). Per ulteriori informazioni, consulta la sezione Aurora Serverless v2 capacità.	3 ottobre 2024
Amazon Aurora supporta Console-to-Code	Ora puoi usarlo Console-to-Code per generare codice a partire da azioni eseguite nella console RDS. Il codice generato può aiutarti a scrivere codice per automatizzare l'uso di altri AWS servizi. Per ulteriori informazioni, consulta Utilizzare Console-to-Code per generare codice per le azioni della console Aurora.	3 ottobre 2024

L'API dati RDS supporta Performance Insights	Ora puoi utilizzare Performance Insights per monitorare le operazioni SQL eseguite utilizzando RDS Data API. Per informazioni sul supporto di Performance Insights, consulta Monitoraggio delle query RDS Data API con Performance Insights.	26 settembre 2024
L'API RDS Data Aurora Serverless v2 e il provisioning sono disponibili per Aurora MySQL	L'API RDS Data è ora disponibile per i cluster Aurora MySQL che utilizzano o forniscono istanze. Aurora Serverless v2 Per informazioni sull'utilizzo di RDS Data API, consulta Using RDS Data API. Per informazioni sulla versione di Aurora MySQL e sul supporto AWS regionale per RDS Data API, consulta Data API with Aurora MySQL Serverless v2 e provisioned.	26 settembre 2024
Amazon Aurora è disponibile nella regione Asia Pacifico (Malesia)	Amazon Aurora è ora disponibile nella regione Asia Pacifico (Malesia). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	22 agosto 2024

Aggiornamento alla politica esistente	Amazon RDS ha rimosso `sns:Publish` l'autorizzazione dal `AmazonRDSBetaServiceRolePolicy` ruolo collegato al `AWSServiceRoleForRDSBeta` servizio. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	7 agosto 2024
Aggiornamento alla politica esistente	Amazon RDS ha rimosso `sns:Publish` l'autorizzazione dal `AmazonRDSPreviewServiceRolePolicy` ruolo collegato al `AWSServiceRoleForRDSPreview` servizio. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	7 agosto 2024
Aurora Serverless v2supporta la pausa automatica () scale-to-zero	Aurora Serverless v2Le istanze DB possono ora scalare fino a zero unità di capacità Aurora ACUs () dopo un periodo di inattività. Per abilitare questa funzionalità, impostate il valore di capacità minima su 0. Per ulteriori informazioni, consulta Utilizzo di Aurora Serverless v2.	25 luglio 2024

AWS Driver ODBC per MySQL generalmente disponibile	Il driver ODBC di Amazon Web Services (AWS) per MySQL è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Per ulteriori informazioni, consulta Connessione ad Aurora MySQL con il driver ODBC di Amazon Web Services ()AWS per MySQL.	18 luglio 2024
RDS Data API for è disponibile in altre regioni Aurora Serverless v2	L'API RDS Data è ora disponibile per Aurora PostgreSQL in diverse regioni aggiuntive. AWS Per informazioni sul supporto regionale per RDS Data API, consulta Data API with Aurora PostgreSQL Serverless v2 e provisioned.	9 luglio 2024
AWS Il driver Python è disponibile a livello generale	Il driver Python di Amazon Web Services (AWS) è progettato come wrapper Python avanzato. Questo wrapper è complementare ed estende le funzionalità del driver open source Psycopg. Per ulteriori informazioni, consulta Connessione ai cluster Aurora DB con i driver. AWS	23 maggio 2024

Integrazioni zero-ETL disponibili nelle regioni della Cina	Le integrazioni zero-ETL sono ora disponibili in Cina (Pechino) e Regioni AWS Cina (Ningxia). Per ulteriori informazioni, consulta Integrazioni zero-ETL con Amazon Redshift.	21 maggio 2024
RDS Proxy è disponibile in altre regioni	Il proxy RDS è ora disponibi le nelle regioni Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Medio Oriente (Emirati Arabi Uniti), Israele (Tel Aviv), Canada occidentale (Calgary) ed Europa (Zurigo). Per ulteriori informazioni sul proxy RDS, consulta Utilizzo di Server proxy per Amazon RDS.	21 maggio 2024
Amazon RDS supporta l'accesso granulare per Performance Insights	Ora puoi consentire o negare l'accesso a singole dimension i in Performance Insights. Questo accesso granulare può essere utilizzato per `GetResourceMetrics` e azioni. `DescribeD imensionKeys` `GetDimensionKeyDet ails` Per ulteriori informazi oni, consulta https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.access-control.dimensionAccess-policy.html.	21 maggio 2024

Estensione del supporto per Amazon RDS	La creazione o il ripristino di un database Aurora MySQL versione 2 o 3 o Aurora PostgreSQL versione 11 ora registra automaticamente quel database in Amazon RDS Extended Support in modo che le applicazioni esistenti continuino a funzionare così come sono. Puoi disattivare RDS Extended Support per evitare addebiti dopo la fine della data di supporto standard di Aurora per il tuo motore di database. Per ulteriori informazioni, consulta Utilizzo dell'estensione del supporto per Amazon RDS.	21 marzo 2024
Filtraggio dei dati per integrazioni zero-ETL	Amazon RDS supporta il filtraggio dei dati a livello di database e tabella per integrazioni zero-ETL con Amazon Redshift. Per ulteriori informazioni, consulta Filtraggio dei dati per le integrazioni Aurora zero-ETL con Amazon Redshift.	20 marzo 2024

Integrazioni MySQL di Aurora con Amazon Bedrock	Ora puoi integrare i database Amazon Aurora MySQL con Amazon Bedrock per alimentare applicazioni di intelligenza artificiale generativa. Per ulteriori informazioni, consulta Uso dell'apprendimento automatico di Amazon Aurora con Aurora MySQL.	8 marzo 2024
Nuova politica gestita AWS	Amazon RDS ha aggiunto una nuova policy gestita denominata `AmazonRDSCustomInstanceProfileRolePolicy` per consentire a RDS Custom di eseguire azioni di automazione e attività di gestione del database tramite un profilo di EC2 istanza. Per ulteriori informazioni, consulta Aggiornamenti di Amazon RDS sulle policy gestite da AWS.	27 febbraio 2024
Supporto Amazon RDS per la AWS Secrets Manager regione di Israele (Tel Aviv)	Amazon RDS supporta Secrets Manager nella regione di Israele (Tel Aviv). Per ulteriori informazioni, consulta Password management with Amazon RDS and AWS Secrets Manager (Gestione delle password per Amazon RDS e AWS Secrets Manager).	21 febbraio 2024

Estensione del supporto per Amazon RDS	Amazon RDS ora abilita automaticamente Amazon RDS Extended Support quando le versioni principali del motore Aurora MySQL e Aurora PostgreSQL nei cluster DB e nei cluster globali raggiungono la data di fine del supporto standard di Aurora. Per ulteriori informazioni, consulta Utilizzo dell'estensione del supporto per Amazon RDS.	15 febbraio 2024
Aurora PostgreSQL 16.1 supporta Babelfish per Aurora PostgreSQL 4.0.0	Aurora PostgreSQL 16.1 supporta Babelfish 4.0.0. Per un elenco delle nuove funzionalità, consulta 16.1. Per un elenco delle caratteristiche supportate in ogni versione di Babelfish, consulta Funzionalità supportate in Babelfish per versione. Per le informazioni di utilizzo, consulta Lavorare con Babelfish per Aurora PostgreSQL.	31 gennaio 2024
Aggiornamento al certificato CA predefinito	Il certificato CA predefinito è impostato su `rds-ca-rsa2048-g1` . Per ulteriori informazioni, vedere Utilizzo di SSL/TLS per crittografare una connessione a un cluster DB.	26 gennaio 2024

RDS Proxy è disponibile nella regione Europa (Spagna)	RDS Proxy è ora disponibile nella regione Europa (Spagna). Per ulteriori informazioni sul proxy RDS, consulta Utilizzo di Server proxy per Amazon RDS.	8 gennaio 2024
API dati RDS con Aurora PostgreSQL Serverless v2 e provisioning	Ora puoi usare RDS Data API con Aurora PostgreSQL Serverless v2 e cluster DB con provisioning. Con RDS Data API, puoi accedere ai cluster Aurora tramite un endpoint HTTP sicuro ed eseguire istruzioni SQL senza utilizzare driver di database o gestire connessioni. Per ulteriori informazioni, consulta Using RDS Data API.	21 dicembre 2023
Integrazioni Aurora PostgreSQL con Amazon Bedrock	Ora puoi integrare i database Amazon Aurora PostgreSQL con Amazon Bedrock per alimentare applicazioni di intelligenza artificiale generativa. Per ulteriori informazioni, consulta Usare l'apprendimento automatico di Amazon Aurora con Aurora PostgreSQL.	21 dicembre 2023
Amazon Aurora è disponibile nella regione Canada occidentale (Calgary)	Amazon Aurora è ora disponibile nella regione Canada occidentale (Calgary). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	20 dicembre 2023

Amazon RDS supporta la visualizzazione e la risposta ai consigli	I consigli di Amazon Aurora ora includono consigli proattivi basati su soglie e reattivi basati sull'apprendimento automatico. Per ulteriori informazioni, consulta Consigli per Amazon Aurora.	19 dicembre 2023
Integrazioni Aurora PostgreSQL Zero-ETL con Amazon Redshift (anteprima)	Ora puoi creare integrazioni zero-ETL con Amazon Redshift utilizzando un cluster DB di origine PostgreSQL Aurora. Per la versione di anteprima, devi creare tutte le integrazioni nell'ambiente di anteprima del database Amazon RDS, negli Stati Uniti orientali (Ohio) (us-east-2). Regione AWS Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	28 novembre 2023
Amazon Aurora PostgreSQL supporta l'inoltro di scrittura in un database globale	È ora possibile abilitare l'inoltro di scrittura su cluster secondari in un database globale basato su Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'inoltro di scrittura in un database globale Aurora PostgreSQL.	9 novembre 2023

Supporto Aurora PostgreSQL per letture ottimizzate	Con Letture ottimizzate per Aurora è possibile velocizzare l'elaborazione delle query per Aurora PostgreSQL. Per ulteriori informazioni, consulta Prestazioni delle query migliorate per Aurora PostgreSQL con Letture ottimizzate per Aurora.	8 novembre 2023
Amazon RDS esporta i parametri di Performance Insights in Amazon CloudWatch	Performance Insights ti consente di esportare i dashboard delle metriche preconfigurati o personalizzati su Amazon. CloudWatch I dashboard delle metriche esportate possono essere visualizzati nella console. CloudWatch Puoi anche esportare un widget metrico di Performance Insights selezionato e visualizzare i dati delle metriche nella CloudWatch console. Per ulteriori informazioni, consulta Esportazione delle metriche di Performance Insights in. CloudWatch	8 novembre 2023

Integrazioni Zero-ETL di Aurora MySQL con Amazon Redshift disponibili a livello generale	Le integrazioni Zero-ETL con Amazon Redshift sono ora disponibili a livello generale per Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	7 novembre 2023
Supporto di Aurora PostgreSQL per implementazioni blu/verde di Amazon RDS	Ora puoi creare un'implementazione blu/verde da un cluster DB Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo delle implementazioni blu/verde Amazon RDS per gli aggiornamenti del database.	26 ottobre 2023
Aurora MySQL supporta la crittografia lato server con (SSE-KMS) AWS KMS keys	In Aurora MySQL versione 3.05 e successive, puoi utilizzare SSE-KMS, incluse Chiavi gestite da AWS le chiavi gestite dai clienti, per la crittografia lato server dei dati caricati o salvati su Amazon S3. Per ulteriori informazioni, consulta Caricamento dei dati in un cluster DB Amazon Aurora MySQL da file di testo in un bucket Amazon S3 e Salvataggio dei dati da un cluster DB Amazon Aurora MySQL nei file di testo in un bucket Amazon S3.	25 ottobre 2023

Le ottimizzazioni di Aurora MySQL riducono il tempo di riavvio del database	In Aurora MySQL 3.05 e versioni successive abbiamo introdotto ottimizzazioni che riducono il tempo di riavvio del database. Queste ottimizzazioni consentono di ridurre fino al 65% i tempi di inattività rispetto alle ottimizzazioni precedenti e di ridurre le interruzioni dei carichi di lavoro del database dopo un riavvio. Per ulteriori informazioni, consulta Ottimizzazioni per ridurre i tempi di riavvio del database.	25 ottobre 2023
AWS Aggiornamento alle politiche gestite	Le policy gestite `AmazonRDSPerformanceInsightsReadOnly` e `AmazonRDSPerformanceInsightsFullAccess` ora includono `Sid` (ID istruzione) come identificativo nell'istruzione della policy. Per ulteriori informazioni, consulta Aggiornamenti di Amazon RDS sulle policy gestite da AWS.	23 ottobre 2023

Amazon RDS pubblica i parametri dei contatori di Performance Insights su Amazon CloudWatch	La funzione matematica dei parametri DB_PERF_INSIGHTS nella console CloudWatch consente di interrogare i contatori di Amazon RDS for Performance Insights. Per ulteriori informazioni, consulta Creazione di CloudWatch allarmi per monitorare Amazon Aurora.	20 settembre 2023
Amazon Aurora supporta il point-in-time ripristino (PITR) con AWS Backup	Ora puoi gestire i backup automatici (continui) di Aurora AWS Backup e ripristinarli a orari specifici da essi. Per informazioni, consulta Ripristino di un cluster di database a un determinato momento (AWS Backup).	7 settembre 2023
Estensione del supporto per Amazon RDS	Amazon Aurora annuncia la possibilità a breve di continuare a eseguire le versioni principali del motore Aurora per MySQL e Aurora per PostgreSQL nelle istanze database oltre la data di fine del supporto standard Aurora. Per ulteriori informazioni, consulta Utilizzo dell'estensione del supporto per Amazon RDS.	1 settembre 2023

Amazon Aurora MySQL estende il supporto per Percona XtraBackup	Ora puoi eseguire migrazioni fisiche dei database MySQL 8.0 verso i cluster database Aurora MySQL versione 3. Per ulteriori informazioni, consulta Migrazione fisica da MySQL utilizzando Percona XtraBackup e Amazon S3.	24 agosto 2023
Il database globale Aurora ora supporta il failover globale del database	Il database globale Aurora ora supporta il failover globale gestito, che consente di semplificare il ripristino in seguito a un reale guasto a livello regionale o a un'interruzione completa del livello di servizio. Per ulteriori informazioni su questa funzionalità, consulta Esecuzione di failover gestiti per database globali Amazon Aurora. La funzionalità precedentemente denominata "failover pianificato gestito" è ora denominata "switchover". Per informazioni sugli switchover, consulta Performing switchovers for Amazon Aurora global databases (Esecuzione di switchover per database globali Amazon Aurora).	21 agosto 2023

Aggiornamento delle autorizzazioni delle policy gestite AWS	La policy gestita `AmazonRDSFullAccess` dispone di nuove autorizzazioni che consentono di generare, visualizzare ed eliminare il report di analisi delle prestazioni per un periodo di tempo. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	17 agosto 2023
Aggiornamento delle autorizzazioni delle policy AWS gestite	L'aggiunta di nuove autorizzazioni alla policy gestita `AmazonRDSPerformanceInsightsReadOnly` e l'aggiunta di una nuova policy gestita `AmazonRDSPerformanceInsightsFullAccess` consente di generare un report di analisi del carico del database per un periodo di tempo. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	16 agosto 2023

Amazon RDS supporta l'analisi del periodo di caricamento del database con Performance Insights	Performance Insights consente di creare report di analisi delle prestazioni per un periodo di tempo specifico. Il report fornisce gli approfondimenti identificati e i suggerimenti per risolvere problemi relativi alle prestazioni. Per ulteriori informazioni, consulta Analisi del carico del database per un periodo di tempo.	16 agosto 2023
Amazon Aurora supporta il mantenimento dei backup automatici per i cluster database	È ora possibile mantenere i backup automatici per i cluster Aurora eliminati e ripristinarli in un momento specifico. Per ulteriori informazioni, consulta Mantenimento dei backup automatici.	4 agosto 2023
Amazon Aurora è disponibile nella regione Israele (Tel Aviv)	Amazon Aurora è ora disponibile nella regione Israele (Tel Aviv). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	1° agosto 2023

Amazon Aurora MySQL supporta l'inoltro di scrittura locale (in un cluster)	È ora possibile inoltrare operazioni di scrittura da un'istanza database di lettura a un'istanza database di scrittura all'interno di un cluster database Aurora MySQL. Per ulteriori informazioni, consulta la sezione relativa all'utilizzo dell'inoltro di scrittura locale in un cluster database Amazon Aurora MySQL.	31 luglio 2023
Amazon Aurora supporta Aurora Serverless v2 in un altro Regione AWS	Ora puoi creare cluster Aurora Serverless v2 DB nella regione Asia-Pacifico (Melbourne). Regione AWS Per ulteriori informazioni su Aurora Serverless v2, consultar e Utilizzo di Aurora Serverless v2.	28 giugno 2023
Amazon Aurora introduce integrazioni Zero-ETL con Amazon Redshift (anteprima)	Le integrazioni Zero-ETL forniscono una soluzione completamente gestita per rendere disponibili i dati transazionali in Amazon Redshift in pochi secondi dalla loro scrittura su un cluster database Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo delle integrazioni Zero-ETL di Aurora con Amazon Redshift.	28 giugno 2023

Amazon RDS offre una visualizzazione combinata di Performance Insights e CloudWatch metriche nella dashboard di Performance Insights.	Amazon RDS ora offre una visualizzazione consolidata di Performance Insights e delle CloudWatch metriche nella dashboard di Performance Insights. Per ulteriori informazioni, consulta Visualizzazione delle metriche combinate con il dashboard Performance Insights.	24 maggio 2023
Amazon Aurora supporta le classi di istanza db.r7g	Ora è possibile utilizzare le classi di istanza db.r7g per creare cluster database Aurora. Per ulteriori informazioni, consulta l'argomento relativo alle classi di istanza database Aurora.	11 maggio 2023
Amazon Aurora supporta una nuova configurazione dell'architettura dei cluster database	Con Aurora I/O-Optimized, i prezzi sono calcolati sono calcolati solo in base all'utilizzo e all'archiviazione dei cluster database, senza costi aggiuntivi per le operazioni di I/O di lettura e scrittura. Per ulteriori informazioni, consultare Configurazioni dell'architettura per i cluster database Amazon Aurora.	11 maggio 2023

Amazon Aurora supporta Aurora Serverless v2 in aggiunta Regioni AWS	Ora puoi creare cluster Aurora Serverless v2 DB nei seguenti paesi Regioni AWS: Asia Pacifico (Hyderabad), Europa (Spagna) Europa (Zurigo) e Medio Oriente (Emirati Arabi Uniti). Per ulteriori informazioni su Aurora Serverless v2, consultare Utilizzo di Aurora Serverless v2.	4 maggio 2023
Aurora Serverless v1 supporta la conversione in allocato	Puoi convertire un cluster di database Aurora Serverless v1 direttamente in un cluster di database allocato. Per ulteriori informazioni, consulta Conversione di un cluster di database Aurora Serverless v1 in allocato.	27 aprile 2023
Aurora Serverless v1 supporta Amazon Aurora PostgreSQL versione 13	Ora puoi creare cluster di database Aurora Serverless v1 che eseguono Aurora PostgreSQL versione 13. Per ulteriori informazioni, consulta Aurora Serverless v1.	27 aprile 2023

Supporto per Amazon Aurora AWS Secrets Manager nelle regioni della Cina	Amazon Aurora supporta Secrets Manager nelle regioni Cina (Pechino) e Cina (Ningxia). Per ulteriori informazioni, consulta Password management with Amazon Aurora and AWS Secrets Manager (Gestione delle password per Amazon Aurora e AWS Secrets Manager).	20 aprile 2023
Amazon Aurora supporta la pubblicazione di eventi contenenti tag per gli abbonati degli argomenti	Le notifiche di eventi di Amazon Aurora inviate ad Amazon Simple Notification Service (Amazon SNS) o EventBridge Amazon ora contengono tag di evento nel corpo del messaggio. Questi tag forniscono i dati sulle risorse interessati dall'evento del servizio. Per ulteriori informazioni, consulta Tag e attributi delle notifiche eventi di Amazon RDS.	17 aprile 2023

Aggiornamento alle autorizzazioni del ruolo collegato ai servizi di IAM	Le `AmazonRDSReadOnlyAccess` policy `AmazonRDSFullAccess` and ora concedono autorizzazioni aggiuntive per consentire la visualizzazione dei risultati di Amazon DevOps Guru nella console RDS. Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	30 marzo 2023
Amazon Aurora supporta i database globali nella Regione Asia Pacifico (Melbourne)	Ora puoi creare i database globali Aurora nella Regione Asia Pacifico (Melbourne). Per informazioni sui database globali Aurora, consulta Utilizzo dei database globali Amazon Aurora.	22 marzo 2023
Aggiornamento delle autorizzazioni delle policy AWS gestite	Le `AmazonRDSReadOnlyAccess` politiche `AmazonRDSFullAccess` and ora concedono autorizzazioni aggiuntive ad Amazon. CloudWatch Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	16 marzo 2023
RDS Proxy è disponibile nelle regioni della Cina	RDS Proxy è ora disponibile nelle regioni Cina (Pechino) e Cina (Ningxia). Per ulteriori informazioni sul proxy RDS, consulta Utilizzo di Server proxy per Amazon RDS.	15 marzo 2023

Supporto Amazon Aurora per Aurora Serverless v2 nelle regioni della Cina	Aurora Serverless v2 è ora disponibile nelle regioni Cina (Pechino) e Cina (Ningxia) . Per ulteriori informazioni, consulta Aurora Serverless v2.	15 marzo 2023
RDS Proxy è disponibile nella regione Asia Pacific (Giacarta)	RDS Proxy è ora disponibile nella regione Asia Pacific (Giacarta). Per ulteriori informazioni sul proxy RDS, consulta Utilizzo di Server proxy per Amazon RDS.	8 marzo 2023
Amazon Aurora MySQL supporta l'autenticazione Kerberos	Puoi ora utilizzare l'autenticazione Kerberos per autenticare gli utenti quando si connettono a cluster di database Aurora MySQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos per Aurora MySQL.	8 marzo 2023
Amazon Aurora supporta database globali in aggiunta Regioni AWS	Puoi creare i database globali nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Hong Kong), Asia Pacifico (Hyderabad), Asia Pacifico (Giacarta), Europa (Milano), Europa (Spagna), Europa (Zurigo), Medio Oriente (Bahrein) e Medio Oriente (Emirati Arabi Uniti). Per informazioni sui database globali Aurora, consulta Utilizzo dei database globali Amazon Aurora.	6 marzo 2023

Amazon Aurora supporta la copia di snapshot di cluster DB in aggiunta Regioni AWS	Puoi creare gli snapshot di cluster di database nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Hong Kong), Asia Pacifico (Hyderabad), Asia Pacifico (Giacarta), Asia Pacifico (Melbourne), Europa (Milano), Europa (Spagna), Europa (Zurigo), Medio Oriente (Bahrein) e Medio Oriente (Emirati Arabi Uniti). Per informazioni sulla copia degli snapshot di cluster di database, consulta Copia di snapshot di cluster di database.	6 marzo 2023
Amazon DevOps Guru for RDS supporta approfondimenti proattivi	Amazon DevOps Guru for RDS pubblica approfondimenti proattivi con consigli per aiutarti a risolvere i problemi nei database Aurora prima che si verifichino. Per ulteriori informazioni, consulta Come funziona Guru for RDS. DevOps	28 febbraio 2023
Amazon Aurora MySQL versione 1 è deprecato	Aurora MySQL versione 1 (compatibile con MySQL 5.6) è stato deprecato. Per ulteriori informazioni, consulta Per quanto tempo le versioni principali di Amazon Aurora rimangono disponibili.	28 febbraio 2023

Aurora Serverless v1 supporta l'impostazione della finestra di manutenzione del cluster di database	È ora possibile impostare la finestra di manutenzione per i cluster di database Aurora Serverless v1. Per ulteriori informazioni, consulta Impostazione della finestra di manutenzione preferita del cluster di database.	27 febbraio 2023
Amazon Aurora supporta i flussi di attività di database nelle regioni Asia Pacifico (Hyderabad), Europa (Spagna) e Medio Oriente (Emirati Arabi Uniti).	Per maggiori informazioni, consulta Flussi di dati del database.	27 gennaio 2023
Amazon Aurora è disponibile nella Regione Asia Pacifico (Melbourne)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Melbourne). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	23 gennaio 2023
Specifica dell'autorità di certificazione (CA) durante la creazione di un cluster database	È ora possibile specificare quale CA utilizzare per il certificato del server del cluster database durante la creazione del cluster database. Per ulteriori informazioni, consulta Certificate authorities (Autorità di certificazione).	5 gennaio 2023

Supporto Aurora MySQL 3.* per il backtracking	Le versioni Aurora MySQL 3.* offrono ora un modo rapido per correggere gli errori degli utenti, come ad esempio il rilascio della tabella sbagliata o l'eliminazione della riga sbagliata. Backtrack permette di riportare il database a un punto nel tempo precedente senza la necessità di ripristinarlo da un backup, in pochi secondi anche per database di grandi dimensioni. Per ulteriori informazioni, consulta Backtrack di un cluster database Aurora.	4 gennaio 2023
Usa Amazon RDS Blue/Green Deployments (disponibile in aggiunta) Regioni AWS	La funzionalità implementazione blu/verde è ora disponibile nelle regioni Cina (Pechino) e Cina (Ningxia). Per ulteriori informazioni, consulta Utilizzo delle implementazioni blu/verde Amazon RDS per gli aggiornamenti del database.	22 dicembre 2022
Aggiornamento alle autorizzazioni del ruolo collegato ai servizi di IAM	La RDSService RolePolicy politica di Amazon ora concede autorizzazioni aggiuntive a. AWS Secrets Manager Per ulteriori informazioni, consulta gli aggiornamenti di Amazon RDS alle policy AWS gestite.	22 dicembre 2022

Amazon Aurora si integra con la gestione delle password AWS Secrets Manager	Aurora può gestire la password dell'utente master per un cluster database in Secrets Manager. Per ulteriori informazioni, consulta Password management with Amazon Aurora and AWS Secrets Manager (Gestione delle password per Amazon Aurora e AWS Secrets Manager).	22 dicembre 2022
Amazon Aurora supporta Aurora Serverless v2 in aggiunta Regioni AWS	Aurora Serverless v2 è ora disponibile nelle regioni Africa (Città del Capo) e UE (Milano). Per ulteriori informazioni, consulta Aurora Serverless v2.	21 dicembre 2022
Aurora PostgreSQL supporta Server proxy per RDS con PostgreSQL 14	Ora è possibile creare un proxy RDS con un cluster database Aurora PostgreSQL 14. Per ulteriori informazioni sul proxy RDS, consulta Utilizzo di Server proxy per Amazon RDS.	13 dicembre 2022
Amazon Aurora ti avvisa delle anomalie recenti rilevate da Amazon Guru DevOps	La pagina dei dettagli del database della console avvisa delle anomalie correnti e delle anomalie delle ultime 24 ore. Per ulteriori informazioni, consulta Come funziona DevOps Guru for RDS.	13 dicembre 2022

Server proxy per Amazon RDS supporta i database globali	Ora puoi utilizzare Server proxy per RDS con i database globali Aurora. Per ulteriori informazioni, consulta Using RDS Proxy with Aurora global databases (Utilizzo di RDS Proxy con i database globali Aurora).	7 dicembre 2022
I cluster database Aurora PostgreSQL supportano Trusted Language Extensions per PostgreSQL	Trusted Language Extensions per PostgreSQL è un kit di sviluppo open source che ti consente di creare estensioni di PostgreSQL ad alte prestazioni ed eseguirle in modo sicuro sul tuo cluster database Aurora PostgreSQL. Per ulteriori informazioni, consulta Working with Trusted Language Extensions for PostgreSQL (Utilizzo di Trusted Language Extensions per PostgreSQL).	30 novembre 2022

Amazon GuardDuty RDS Protection monitora le minacce di accesso	Quando attivi la protezione GuardDuty RDS, GuardDuty consuma gli eventi di accesso RDS dai database Aurora, monitora questi eventi e li profila per potenziali minacce interne o attori esterni. Quando GuardDuty RDS Protection rileva una potenzial e minaccia, GuardDuty genera una nuova scoperta con dettagli sul database potenzial mente compromesso. Per ulteriori informazioni, consulta Monitoraggio delle minacce con GuardDuty RDS Protectio n.	30 novembre 2022
Uso delle implementazioni blu/ verde Amazon RDS per gli aggiornamenti del database	È possibile apportare modifiche a un cluster database in un ambiente di gestione temporanea e testarle senza influire sul cluster database di produzion e. Quando sei pronto, puoi promuovere l'ambiente di gestione temporane a nel nuovo ambiente di produzione, con tempi di inattività minimi. Per ulteriori informazioni, consulta Utilizzo delle implementazioni blu/ verde Amazon RDS per gli aggiornamenti del database.	27 novembre 2022

Amazon Aurora è disponibile nel regione Asia Pacifico (Hyderabad)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Hyderabad). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	22 novembre 2022
Amazon Aurora è disponibile nella regione Europa (Spagna)	Amazon Aurora è ora disponibile nella Regione Europa (Spagna). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	16 novembre 2022
Amazon Aurora è disponibile nella Regione Europa (Zurigo)	Amazon Aurora è ora disponibile nella regione Europa (Zurigo). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	9 novembre 2022
Amazon Aurora supporta l'esportazione di dati in Amazon S3 da cluster database	Ora puoi esportare i dati del cluster Aurora direttamente in S3, senza dover prima creare uno snapshot. Per ulteriori informazioni, consulta Exporting DB cluster data to Amazon S3 (Esportazione dei dati del cluster database in Amazon S3).	27 ottobre 2022

Amazon Aurora MySQL supporta esportazioni più veloci in Amazon S3	Ora puoi ottenere prestazioni fino a 10 volte più veloci per l'esportazione dei dati dello snapshot del cluster database in S3 per i cluster Aurora MySQL compatibili con MySQL 5.7 e 8.0. Per ulteriori informazioni, consulta Exporting DB cluster snapshot data to Amazon S3 (Esportazione dei dati dello snapshot del cluster database in Amazon S3).	20 ottobre 2022
Amazon Aurora supporta la configurazione automatica della connettività tra un cluster Aurora DB e un'istanza EC2	È possibile utilizzare il AWS Management Console per configurare la connettività tra un cluster Aurora DB esistente e un' EC2 istanza. Per ulteriori informazioni, vedere Connessione automatica di un' EC2 istanza e di un cluster Aurora DB.	14 ottobre 2022
AWS Driver JDBC per PostgreSQL disponibile a livello generale	Il driver AWS JDBC per PostgreSQL è un driver client progettato per Aurora PostgreSQL. Il driver AWS JDBC per PostgreSQL è ora disponibile a livello generale. Per ulteriori informazioni, consulta Connessione con il driver AWS JDBC per PostgreSQL.	6 ottobre 2022

Amazon Aurora supporta l'aggiornamento locale di Aurora MySQL con compatibilità MySQL 5.7	È possibile eseguire un aggiornamento locale per modificare un cluster Aurora MySQL compatibile con MySQL 5.7 esistente in un cluster Aurora MySQL compatibile con MySQL 8.0. Per ulteriori informazioni, consulta Aggiornamento da Aurora MySQL 2.x a 3.x.	26 settembre 2022
Approfondimenti sulle prestazioni mostra le prime 25 query SQL	Nel pannello di controllo di Approfondimenti sulle prestazioni, la scheda Top SQL (Prime istruzioni SQL) mostra le 25 query SQL che contribuiscono di più al carico del database. Per ulteriori informazioni, consulta Panoramica della scheda Prime istruzioni SQL.	13 settembre 2022
Aurora MySQL supporta una nuova classe di istanza database	Ora puoi utilizzare la classe di istanza database db.r6i per i cluster database Aurora MySQL. Per ulteriori informazioni, consulta le classi di istanza database.	13 settembre 2022
Amazon Aurora è disponibile nella Regione Medio Oriente (Emirati Arabi Uniti)	Amazon Aurora è ora disponibile nella Regione Medio Oriente (Emirati Arabi Uniti). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	30 agosto 2022

Amazon Aurora supporta la configurazione automatica della connettività con un'istanza EC2	Quando crei un cluster Aurora DB, puoi utilizzarlo AWS Management Console per configurare la connettività tra un'istanza Amazon Elastic Compute Cloud e il nuovo cluster DB. Per ulteriori informazioni, consulta Configurare la connettività di rete automatica con un' EC2 istanza.	22 agosto 2022
Amazon Aurora supporta la modalità dual-stack	Ora è possibile eseguire i cluster database in modalità dual-stack. In modalità dual-stack, le risorse possono comunicare con il cluster DB tramite o IPv4 entrambi IPv6. Per ulteriori informazioni, consulta Assegnazione di indirizzi IP con Amazon Aurora.	17 agosto 2022
Amazon Aurora supporta l'aggiornamento locale di Aurora Serverless v1 con compatibilità PostgreSQL	È possibile eseguire un aggiornamento locale per un cluster Aurora Serverless v1 compatibile con PostgreSQL 10 per modificare un cluster esistente in un cluster Aurora Serverless v1 compatibile con PostgreSQL 11. Per la procedura di aggiornamento locale, consulta Modifica di un cluster database Aurora Serverless v1.	8 agosto 2022

Performance Insights supporta la regione Asia Pacific (Giacarta)	In precedenza, non era possibile utilizzare Approfondimenti sulle prestazioni nella regione Asia Pacific (Giacarta). Questa limitazione è stata eliminata. Per ulteriori informazioni, consulta Regione AWS che supportano Approfondimenti sulle prestazioni	21 luglio 2022
Amazon Aurora supporta una nuova classe di istanza database	Ora è possibile creare cluster di database db.r6i per cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta le classi di istanza database.	14 luglio 2022
RDS Performance Insights supporta periodi di conservazione aggiuntivi	In precedenza, Approfondimenti sulle prestazioni offriva solo due periodi di conservazione: 7 giorni (impostazione predefinita) o 2 anni (731 giorni). Ora, se devi mantenere i dati sulle prestazioni per più di 7 giorni, puoi specificare da 1 a 24 mesi. Per ulteriori informazioni, consulta Prezzi e conservazione dei dati per Approfondimenti sulle prestazioni.	1 luglio 2022

Amazon Aurora supporta l'aggiornamento locale di Aurora Serverless v1 con compatibilità MySQL	È possibile eseguire un aggiornamento in loco per un cluster Aurora Serverless v1 compatibile con MySQL 5.6 per modificare un cluster esistente in un cluster Aurora Serverless v1 compatibile con MySQL 5.7. Per la procedura di aggiornamento locale, consulta Modifica di un cluster database Aurora Serverless v1.	16 giugno 2022
Aurora supporta l'attivazione di Amazon DevOps Guru nella console RDS	Puoi attivare la copertura DevOps Guru per i tuoi database Aurora direttamente dalla console RDS. Per ulteriori informazioni, consulta Configurazione di DevOps Guru for RDS.	9 giugno 2022
Amazon Aurora supporta la pubblicazione di eventi su argomenti Amazon SNS crittografati	Amazon Aurora può ora pubblicare eventi in Servizio di notifica semplice Amazon (Amazon SNS) con crittografia lato server (SSE) abilitata, per una protezione aggiuntiva degli eventi che contengono dati sensibili. Per ulteriori informazioni, consulta Sottoscrizione alle notifiche eventi di Amazon RDS.	1 giugno 2022

Amazon RDS pubblica i parametri di utilizzo su Amazon CloudWatch	Il `AWS/Usage` namespace in Amazon CloudWatch include parametri di utilizzo a livello di account per le quote dei servizi Amazon RDS. Per ulteriori informazioni, consulta i parametri di CloudWatch utilizzo di Amazon per Amazon Aurora.	28 aprile 2022
Set di risultati dell'API dati in formato JSON	Un parametro opzionale all'interno dellla funzione `ExecuteStatement` fa sì che il set di risultati della query venga restituito come stringa in formato JSON. Trasformare il set di risultati JSON in una struttura dati nel linguaggio della tua applicazione è semplice e pratico. Per ulteriori informazioni, consulta Elaborazione dei risultati delle query in formato JSON.	27 aprile 2022
Amazon Aurora Serverless v2 è ora disponibile a livello generale	Amazon Aurora Serverless v2 è ora disponibile per tutti gli utenti. Per ulteriori informazioni, consulta Utilizzo di Aurora Serverless v2.	21 aprile 2022

Aurora MySQL supporta suite di cifratura configurabili	Con Aurora MySQL, è possibile utilizzare le suite di cifratura per controllare la crittografia di connessione accettata dal server di database. Per ulteriori informazioni, consulta Configurazione delle suite di cifratura per le connessioni ai cluster database Aurora MySQL.	15 aprile 2022
Aurora PostgreSQL supporta RDS Proxy con PostgreSQL 13	Ora è possibile creare un proxy RDS con un cluster database Aurora PostgreSQL 13. Per ulteriori informazioni sul proxy RDS, consulta Amazon RDS Proxy.	4 aprile 2022
Note di rilascio di Aurora PostgreSQL	Ora è disponibile una guida separata per le note di rilascio di Amazon Aurora PostgreSQL. Per ulteriori informazioni, consultare Note di rilascio di Aurora PostgreSQL.	22 marzo 2022
Note di rilascio di Aurora MySQL	Ora è disponibile una guida separata per le note di rilascio di Amazon Aurora MySQL. Per ulteriori informazioni, consultare Note di rilascio di Aurora MySQL.	22 marzo 2022

Aurora PostgreSQL supporta più aggiornamenti delle versioni principali	Ora è possibile eseguire aggiornamenti di versione dei cluster di database Aurora PostgreSQL su più versioni principali. Per ulteriori informazioni, consultare Come eseguire l'aggiornamento a una versione principale.	4 marzo 2022
Aurora PostgreSQL supporta suite di cifratura configurabili	Con Aurora PostgreSQL versioni 11.8 e successive, ora è possibile utilizzare suite di cifratura configurabili per controllare la crittografia di connessione accettata dal server di database. Per ulteriori informazioni sull'utilizzo di suite di cifratura configurabili con Aurora PostgreSQL, consulta Configurazione di suite di cifratura per connessioni ai cluster di database Aurora PostgreSQL.	4 marzo 2022
AWS Driver JDBC per MySQL generalmente disponibile	Il driver AWS JDBC per MySQL è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Il driver AWS JDBC per MySQL è ora disponibile a livello generale. Per maggiori informazioni, consulta Connessione con il driver JDBC per MySQL di Amazon Web Services.	2 marzo 2022

Aurora PostgreSQL 13.5 supporta Babelfish per Aurora PostgreSQL 1.1.0	Aurora PostgreSQL 13.5 supporta Babelfish 1.1.0. Per un elenco delle nuove caratteristiche, consulta13.5. Per un elenco delle caratteri stiche supportate in ogni versione di Babelfish, consulta Funzionalità supportate in Babelfish per versione. Per ulteriori informazioni, consulta Lavorare con Babelfish per Aurora PostgreSQL.	28 febbraio 2022
Amazon Aurora supporta i flussi di attività di database nella regione Asia Pacifico (Jakarta)	Per ulteriori informazioni, consulta Supporto Regioni AWS per i flussi di attività del database.	16 febbraio 2022
Performance Insights supporta nuove operazioni API	Performance Insights ora supporta le seguenti operazioni API: `GetResourceMetadata` , `ListAvailableResourceDimensions` e `ListAvailableResourceMetrics` . Per ulteriori informazioni, consulta Recupero dei parametri con l'API Performance Insights in questo manuale e la Documentazione di riferimento dell'API di Amazon RDS Performance Insights.	12 gennaio 2022

Amazon RDS Proxy supporta gli eventi	RDS Proxy ora genera eventi a cui puoi abbonarti e visualizzarli in CloudWatch Eventi o configurare per l'invio ad Amazon EventBridge. Per maggiori informazioni, consulta Utilizzo degli eventi RDS Proxy.	11 gennaio 2022
Proxy RDS disponibile in aggiunta Regioni AWS	RDS Proxy è ora disponibile nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Hong Kong), Asia Pacifico (Osaka), Europa (Milano), Europa (Parigi), Europa (Stoccolma), Medio Oriente (Bahrein) e Sud America (San Paolo). Per ulteriori informazioni sul proxy RDS, consulta Amazon RDS Proxy.	5 gennaio 2022
Amazon Aurora è disponibile nella Regione Asia Pacific (Giacarta)	Amazon Aurora è ora disponibile nella Regione Asia Pacific (Giacarta). Per ulteriori informazioni, consultare Regioni e zone di disponibilità.	13 dicembre 2021

DevOpsGuru for Amazon RDS fornisce informazioni dettagliate e consigli per Amazon Aurora	DevOpsGuru for RDS utilizza Performance Insights per i dati relativi alle prestazioni. Utilizzando questi dati, il servizio analizza le prestazioni delle istanze database Amazon Aurora e può aiutare a risolvere i problemi di prestazioni. Per ulteriori informazioni, consulta Analyzing performance anomalies with DevOps Guru for RDS in questa guida e vedi Overview of DevOps Guru for RDS nella Amazon Guru User Guide. DevOps	1° dicembre 2021
Aurora PostgreSQL supporta RDS Proxy con PostgreSQL 12	Ora è possibile creare un proxy RDS con un cluster di database Aurora PostgreSQL 12. Per ulteriori informazioni sul proxy RDS, consulta Amazon RDS Proxy.	22 novembre 2021
Aurora supporta le classi di istanze AWS Graviton2 per Database Activity Streams	È possibile utilizzare flussi di attività del database con la classe di istanza db.r6g per Aurora MySQL e Aurora PostgreSQL. Per ulteriori informazioni, consultare Motori di database supportati per tutte le classi di istanza database disponibili.	3 novembre 2021

Supporto Amazon Aurora per più account AWS KMS keys	È possibile utilizzare una chiave KMS di un'altra chiave Account AWS per la crittografia durante l'esportazione di snapshot DB su Amazon S3. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot DB su Simple Storage Service (Amazon S3).	3 novembre 2021
Amazon Aurora supporta Babelfish per Aurora PostgreSQL	Babelfish per Aurora PostgreSQL estende l'Edizione compatibile con PostgreSQL database Amazon Aurora con la possibilità di accettare connessioni al database dai client Microsoft SQL Server. Per ulteriori informazioni, consulta Lavorare con Babelfish per Aurora PostgreSQL.	28 ottobre 2021
Aurora Serverless v1 può richiedere SSL per le connessioni	I parametri del cluster Aurora `force_ssl` per PostgreSQL e `require_secure_transport` per MySQL ora sono supportati per Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo di TLS/SSL con Aurora Serverless v1.	26 ottobre 2021

Amazon Aurora supporta Performance Insights in aggiunta Regioni AWS	Approfondimenti sulle prestazioni è disponibile nelle regioni Medio Oriente (Bahrein), Africa (Città del Capo), Europa (Milano) e Asia Pacifico (Osaka). Per ulteriori informazioni, consulta Supporto delle regioni Regione AWS per Approfondimenti sulle prestazioni.	5 ottobre 2021
Timeout di scalabilità automatica configurabile per Aurora Serverless v1	È possibile scegliere per quanto tempo Aurora Serverless v1 deve attendere per trovare un punto di scalabilità automatica. Se durante tale periodo non viene rilevato alcun punto di scalabilità automatica, Aurora Serverless v1 annulla l'evento di ridimensionamento o forza la modifica della capacità, a seconda dell'azione di timeout selezionata. Per ulteriori informazioni, consultare Scalabilità automatica per Aurora Serverless v1.	10 settembre 2021

Aurora supporta le classi di istanza X2g e T4g	Sia Aurora MySQL che Aurora PostgreSQL possono ora utilizzare le classi di istanza X2g e T4g. Le classi di istanza che è possibile utilizzare dipendono dalla versione di Aurora MySQL o Aurora PostgreSQL. Per informazioni sui tipi di istanza supportati, consultare classi di istanza database.	10 settembre 2021
Amazon RDS supporta RDS Proxy in un VPC condiviso	Ora è possibile creare un proxy RDS in un virtual private cloud (VPC) condiviso. Per maggiori informazioni su RDS Proxy, consulta la sezione "Gestione delle connessioni con Amazon RDS Proxy" nella Guida per l'utente di Amazon RDS o nella Guida per l'utente di Aurora.	6 agosto 2021
Pagina delle policy della versione Aurora	La Guida per l'utente di Amazon Aurora ora include una sezione con informazioni generali sulle versioni di Aurora e sulle policy associate. Per informazioni dettagliate, consulta Versioni di Amazon Aurora.	14 luglio 2021
Escludi gli eventi Data API da un percorso AWS CloudTrail	Puoi escludere gli eventi Data API da un CloudTrail trail. Per ulteriori informazioni, consulta Escludere gli eventi Data API da un AWS CloudTrail trail.	2 luglio 2021

L'edizione compatibile con PostgreSQL di Amazon Aurora supporta estensioni aggiuntive	Le estensioni appena supportate includono pg_bigm, pg_cron, pg_partman e pg_proctab. Per ulteriori informazioni, consulta Versioni delle estensioni per l'edizione compatibile con PostgreSQL di Amazon Aurora.	17 giugno 2021
Clonazione per i cluster Aurora Serverless	È ora possibile creare cluster clonati che siano Aurora Serverless. Per ulteriori informazioni sulla clonazione, consulta Clonazione di un volume per un cluster database Aurora.	16 giugno 2021
Disponibilità dei database globali di Aurora nelle regioni Cina (Pechino) e Cina (Ningxia)	Ora è possibile creare i database globali Aurora nelle regioni Cina (Pechino) e Cina (Ningxia). Per informazioni sui database globali Aurora, consultare Lavorare con database globali Amazon Aurora.	19 maggio 2021
Supporto FIPS 140-2 per API dati	Data API supporta la Federal Information Processing Standard Publication 140-2 (FIPS 140-2) per connessioni SSL/TLS. Per ulteriori informazioni, consulta Disponibilità dell'API dati.	14 maggio 2021

AWS Driver JDBC per PostgreSQL (anteprima)	Il driver AWS JDBC per PostgreSQL, ora disponibile in anteprima, è un driver client progettato per l'elevata disponibilità di Aurora PostgreSQL. Per ulteriori informazioni, consulta Connessione con il driver JDBC di AWS per PostgreSQL (anteprima).	27 aprile 2021
L'API Data è disponibile in aggiunta Regioni AWS	L'API dati è ora disponibile nelle regioni Asia Pacifico (Seoul) e Canada (Centrale). Per ulteriori informazioni, consulta Disponibilità della Data API.	9 aprile 2021
Amazon Aurora supporta le classi di istanza database Graviton2	Ora è possibile utilizzare le classi di istanza database Graviton2 db.r6g.x per creare cluster di database che eseguono MySQL o PostgreSQL. Per ulteriori informazioni, consulta Tipi di classe di istanza database.	12 marzo 2021

Miglioramenti degli endpoint RDS Proxy	È possibile creare endpoint aggiuntivi associati a ciascun proxy RDS. La creazione di un endpoint in un VPC diverso consente l'accesso tra VPC per il proxy. I proxy per i cluster Aurora MySQL possono avere anche endpoint di sola lettura. Questi endpoint di lettura si connettono alle istanze database del lettore nei cluster e possono migliorare la scalabilità di lettura e la disponibilità per le applicazioni che richiedono un uso intensivo di query. Per maggiori informazioni su RDS Proxy, consulta la sezione "Gestione delle connessioni con Amazon RDS Proxy" nella Guida per l'utente di Amazon RDS o nella Guida per l'utente di Aurora.	8 marzo 2021
Amazon Aurora è disponibile nella Regione Asia Pacifico (Osaka-Locale)	Amazon Aurora è ora disponibile nella Regione Asia Pacifico (Osaka-Locale). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	1 marzo 2021

Aurora PostgreSQL supporta l'abilitazione dell'autenticazione IAM e Kerberos sullo stesso cluster di database	Aurora PostgreSQL ora supporta l'abilitazione sia dell'autenticazione IAM che dell'autenticazione Kerberos sullo stesso cluster di database. Per ulteriori informazioni, consulta Autenticazione del database con Amazon Aurora.	24 febbraio 2021
Il database globale di Aurora ora supporta il failover pianificato gestito	Il database globale di Aurora ora supporta il failover pianificato gestito, consentendo di modificare più facilmente la regione AWS principale del database globale Aurora. È possibile utilizzare il failover pianificato gestito solo con database globali Aurora integri. Per ulteriori informazioni, consulta Ripristino di emergenza e database globali Amazon Aurora. Per informazioni di riferimento, consulta `FailoverGlobalCluster` in Amazon RDS API Reference.	11 febbraio 2021

L'Data API per Aurora Serverless ora supporta più tipi di dati	L'API dati per Aurora Serverless ora ti permette di utilizzare i tipi di dati `UUID` e `JSON` come input per il database. Inoltre, con l'API dati per Aurora Serverless è possibile avere un valore di tipo `LONG` restituito dal database come valore `STRING`. Per ulteriori informazioni, consulta Chiamata della Data API. Per informazioni di riferimento sui tipi di dati supportati, consulta `SqlParameter` in Documentazione di riferimento delle API del servizio dati Amazon RDS.	2 febbraio 2021
Aurora PostgreSQL supporta gli aggiornamenti delle versioni principali a PostgreSQL 12	Con Aurora PostgreSQL, adesso puoi aggiornare il motore del database alla versione principale 12. Per ulteriori informazioni, consulta Aggiornamento del motore del database PostgreSQL per Aurora PostgreSQL.	28 gennaio 2021

Aurora MySQL supporta l'aggiornamento in loco	È possibile aggiornare il cluster Aurora MySQL 1.x a Aurora MySQL 2.x, preservando le istanze database, gli endpoint e così via del cluster originale. Questa tecnica di aggiornamento in loco evita l'inconveniente di configurare un cluster completamente nuovo ripristinando uno snapshot. Evita inoltre il sovraccarico dato dalla copia di tutti i dati della tabella in un nuovo cluster. Per ulteriori informazioni, consulta Aggiornamento della versione principale di un cluster database Aurora MySQL da 1.x a 2.x.	11 gennaio 2021
AWS Driver JDBC per MySQL (anteprima)	Il driver AWS JDBC per MySQL, ora disponibile in anteprima, è un driver client progettato per l'elevata disponibilità di Aurora MySQL. Per maggiori informazioni, consulta Connessione con il driver JDBC di Amazon Web Services per MySQL (anteprima).	7 gennaio 2021

Aurora supporta i flussi di attività del database su cluster secondari di un database globale	È possibile avviare un flusso di attività del database in un cluster primario o secondari o di Aurora PostgreSQL o Aurora MySQL. Per le versioni del motore supportat e, consulta Limitazioni dei database globali Aurora.	22 dicembre 2020
Cluster multi-master con 4 istanze database	Il numero massimo di istanze database in un cluster Aurora MySQL multi-master è ora quattro. In precedenz a, il massimo era di due istanze database. Per ulteriori informazioni, consulta Utilizzo di cluster multi-master Aurora.	17 dicembre 2020
Aurora PostgreSQL supporta funzioni AWS Lambda	Ora puoi richiamare la AWS Lambda funzione per i tuoi cluster Aurora PostgreSQ L DB. Per ulteriori informazi oni, consulta Richiamo di una funzione Lambda da un cluster database Aurora PostgreSQL.	11 dicembre 2020
Amazon Aurora supporta le classi di istanza database Graviton2 in anteprima	Ora è possibile utilizzare le classi di istanza database Graviton2 db.r6g.x in anteprima per creare cluster di database che eseguono MySQL o PostgreSQL. Per ulteriori informazioni, consulta la pagina relativa ai tipi di classe di istanza database.	11 dicembre 2020

Amazon Aurora Serverless v2 è ora disponibile in anteprima.	Amazon Aurora Serverless v2 è disponibile in anteprima. Per utilizzare Amazon Aurora Serverless v2, è necessari o richiedere l'accesso. Per ulteriori informazioni, consulta la pagina Aurora Serverless v2.	1 dicembre 2020

Aurora PostgreSQL è ora disponibile per Aurora Serverless in più Regioni AWS.	Aurora PostgreSQL è ora disponibile per Aurora Serverless in più Regioni AWS. Ora puoi scegliere di eseguire Aurora PostgreSQL Serverless v1 la stessa offerta. Regioni AWS Aurora MySQL Serverless v1 Il Aurora Serverless supporto include inoltre Stati Uniti occidentali (California settentrionale), Asia Pacifico (Singapore) Asia Pacifico (Sydney) Asia Pacifico (Seoul) Asia Pacifico (Mumbai) Canada (Europa centrale) (Londra) ed Europa (Parigi). Regioni AWS Per un elenco di tutte le regioni e i motori Aurora DB supportati perAurora Serverless, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v1. Anche l'API dati di Amazon RDS per Aurora Serverless è ora disponibile nelle Regioni AWS sopra menzionate. Per un elenco di tutte le regioni con supporto per l'API Data perAurora Serverless, consulta Data API with Aurora MySQL Serverless v1	24 novembre 2020

Amazon RDS Performance Insights introduce nuove dimensioni	È possibile raggruppare il carico del database in base ai gruppi di dimensioni per database, applicazione (PostgreSQL) e tipo di sessione (PostgreSQL). Amazon RDS supporta anche le dimensioni db.name, db.application.name (PostgreSQL) e db.session_type.name (PostgreSQL). Per ulteriori informazioni, consulta Tabelle dal carico superiore.	24 novembre 2020
Aurora Serverless supporta Aurora PostgreSQL versione 10.12	Aurora PostgreSQL per Aurora Serverless è stato aggiornato ad Aurora PostgreSQL versione 10.12 in tutte le regioni AWS in cui è supportato Aurora PostgreSQL per Aurora Serverless. Per ulteriori informazioni, consulta Regioni supportate e motori Aurora DB per Aurora Serverless v1.	4 novembre 2020
La Data API ora supporta l'autorizzazione basata su tag	L'API dati supporta l'autorizzazione basata su tag. Se le risorse cluster RDS sono state etichettate con tag, è possibile utilizzare questi tag nelle istruzioni delle policy per controllare l'accesso tramite La Data API. Per ulteriori informazioni, consulta Autorizzare l'accesso all'API dati.	27 ottobre 2020

Amazon Aurora estende il supporto per l'esportazione di istantanee su Simple Storage Service (Amazon S3)	È ora possibile esportare i dati di snapshot DB in Simple Storage Service (Amazon S3) in tutte le Regioni AWS commercia li. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot DB su Simple Storage Service (Amazon S3).	22 ottobre 2020
Il database globale Aurora supporta la clonazione	Ora è possibile creare cloni dei cluster di database primari e secondari dei database globali Aurora. È possibile farlo utilizzando AWS Management Console e scegliendo l'opzione di menu Crea clone. È inoltre possibile utilizzare AWS CLI ed eseguire il `restore-db-cluster-to-point-in-time` comando con l'`--restore-type copy-on-write` opzione. Utilizzando AWS Management Console o the AWS CLI, puoi anche clonare i cluster DB dai tuoi database globali Aurora tra più account. AWS Per ulteriori informazioni sulla clonazion e, consulta Clonazione di un volume di cluster di database Aurora.	19 ottobre 2020

Amazon Aurora supporta il ridimensionamento dinamico per il volume del cluster	A partire da Aurora MySQL 1.23 e 2.09 e da Aurora PostgreSQL 3.3.0 e Aurora PostgreSQL 2.6.0, Aurora riduce la dimensione del volume del cluster dopo la rimozione dei dati tramite operazioni quali `DROP TABLE`. Per sfruttare questo miglioramento, esegui l'aggiornamento a una delle versioni appropriate a seconda del motore del database utilizzato dal cluster. Per informazioni su questa funzionalità e su come controllare lo spazio di storage utilizzato e disponibile per un cluster Aurora, consultare Gestione delle prestazioni e del dimensionamento dei cluster di database Aurora.	13 ottobre 2020
Amazon Aurora supporta dimensioni di volume fino a 128 TiB	I volumi dei cluster Aurora nuovi ed esistenti possono ora crescere fino a raggiungere la dimensione massima di 128 tebibytes (TiB). Per ulteriori informazioni, consulta Modalità di crescita dello storage Aurora.	22 settembre 2020

Aurora PostgreSQL supporta le classi di istanza database db.r5 e db.t3 nella regione Cina (Ningxia)	È ora possibile creare cluster di database Aurora PostgreSQL nella regione Cina (Ningxia) che utilizza le classi di istanza database db.r5 e db.t3. Per ulteriori informazioni, consulta Classi di istanza database.	3 settembre 2020

Miglioramenti delle query parallele in Aurora	A partire da Aurora MySQL 2.09 e 1.23, è possibile sfruttare i miglioramenti apportati alla funzionalità di query parallela. La creazione di un cluster di query parallele non richiede più una modalità motore speciale. È ora possibile attivare e disattiva re la query parallela utilizzan do l'opzione di configurazione `aurora_parallel_qu ery` per qualsiasi cluster di cui è in esecuzione una versione Aurora MySQL compatibile. È possibile aggiornare un cluster esistente a una versione Aurora MySQL compatibile e utilizzare una query parallela, anziché creare un nuovo cluster e importare dati in esso. È possibile utilizzare Performance Insights per cluster di query parallele. È possibile arrestare e avviare cluster di query parallele. È possibile creare cluster di query Aurora parallele compatibili con MySQL 5.7. La query parallela funziona per le tabelle che utilizzano il formato di riga `DYNAMIC`. I cluster di query paralleli possono utilizzare l'autenti cazione AWS Identity and Access Management (IAM).	2 settembre 2020

	Le istanze database di lettura in cluster di query parallele possono sfruttare il livello di isolamento `READ COMMITTED` . È inoltre possibile creare cluster di query parallele in Regioni AWS aggiuntive. Per ulteriori informazioni sulla funzionalità di query parallela e su questi miglioramenti, consulta Parallel query for Aurora MySQL.	
Modifiche al parametro Aurora MySQL binlog_rows_query_log_events	È ora possibile modificare il valore del parametro Aurora MySQL di configurazione `binlog_rows_query_log_events` . In precedenza, questo parametro non era modificabile.	26 agosto 2020

Supporto per gli aggiornamenti automatici delle versioni secondarie per Aurora MySQL	Con Aurora MySQL, l'impostazione Enable auto minor version upgrade (Abilita aggiornamento automatico della versione secondaria) ora ha effetto quando la specifici per un cluster database Aurora MySQL. Quando abiliti l'aggiornamento automatico della versione secondaria, Aurora esegue automaticamente l'aggiornamento alle nuove versioni secondarie non appena vengono rilasciate. Gli aggiornamenti automatici vengono eseguiti durante la finestra di manutenzione del database. Per Aurora MySQL, questa funzione si applica solo ad Aurora MySQL versione 2, che è compatibile con MySQL 5.7. Inizialmente, la procedura di aggiornamento automatico porta i cluster di database Aurora MySQL alla versione 2.07.2. Per ulteriori informazioni sul funzionamento di questa caratteristica con Aurora MySQL, consulta Aggiornamenti e patch del database per Amazon Aurora MySQL.	3 agosto 2020

Aurora PostgreSQL supporta gli aggiornamenti delle versioni principali a PostgreSQL versione 11	Con Aurora PostgreSQL, puoi ora aggiornare il motore del database alla versione principale 11. Per ulteriori informazioni, consulta Aggiornamento del motore del database PostgreSQL per Aurora PostgreSQL.	28 luglio 2020
Amazon Aurora supporta AWS PrivateLink	Amazon Aurora ora supporta la creazione di endpoint Amazon VPC per le chiamate API Amazon RDS per mantenere il traffico tra le applicazioni e Aurora nella rete. AWS Per ulteriori informazioni consulta Amazon Aurora ed endpoint VPC di interfaccia (AWS PrivateLink).	9 luglio 2020
Proxy RDS disponibile a livello generale	Proxy RDS è ora disponibile a livello generale. È possibile utilizzare RDS Proxy con RDS for MySQL, Aurora MySQL, RDS for PostgreSQL e Aurora PostgreSQL per carichi di lavoro di produzione. Per maggiori informazioni su RDS Proxy, consulta la sezione "Gestione delle connessioni con Amazon RDS Proxy" nella Guida per l'utente di Amazon RDS o nella Guida per l'utente di Aurora.	30 giugno 2020

Inoltro di scrittura del database globale Aurora	È ora possibile abilitare la funzionalità di scrittura su cluster secondari in un database globale. Con l'inoltro di scrittura, inserisci istruzioni DML in un cluster secondario, Aurora inoltra la scrittura al cluster primario e i dati aggiornati vengono replicati in tutti i cluster secondari. Per ulteriori informazioni, consulta Inoltro di scrittura secondario Regioni AWS con un database globale Aurora.	18 giugno 2020
Aurora supporta l'integrazione con AWS Backup	È possibile AWS Backup utilizzarlo per gestire i backup dei cluster Aurora DB. Per ulteriori informazioni, consulta Panoramica di backup e ripristino di un cluster di database Aurora.	10 giugno 2020
Aurora PostgreSQL supporta le classi di istanza database db.t3.large.	Ora è possibile creare cluster di database Aurora PostgreSQL basati su classi di istanza database db.t3.large. Per ulteriori informazioni, consulta Classi di istanza database	5 giugno 2020

Il database globale di Aurora supporta PostgreSQL versione 11.7 e l'obiettivo del punto di ripristino (RPO, Recovery Point Objective) gestito	Puoi ora creare database globali Aurora per il motore del database PostgreSQL versione 11.7. È inoltre possibile gestire il ripristino di un database globale PostgreSQL da un errore utilizzando un obiettivo del punto di ripristino (RPO). Per ulteriori informazioni, vedere Ripristino di emergenza tra regioni per i database globali Aurora.	4 giugno 2020
Aurora MySQL supporta il monitoraggio del database con flussi di attività di database	Aurora MySQL ora include flussi di attività del database, che forniscono un flusso di near-real-time dati dell'attività del database nel database relazionale. Per maggiori informazioni, consulta Utilizzo dei flussi di dati di attività del database.	2 giugno 2020
L'editor di query è disponibile in aggiunta Regioni AWS	L'editor di query per Aurora Serverless è ora disponibile in versione aggiuntiva. Regioni AWS Per ulteriori informazioni, consulta Disponibilità dell'editor di query.	28 maggio 2020
L'API Data è disponibile in aggiunta Regioni AWS	L'API Data è ora disponibile in modalità aggiuntiva Regioni AWS. Per ulteriori informazioni, consulta Disponibilità della Data API.	28 maggio 2020

Proxy RDS disponibile nella regione Canada (Centrale)	È ora possibile utilizzare l'anteprima Proxy RDS in Canada (Centrale). Per ulteriori informazioni sul proxy RDS, consulta Gestione delle connessioni con Amazon RDS Proxy (Anteprima).	28 maggio 2020
Aurora Global Database e repliche di lettura tra regioni	Per Aurora Global Database, non è possibile creare una replica di lettura tra regioni Aurora MySQL dal cluster primario nella stessa regione del cluster secondario. Per ulteriori informazioni su Aurora Global Database e repliche di lettura tra regioni, consulta Utilizzo di Amazon Aurora Global Database e Repliche di database Amazon Aurora MySQL.	18 maggio 2020
RDS Proxy è disponibile in altre versioni Regioni AWS	È ora possibile utilizzar e l'anteprima proxy RDS nelle regioni Stati Uniti occidentali (California settentri onale), Europa (Londra), Europa (Francoforte), Asia Pacifico (Seoul), Asia Pacifico (Mumbai), Asia Pacifico (Singapore) e Asia Pacifico (Sydney). Per ulteriori informazioni sul proxy RDS, consulta Gestione delle connessioni con Amazon RDS Proxy (Anteprima).	13 maggio 2020

Aurora edizione compatibile con PostgreSQL supporta Microsoft Active Directory locale o autogestito	È ora possibile utilizzare una Active Directory locale o autogestita per l'autenticazione Kerberos di utenti quando si connettono ai cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos con Aurora PostgreSQL.	7 maggio 2020
Cluster multi-master Aurora MySQL disponibili in altre versioni Regioni AWS	È ora possibile creare cluster multi-master Aurora nelle regioni Asia Pacifico (Seoul), Asia Pacifico (Tokyo), Asia Pacifico (Mumbai) ed Europa (Francoforte). Per ulteriori informazioni sui cluster multi-master, consulta Utilizzo di cluster multi-master Aurora.	7 maggio 2020
Performance Insights supporta l'analisi delle statistiche di esecuzione di query Aurora MySQL	È ora possibile analizzare le statistiche delle query in esecuzione con Performance Insights per le istanze database Aurora MySQL. Per ulteriori informazioni, consulta Analisi delle statistiche delle query in esecuzione.	5 maggio 2020

Libreria client Java per l'API dati disponibile a livello generale	La libreria client Java per Data API è ora disponibile a livello generale. È possibile scaricare e utilizzare una libreria client Java per La Data API. Questa consente di mappare le classi sul lato client alle richieste e alle risposte della Data API. Per ulteriori informazioni, consulta Utilizzo dell'API dati per Aurora Serverless.	30 aprile 2020
Amazon Aurora è disponibile nella Regione Europa (Milano)	Amazon Aurora è ora disponibile nella Regione Europa (Milano). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	28 aprile 2020
Amazon Aurora è disponibile nella Regione Europa (Milano)	Amazon Aurora è ora disponibile nella Regione Europa (Milano). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	27 aprile 2020
Amazon Aurora è disponibile nella Regione Africa (Città del Capo)	Amazon Aurora è ora disponibile nella Regione Africa (Città del Capo). Per ulteriori informazioni, consulta Regioni e zone di disponibilità.	22 aprile 2020

Aurora PostgreSQL ora supporta le classi di istanza database db.r5.16xlarge e db.r5.8xlarge	È ora possibile creare cluster database Aurora PostgreSQL che eseguono PostgreSQL che utilizzano le classi di istanze database db.r5.16xlarge e db.r5.8xlarge. Per ulteriori informazioni, consulta Specifiche hardware per le classi di istanza database per Aurora.	8 aprile 2020
Proxy Amazon RDS per PostgreSQL	Amazon RDS Proxy è ora disponibile per PostgreSQL. È possibile utilizzare RDS Proxy per ridurre il sovraccarico di gestione delle connessioni sul cluster e anche la possibilità di errori di "troppe connessioni". RDS Proxy è attualmente in anteprima pubblica per PostgreSQL. Per ulteriori informazioni, consulta Gestione delle connessioni con il proxy Amazon RDS (Anteprima).	8 aprile 2020

I database globali Aurora ora supportano Aurora PostgreSQL	Puoi ora creare database globali Aurora per il motore del database PostgreSQL. Un database globale Aurora si estende su più pagine Regioni AWS, abilitando letture globali a bassa latenza e disaster recovery in caso di interruzioni a livello regionale. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Global Database.	10 marzo 2020
Supporto per gli aggiornamenti di versione principali per Aurora PostgreSQL	Con Aurora PostgreSQL, puoi ora aggiornare il motore del database a una versione principale. In questo modo, puoi passare a una versione principale più recente quando aggiorni le versioni del motore PostgreSQL selezionate. Per ulteriori informazioni, consulta Aggiornamento del motore del database PostgreSQL per Aurora PostgreSQL.	4 marzo 2020
Aurora PostgreSQL supporta l'autenticazione Kerberos	Puoi ora utilizzare l'autenticazione Kerberos per autenticare gli utenti quando si connettono a cluster di database Aurora PostgreSQL. Per ulteriori informazioni, consulta Utilizzo dell'autenticazione Kerberos con Aurora PostgreSQL.	28 febbraio 2020

L'API Data supporta AWS PrivateLink	L'API Data ora supporta la creazione di endpoint Amazon VPC per chiamate Data API per mantenere il traffico tra le applicazioni e l'API Data nella rete. AWS Per ulteriori informazioni, consulta Creazione di un endpoint Amazon VPC (AWS PrivateLink) per l'API dati.	6 febbraio 2020
Supporto Machine Learning di Aurora in Aurora PostgreSQL	L'estensione `aws_ml` Aurora PostgreSQL fornisce funzioni da utilizzare nelle query del database per chiamare Amazon Comprehend per l'analisi dei sentimenti e l'intelligenza artificiale per eseguire i tuoi modelli di machine learning. SageMaker Per ulteriori informazioni, consulta Utilizzo delle funzionalità di Machine Learning (ML) con Amazon Aurora.	5 febbraio 2020
Aurora PostgreSQL supporta l'esportazione di dati in Simple Storage Service (Amazon S3)	Puoi eseguire una query sui dati di un cluster di database Aurora PostgreSQL ed esportarli direttamente in file archiviati in un bucket Simple Storage Service (Amazon S3). Per ulteriori informazioni, consulta Esportazione dei dati da un cluster Aurora PostgreSQL DB in Simple Storage Service (Amazon S3).	5 febbraio 2020

Supporto per l'esportazione di dati dello snapshot DB in Simple Storage Service (Amazon S3)	Amazon Aurora supporta l'esportazione di dati snapshot DB in Simple Storage Service (Amazon S3) per MySQL e PostgreSQL. Per ulteriori informazioni, consulta Esportazione dei dati dello snapshot DB su Simple Storage Service (Amazon S3).	9 gennaio 2020
Note di rilascio di Aurora MySQL in Cronologia documenti	Questa sezione include ora le voci della cronologia per le note di rilascio di Aurora edizione compatibile con MySQL per le versioni rilasciate dopo il 31 agosto 2018. Per le note di rilascio complete per una versione specifica, scegliere il collegamento nella prima colonna della voce della cronologia.	13 dicembre 2019

Proxy Amazon RDS	Puoi ridurre il sovraccarico della gestione delle connessioni nel cluster e ridurre la possibilità di errori del tipo «troppe connessioni» utilizzando il proxy Amazon RDS. Puoi associare ogni proxy a un'istanza RDS di DB Servizi di dominio Active Directory o cluster di DB Aurora. Quindi utilizzi l'endpoint proxy nella stringa di connessione per l'applicazione. Il proxy Amazon RDS è attualmente in uno stato di anteprima pubblica. Supporta il motore del database Aurora MySQL. Per ulteriori informazioni, consulta Gestione delle connessioni con il proxy Amazon RDS (Anteprima).	3 dicembre 2019
La Data API per Aurora Serverless v1 supporta suggerimenti per la mappatura dei tipi di dati	È ora possibile utilizzare un suggerimento per indicare alLa Data API per Aurora Serverless v1 di inviare un valore `String` al database come un tipo diverso. Per ulteriori informazioni, consulta Invocazione della Data API.	26 novembre 2019

La Data API di Aurora Serverless v1 supporta una libreria client Java (Anteprima)	È possibile scaricare e utilizzare una libreria client Java per La Data API. Questa consente di mappare le classi sul lato client alle richieste e alle risposte della Data API. Per ulteriori informazioni, consulta Utilizzo dell'API dati per Aurora Serverless.	26 novembre 2019
Aurora PostgreSQL è idoneo per FedRAMP HIGH	Aurora PostgreSQL è idoneo per FedRAMP HIGH. Per i dettagli AWS e gli sforzi di conformità, consulta AWS la sezione Servizi rientranti nell'ambito del programma di conformità.	26 novembre 2019

Livello di isolamento READ COMMITTED abilitato per le repliche Amazon Aurora MySQL	È ora possibile abilitare il livello di isolamento `READ COMMITTED` sulle repliche Aurora MySQL. Questa operazione richiede l'abilitazione delle impostazioni di configurazione `aurora_read_replica_read_committed_isolation_enabled` a livello di sessione. L'uso del livello di isolamento `READ COMMITTED` per query a esecuzione prolungata sui cluster OLTP può aiutare a risolvere i problemi relativi alla lunghezza dell'elenco della cronologia. Prima di abilitare questa impostazione, accertarsi di comprendere la differenza del comportamento di isolamento sulle repliche Aurora si differenzia dalla normale implementazione MySQL di `READ COMMITTED`. Per ulteriori informazioni, consulta Livelli di isolamento di Aurora MySQL.	25 novembre 2019

Performance Insights supporta l'analisi delle statistiche di esecuzione di query Aurora PostgreSQL	È ora possibile analizzare le statistiche delle query in esecuzione con Performance Insights per le istanze database PostgreSQL per Aurora. Per ulteriori informazioni, consulta Analisi delle statistiche delle query in esecuzione.	25 novembre 2019
Ulteriori cluster in un database globale Aurora	È possibile aggiungere più regioni secondarie a un database globale Aurora. È possibile sfruttare le letture globali a bassa latenza e il ripristino di emergenza in un'area geografica più ampia. Per informazioni sui database globali Aurora, consultare Utilizzo di Amazon Aurora Global Database.	25 novembre 2019

Supporto Machine Learning di Aurora in Aurora MySQL	In Aurora MySQL 2.07 e versioni successive, puoi chiamare Amazon Comprehend per l'analisi del sentimento e l'intelligenza artificiale per un'ampia varietà di algoritmi di apprendimento automatico. SageMaker I risultati vengono utilizzati direttamente nelle applicazioni del database incorporando le chiamate nelle funzioni archiviate nelle query. Per ulteriori informazioni, consulta Utilizzo delle funzionalità di Machine Learning (ML) con Amazon Aurora.	25 novembre 2019
Aurora Global Database non richiede più l'impostazione della modalità motore	Non è più necessario `--engine-mode=global` specificare quando si crea un cluster destinato a far parte di un database globale Aurora. Tutti i cluster Aurora che soddisfano i requisiti di compatibilità sono idonei a far parte di un database globale. Ad esempio, il cluster deve al momento utilizzare Aurora MySQL versione 1 con la compatibilità MySQL 5.6. Per informazioni sui database globali Aurora, consultare Lavorare con database globali Amazon Aurora.	25 novembre 2019

Aurora Global Database è disponibile per Aurora MySQL versione 2	A partire da Aurora MySQL 2.07, è possibile creare un database globale Aurora con compatibilità MySQL 5.7. Non è necessario specificare la modalità motore `global` per i cluster primari e secondari . Con Aurora MySQL 2.07 o versioni successive è possibile aggiungere a Aurora Global Database cluster con provision ing. Per informazioni su Aurora Global Database, consultar e Utilizzo di Amazon Aurora Global Database.	25 novembre 2019
Ottimizzazione dei conflitti delle righe disponibile in Aurora MySQL senza modalità laboratorio	L'ottimizzazione dei conflitti delle righe è ora generalme nte disponibile per Aurora MySQL e non richiede l'attivaz ione della modalità laborator ioAurora. Questa caratteristica migliora significativamente il throughput per i carichi di lavoro in cui più transazioni si contendono le righe della stessa pagina. Il miglioram ento riguarda la modifica dell'algoritmo di rilascio del blocco utilizzato da Aurora MySQL.	19 novembre 2019

Gli hash join Aurora MySQL sono disponibili senza la modalità laboratorio	La caratteristica hash join è ora generalmente disponibile per Aurora MySQL e non richiede che la modalità laboratorio Aurora sia attivata. Questa funzionalità può migliorare le prestazioni delle query se devi eseguire il join di un'ingente quantità di dati tramite una query equijoin. Per ulteriori informazioni sull'utilizzo di questa caratteristica, consultare l'argomento relativo all'utilizzo degli hash join in Aurora MySQL.	19 novembre 2019
Supporto Aurora MySQL 2.* per ulteriori classi di istanza db.r5	I cluster Aurora MySQL ora supportano i tipi di istanza db.r5.8xlarge, db.r5.16xlarge e db.r5.24xlarge. Per ulteriori informazioni sui tipi di istanza per i cluster Aurora MySQL, consulta Scelta della classe di istanza database.	19 novembre 2019

Supporto Aurora MySQL 2.* per il backtracking	Le versioni Aurora MySQL 2.* offrono ora un modo rapido per correggere gli errori degli utenti, come ad esempio il rilascio della tabella sbagliata o l'eliminazione della riga sbagliata. Backtrack permette di riportare il database a un punto nel tempo precedente senza la necessità di ripristinarlo da un backup, in pochi secondi anche per database di grandi dimensioni. Per ulteriori informazioni, consulta Backtracking di un cluster database Aurora.	19 novembre 2019

Supporto dei tag di fatturazione per Aurora	Ora è possibile utilizzare i tag per tenere traccia dell'allocazione dei costi per risorse come cluster Aurora, istanze database all'interno dei cluster Aurora, operazioni I/O, backup, snapshot e così via. Puoi vedere i costi associati a ciascun tag utilizzando AWS Cost Explorer. Per ulteriori informazioni sull'utilizzo dei tag con Aurora, consulta Assegnazione di tag alle risorse Amazon RDS. Per informazioni generali sui tag e su come utilizzarli per l'analisi dei costi, consulta Utilizzo dei tag per l'allocazione dei costi e Tag per l'allocazione dei costi definiti dall'utente.	23 ottobre 2019
Data API per Aurora PostgreSQL	Aurora PostgreSQL ora supporta l'uso di Data API con cluster di database Amazon Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo dall'API dati per Aurora Serverless v1.	23 settembre 2019

Aurora PostgreSQL supporta il caricamento dei log del database nei log CloudWatch	Puoi configurare il tuo cluster Aurora PostgreSQL DB per pubblicare i dati di log in un gruppo di log in Amazon Logs. CloudWatch Con CloudWatch Logs, puoi eseguire analisi in tempo reale dei dati di log e utilizzarli CloudWatch per creare allarmi e visualizzare metriche. È possibile utilizzare CloudWatch Logs per archiviare i record di registro in un archivio altamente durevole. Per ulteriori informazioni, consulta Pubblicazione dei log di Aurora PostgreSQL su Amazon Logs. CloudWatch	9 agosto 2019
Cluster multi-master per Aurora MySQL	Puoi impostare cluster multi-master Aurora MySQL In questi cluster, ogni istanza database ha delle funzionalità di lettura-scrittura. Per ulteriori informazioni, consulta Utilizzo di cluster multi-master Aurora.	8 agosto 2019

Aurora PostgreSQL supporta Aurora Serverless v1	È ora possibile utilizzare Amazon Aurora Serverless v1 con Aurora PostgreSQL. Un cluster di database Aurora Serverless si avvia, si interrompe e scala automaticamente la capacità di calcolo in base alle esigenze dell'applicazione. Per ulteriori informazioni, consulta Utilizzo di Amazon Aurora Serverless v1.	9 luglio 2019
Clonazione tra più account per Aurora MySQL	Ora puoi clonare il volume del cluster per un cluster Aurora MySQL DB tra account. AWS Autorizzi la condivisione tramite AWS Resource Access Manager (AWS RAM). Il volume del cluster clonato utilizza un copy-on-write meccanismo che richiede solo spazio di archiviazione aggiuntivo per dati nuovi o modificati. Per ulteriori informazioni sulla clonazione per Aurora, consulta Clonazione dei database in un cluster database Aurora.	2 luglio 2019
Aurora PostgreSQL supporta le classi di istanza database db.t3	Ora è possibile creare cluster di database di Aurora PostgreSQL basati su classi di istanza database db.t3. Per ulteriori informazioni, consulta Classe di istanza database.	20 giugno 2019

Supporto per l'importazione dei dati da Simple Storage Service (Amazon S3) per Aurora PostgreSQL	Ora è possibile importare i dati da un file su Simple Storage Service (Amazon S3) in una tabella in un cluster di database di Aurora PostgreSQL. Per ulteriori informazioni, consulta Importazione dei dati da Simple Storage Service (Amazon S3) in un cluster di database Aurora PostgreSQL.	19 giugno 2019
Aurora PostgreSQL ora offre un rapido recupero dai failover con la gestione della cache del cluster	Aurora PostgreSQL offre ora la gestione della cache del cluster per garantire un rapido ripristino dell'istanza database primaria in caso di failover. Per ulteriori informazioni, consulta Ripristino rapido dopo il failover con la gestione della cache del cluster.	11 giugno 2019
La Data API per Aurora Serverless v1 è disponibile a livello generale	Puoi accedere ai cluster Aurora Serverless v1 con applicazioni basate sui servizi Web mediante La Data API. Per ulteriori informazioni, consultare la sezione relativa all'uso della Data API per Aurora Serverless v1	30 maggio 2019

Aurora PostgreSQL supporta il monitoraggio del database con flussi di attività di database	Aurora PostgreSQL ora include flussi di attività del database, che forniscono un flusso di near-real-time dati dell'attività del database nel database relazionale. Per maggiori informazioni, consultare la sezione relativa all'uso dei flussi di dati di database.	30 maggio 2019
Suggerimenti di Amazon Aurora	Amazon Aurora offre ora raccomandazioni automatiche per le risorse di Aurora. Per ulteriori informazioni, consulta la pagina relativa ai Consigli sull'utilizzo di Amazon Aurora.	22 maggio 2019
Supporto di Performance Insights per Aurora Global Database	Ora è possibile utilizzare Performance Insights con Aurora Global Database. Per informazioni su Performance Insights per Aurora, consultare Utilizzo di Performance Insights di Amazon RDS. Per informazioni sui database globali di Aurora, consultare Lavorare con Aurora Global Database.	13 maggio 2019

Disponibilità di Performance Insights per Aurora MySQL 5.7	Amazon RDS Performance Insights è ora disponibile per le versioni 2.x di Aurora MySQL, che sono compatibili con MySQL 5.7. Per ulteriori informazioni, consulta Utilizzo di Amazon RDS Performance Insights.	3 maggio 2019
Database globali Aurora disponibili in più Regioni AWS	Ora puoi creare database globali Aurora nella maggior parte dei paesi in cui Regioni AWS Aurora è disponibile. Per informazioni sui database globali Aurora, consulta Lavorare con database globali Amazon Aurora.	30 aprile 2019
Capacità minima di 1 per Aurora Serverless v1	L'impostazione della capacità minima che è possibile utilizzare per un cluster Aurora Serverless v1 è 1. Precedentemente, la capacità minima era 2. Per informazioni sulla specifica dei valori di capacità di Aurora Serverless, consultare Impostazione della capacità di un cluster di database di Aurora Serverless v1.	29 aprile 2019

Operazione di timeout di Aurora Serverless v1	Ora è possibile specifica re l'operazione da eseguire quando una modifica della capacità di Aurora Serverless v1 va in timeout. Per ulteriori informazioni, consultare Operazione di timeout per le modifiche della capacità.	29 aprile 2019
Fatturazione al secondo	Amazon RDS ora viene fatturato in incrementi di 1 secondo in tutte le istanze Regioni AWS ad eccezione di AWS GovCloud (Stati Uniti) per le istanze on-demand. Per ulteriori informazioni, consulta Fatturazione delle istanze database per Aurora.	25 aprile 2019
Condivisione di istantanee tra Aurora Serverless v1Regioni AWS	Con Aurora Serverless v1 le operazioni con gli snapshot sono sempre crittografate. Se si crittografa l'istantanea con la propria AWS KMS key, ora è possibile copiare o condivide re l'istantanea tra di loro. Regioni AWS Per informazioni sugli snapshot dei cluster di database di Aurora Serverles s v1, consulta Aurora Serv erless v1 e snapshot.	17 aprile 2019

Ripristino dei backup MySQL 5.7 da Simple Storage Service (Amazon S3)	Puoi ora creare un backup del database MySQL versione 5.7, archiviarlo su Simple Storage Service (Amazon S3) e quindi ripristinare il file di backup su un nuovo cluster di database Aurora MySQL. Per ulteriori informazioni, consulta Migrazione di dati da un database MySQL esterno a un cluster database Aurora MySQL.	17 aprile 2019
Condivisione di snapshot Aurora Serverless v1 tra regioni	Con Aurora Serverless v1 le operazioni con gli snapshot sono sempre crittografate. Se si crittografa l'istantanea con la propria AWS KMS key, ora è possibile copiare o condivide re l'istantanea tra regioni. Per informazioni sugli snapshot dei cluster di database di Aurora Serverless v1, consultare la pagina relativa a Serverless e snapshot Aurora.	16 aprile 2019
Tutorial Aurora proof-of-concept	Puoi imparare a eseguire un proof of concept per testare la tua applicazione e il carico di lavoro su Aurora. Per l'esercit azione completa, consultare Eseguire un proof of concept di Aurora.	16 aprile 2019

Aurora Serverless v1 supporta il ripristino da un backup Amazon S3	Puoi ora importare backup da Simple Storage Service (Amazon S3) in un cluster Aurora Serverless. Per ulteriori dettagli su questa procedura , consultare Migrazione dei dati da MySQL utilizzando un bucket Simple Storage Service (Amazon S3).	16 aprile 2019

Nuovi parametri modificabili per Aurora Serverless v1	Ora è possibile modificare i seguenti parametri del database per un cluster di Aurora Serverless v1: `innodb_file_format` , `innodb_file_per_table` , `innodb_large_prefix` , `innodb_lock_wait_timeout` , `innodb_monitor_disable` , `innodb_monitor_enable` , `innodb_monitor_reset` , `innodb_monitor_reset_all` , `innodb_print_all_deadlocks` , `log_warnings` , `net_read_timeout` , `net_retry_count` , `net_write_timeout` , `sql_mode` e `tx_isolation` . Per ulteriori informazioni sui parametri di configurazione per i cluster di Aurora Serverless v1, consulta Aurora Serverless v1 e gruppi di parametri.	4 Aprile 2019
Aurora PostgreSQL supporta le classi di istanza database db.r5.	Ora è possibile creare cluster di database di Aurora PostgreSQL basati su classi di istanza database db.r5. Per ulteriori informazioni, consultare la pagina relativa alla classe di istanza database.	4 Aprile 2019

Replica logica di Aurora PostgreSQL	Ora è possibile utilizzare la replica logica di PostgreSQL per replicare parti di un database per un cluster di database di Aurora PostgreSQL. Per ulteriori informazioni sulla replica logica di PostgreSQL, consultare Utilizzo della replica logica di PostgreSQL.	28 marzo 2019
Supporto GTID per Aurora MySQL 2.04	Ora è possibile utilizzare la replica con la funzione Global Transaction ID (GTID) di MySQL 5.7. Questa funzione semplifica la replica del log binario (binlog) tra Aurora MySQL e un database esterno compatibile con MySQL 5.7. La replica può utilizzare il cluster di Aurora MySQL come origine o come destinazione. Questa funzione è disponibile su Aurora MySQL 2.04 e versioni successive. Per ulteriori informazioni sulla replica basata su GTID e Aurora MySQL, consultare Utilizzo della replica basata su GTID per Aurora MySQL.	25 marzo 2019

Caricamento dei Aurora Serverless v1 log su Amazon CloudWatch	Ora puoi fare in modo che Aurora carichi i log del database CloudWatch per un cluster. Aurora Serverless v1 Per ulteriori informazioni, consulta Visualizzazione di un cluster database serverless Aurora. Come parte di questo miglioramento, ora è possibile definire i valori per i parametri a livello di istanza in un gruppo di parametri del cluster di database e tali valori si applicano a tutte le istanze database del cluster, a meno che non vengano sovrascritti nel gruppo di parametri del database. Per ulteriori informazioni, consulta Utilizzo di gruppi di parametri del database e gruppi di parametri cluster database.	25 febbraio 2019
Aurora MySQL supporta le classi di istanza database db.t3.	Ora è possibile creare cluster di database di Aurora MySQL basati su classi di istanza database db.t3. Per ulteriori informazioni, consultare la pagina relativa alla classe di istanza database.	25 febbraio 2019

Aurora MySQL supporta le classi di istanza database db.r5???	Ora è possibile creare cluster di database di Aurora MySQL basati su classi di istanza database db.r5. Per ulteriori informazioni, consultare la pagina relativa alla classe di istanza database.	25 febbraio 2019
Contatori di Performance Insights per Aurora MySQL	Ora è possibile aggiunger e unità di conteggio delle prestazioni ai grafici di Performance Insights per le istanze database di Aurora MySQL. Per ulteriori informazi oni, consulta Componenti del pannello di controllo di Performance Insights.	19 febbraio 2019
Amazon RDS Performance Insights supporta la visualizz azione di una maggiore quantità di testo SQL per Aurora MySQL	Amazon RDS Performan ce Insights ora supporta la visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo delle istanze database di Aurora MySQL. Per ulteriori informazioni, consultar e Visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights.	6 febbraio 2019

Amazon RDS Performance Insights supporta la visualizzazione di una maggiore quantità di testo SQL per Aurora PostgreSQL	Amazon RDS Performance Insights ora supporta la visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo delle istanze database di Aurora PostgreSQL. Per ulteriori informazioni, consultare Visualizzazione di una maggiore quantità di testo SQL nel pannello di controllo di Performance Insights.	24 gennaio 2019
Fatturazione dei backup Aurora	Puoi utilizzare i CloudWatch parametri `TotalBackupStorageBilled` di Amazon e `BackupRetentionPeriodStorageUsed` monitorare l'utilizzo dello spazio dei tuoi backup Aurora. `SnapshotStorageUsed` Per ulteriori informazioni su come utilizzare i CloudWatch parametri, consulta Panoramica del monitoraggio. Per ulteriori informazioni su come gestire lo storage dei dati di backup, consultare Informazioni sull'utilizzo dello storage di backup Aurora.	3 gennaio 2019

Contatori di Performance Insights	Ora è possibile aggiunger e unità di conteggio delle prestazioni nei grafici di Performance Insights. Per ulteriori informazioni, consulta Componenti del pannello di controllo di Performance Insights.	6 dicembre 2018
Database globale Aurora	Ora è possibile creare database globali Aurora. Un database globale Aurora si estende su più pagine Regioni AWS, abilitando letture globali a bassa latenza e disaster recovery in caso di interruzi oni a livello regionale. Per ulteriori informazioni, vedi Utilizzo di Amazon Aurora Global Database.	28 Novembre 2018
Gestione del piano di query per Aurora PostgreSQL	Aurora PostgreSQL fornisce ora la gestione del piano di query che puoi usare per gestire i piani di esecuzion e delle query PostgreSQL. Per ulteriori informazioni, vedi Gestione dei piani di esecuzione delle query per Aurora PostgreSQL.	20 Novembre 2018

Editor di query per Aurora Serverless v1 (beta)	È possibile eseguire istruzioni SQL nella console Amazon RDS sui cluster di Aurora Serverless v1. Per ulteriori informazioni, consulta Utilizzo dell'editor delle query per Aurora Serverless v1.	20 Novembre 2018
Data API per Aurora Serverless v1 (beta)	Puoi accedere ai cluster Aurora Serverless v1 con applicazioni basate sui servizi Web mediante La Data API. Per ulteriori informazioni, consulta Utilizzo dell'API dati per Aurora Serverless.	20 novembre 2018
Supporto TLS per Aurora Serverless v1	I cluster Aurora Serverless v1 supportano la crittografia TLS/SSL. Per ulteriori informazioni, consulta TLS/SSL per Aurora Serverless.	19 novembre 2018

Endpoint personalizzati	Ora è possibile creare endpoint associati a un set arbitrario di istanze database. Questa caratteristica aiuta con il sistema di bilanciamento del carico e l'elevata disponibilità per i cluster Aurora in cui alcune istanze database hanno capacità o configurazione diverse rispetto ad altre. Puoi utilizzare gli endpoint personalizzati anziché connetterti a un'istanza database specifica tramite l'endpoint dell'istanza. Per ulteriori informazioni, consulta la pagina relativa agli endpoint Amazon Aurora.	12 novembre 2018
Supporto dell'autenticazione IAM in Aurora PostgreSQL	Aurora PostgreSQL ora supporta l'autenticazione IAM. Per ulteriori informazioni, consulta Autenticazione database IAM.	8 novembre 2018

Gruppo di parametri personalizzato per il recupero e il ripristino point-in-time	Ora è possibile specificare un gruppo di parametri personalizzato quando si ripristina uno snapshot o si esegue un'operazione di ripristino point-in-time (PITR). Per ulteriori informazioni, consultare Ripristino da uno snapshot del cluster di database e Ripristino di un cluster di database a un istante temporale specifico.	15 ottobre 2018
Protezione dall'eliminazione per i cluster di database di Aurora	Quando abiliti la protezione da eliminazione per un cluster di database, il database non può essere eliminato dagli utenti. Per ulteriori informazioni, consulta Eliminazione di un cluster database.	26 settembre 2018
Funzionalità di arresto/avvio di Aurora	Ora è possibile interrompere o avviare un intero cluster di Aurora con una sola operazione. Per ulteriori informazioni, consulta Arresto e avvio di un cluster Aurora.	24 settembre 2018

Funzionalità di query parallele per Aurora MySQL	Aurora MySQL ora offre un'opzione per parallelizzare il lavoro I/O per le query nell'infrastruttura di storage Aurora. Questa caratteristica velocizza le query di analisi con un uso intensivo dei dati, which che spesso rappresentano le operazioni più gravose in termini di tempo in un carico di lavoro. Per ulteriori informazioni, consulta Parallel query for Aurora MySQL.	20 settembre 2018
Nuova guida	La prima versione della Guida per l'utente di Amazon Aurora.	31 agosto 2018

AWS Glossario

Per la AWS terminologia più recente, consultate il AWS glossario nella sezione Reference.Glossario AWS

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.